Universiteit Leiden
The Netherlands

Computer Science (A.I.)

Classification of pre-defined movement patterns:

A comparison between GNSS and UWB technology.

Rodi Laanen

Supervisors:
Dr. M. Baratchi & M. Nasri (PhD)

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

30/06/2022

Abstract

Advanced real-time location systems (RTLS) allow for collecting spatio-temporal data from human movement behaviours. These trajectories can provide insights into various applications e.g., detecting abnormal behaviour in public spaces, controlling traffic flow and monitoring children's movement patterns during recess. However, tracking individuals in small areas such as schoolyards might impose difficulties for RTLS in terms of positioning accuracy. To date, few studies have investigated the performance of different localisation systems regarding the classification of human movement patterns in small areas. Therefore, the current study aims to design and evaluate an automated framework to classify human movement trajectories obtained from two different RTLS: Global Navigation Satellite System (GNSS) and Ultra-wideband (UWB), in areas of approximately $100m^2$. Specifically, we designed a versatile framework which takes GNSS or UWB data as input, extracts features from these data and classifies them according to the annotated spatial patterns. The automated framework contains three choices for applying noise removal: no noise removal, Savitzky Golay Filter on the raw location data and Savitzky Golay filter on the extracted features as well as three choices regarding the classification algorithm: Decision Tree, Random Forest and Support Vector Machine. We integrated different stages within the framework with the Sequential Model-Based Algorithm Configuration (SMAC) to perform automated hyper-parameter optimisation. The best performance is achieved with a pipeline configuration consisting of noise removal applied to the raw location data in combination with a Random Forest model for the GNSS and in combination with a Support Vector Machine model for the UWB. Through hypothesis testing, we demonstrate that UWB performs significantly better than GNSS.

Acknowledgements

I would like to express my deepest gratitude to Dr. Mitra Baratchi and Maedeh Nasri who guided me throughout this thesis with their expertise and enthusiasm. Thereby, special thanks to Richard van Dijk for his critical insights and code reviews.

Contents

1 Introduction

Nowadays, advanced localisation and surveillance systems make it possible to capture trajectories of a plethora of moving objects including but not limited to humans, transportation modes and cargo [1, 2, 3]. For instance, individuals can log their commuting behaviours via portable Global Positioning System (GPS) devices [4] or via smartphones and smartwatches [5]. According to Zheng [2], the trajectories of moving objects can be partitioned into the following four groups: mobility of people, mobility of transportation vehicles, mobility of animals and mobility of natural phenomena. In this thesis, we are interested in the mobility of people. Mining human trajectories can provide insights into identifying potential criminals in public spaces through anomalous trajectories [5], pedestrian movement dynamics [6], point-of-interest (POI) recommendation [7], understanding periodic behavioural patterns [8], anomaly detection among personal behaviour patterns for people suffering from Alzheimer's disease [9], reducing commuting time and optimizing transport planning via transportation mode classification [10, 11], extracting social tie information [12] and tracking movement behaviours of children on a schoolyard during recess [13, 14].
A direct application for tracking young children's movement patterns is used in the schoolyard project[1]. The aim of this project is to design a more inclusive schoolyard environment for children. An important aspect of the social behaviour part of this project entails how children utilise the environment including the form of their movement patterns. Relatively small areas, such as a schoolyard, might impose difficulties for a Global Navigation Satellite System (GNSS) device in terms of positioning accuracy. Under ideal circumstances, GNSS devices can achieve a positioning accuracy of approximately 2-3 m [15, 16]. However, according to Dabove & Pietra [15], the positioning error can deteriorate up to 20 m or higher due to noise of, for instance, the reflections on the surface of the GNSS device. Several studies [17, 18, 19] have demonstrated promising results regarding the positioning accuracy through Ultra-wideband (UWB) technology. As stated in these studies, UWB technology can achieve a positioning accuracy between a 10-15 cm range as opposed to GNSS with a range between 2-20 m.
Recent studies compared the positioning accuracy of GPS and UWB data in sport fields [20, 21]. Waqar et al. [20] compared the positioning accuracy from a GPS and UWB system on a tennis field. Based on their quantitative analysis, the authors concluded that UWB outperforms GPS in terms of localisation accuracy [20]. In the study of Bastida et al. [21], the two technologies have been assessed on a soccer field. The UWB system proved to achieve higher accuracy in terms of localisation compared to the GPS system [21]. Despite the interesting results found in these two studies on positioning accuracy, they do not address movement pattern mining tasks performed through GNSS and UWB systems. Yet, to the best of our knowledge, there is no study comparing these two technologies in a movement pattern mining task (i.e., a classification problem). Additionally, little is known in terms of classification accuracy on human movement trajectories obtained via GNSS and UWB technologies. Therefore, the current thesis aims to design an automated framework to classify pre-defined human movement patterns obtained via GNSS and UWB technologies and evaluate the model's classification performance between the two technologies. Moreover, we will answer the following research questions:

1. How to design an entire pipeline to collect GNSS and UWB data, preprocess and classify them to the correct patterns?

[1] https://www.focusonemotions.nl/play/social-inclusion

2. How to select the optimum hyperparameter configurations for the data partitioning, noise removal and classification parts within the pipeline which result in the highest performance?

3. Which technology, GNSS versus UWB, performs better at movement pattern classification in small areas of approximately 100m^2?

First, in Chapter 2, we will introduce the necessary definitions and both real-time location systems (RTLS). In Chapter 3, we will focus on previously conducted research in the movement pattern mining field related to the prediction of transportation modes and clustering of movement patterns. Next, we will give a detailed overview in Chapter 4 of our versatile framework for classifying movement patterns via GNSS and UWB technology. Subsequently, in Chapter 5, we will describe the data collection part which took place under controlled conditions. In addition, we will mention the experimental setup. Finally, in Chapter 6, we will summarise and discuss the results of our conducted experiments.

1.1 Contributions

This section contains contributions in the form of bullet points that describe what we proposed (■), what we concretely did (•) and which results we obtained (▲):

- ■ Assess how to design an automated framework which takes as raw input spatio-temporal data collected with GNSS or UWB technology, pre-processes these data and afterwards classifies them according to the correct patterns.
- ■ Investigate different options to perform (automated) hyperparameter optimization within the proposed framework.
- ■ Evaluate which RTLS technology, GNSS versus UWB, performs better at pre-defined human movement pattern classification.
- • Collected a GNSS as well as a UWB data set both consisting of 104 pre-defined human movement patterns.
- • Designed an automated framework which aims to correctly classify pre-defined movement patterns gathered with GNSS and UWB technology. It contains three options for the noise removal application: no noise removal, Savitzky Golay Filter on the raw location data or Savitzky Golay Filter on the extracted features as well as three options regarding the classification model: Decision Tree, Random Forest or Support Vector Machine.
- • Performed automated hyperparameter optimization within the data preparation, noise removal and classification stages of the automated framework with SMAC.
- ▲ The highest achieved pipeline configuration for the GNSS technology consists of Savitzky Golay filter on the raw location data in combination with an RF model.
- ▲ The highest achieved pipeline configuration for the UWB technology consists of Savitzky Golay filter on the raw location data in combination with an SVM model.

- By comparing these two configurations through hypothesis testing, we demonstrate that, with a significance level of 99%, UWB performs significantly better than GNSS on correctly classifying pre-defined movement trajectories.

2 Background Theory

This chapter describes the fundamental theory on which we build our pipeline configurations. First, we will provide several definitions. Then, we will give the necessary information to understand both technologies, including the advantages and disadvantages of the two positioning systems. Next, we will introduce three classification algorithms. Last, we will introduce an existing application for automated HPO [22] named Sequential Model Algorithm Configuration (SMAC) [23].

2.1 Definitions

Discrete trajectory [24] represents, for a time interval $[t_{Start}, ..., t_{End}]$, a finite series of spatio-temporal records $[1, R]$ related to the movement of a certain object. In this respect, the discrete trajectory of object o can be described as follows: $Traj_o = [r_{t_{Start}}, ..., r_{t_{End}}]$, where $r \in R$ represents a single record containing spatio-temporal information of object o at time step r_t.

Segmented trajectory, adapted from [24], stands for the time-wise division without disrupting the continuity of a discrete trajectory of object o given by:
$Sub\_traj_o = [r_{t_i}, r_{t_{i+1}}, ..., r_{t_N}] \subseteq Traj_o$, in which the range $[i, N]$ is smaller than $[1, R]$. A discrete trajectory might be divided into multiple segmented trajectories.

Two-point Features, adapted from [10], describe the movement characteristics of a discrete trajectory based on the logged spatio-temporal data of two consecutive time steps t_i and t_{i+1} within time interval $[t_{Start}, ..., t_{End}]$.

Trajectory features [10] denotes, in a statistical manner, information about the entire discrete trajectory, referred to as global trajectory features, or about a subpart of the discrete trajectory, considered local trajectory features.

2.2 Positioning Systems

2.2.1 Global Navigation Satellite System

The majority of mobile devices, such as smartphones and tablets, contain consumer-grade GNSS chips [5, 15, 25]. These devices allow individuals to position themselves through the radio signals received from, for instance, GPS and Beidou constellations. To establish the position of a user, GNSS uses one-way Time of Arrival (TOA) ranging [26]. TOA works as follows [26]: GNSS satellites, having synchronised clocks, transmit two unique messages, including the location information of a satellite and ranging codes, in the form of radio signals [26]. The former allows the GNSS device to establish the position of a satellite-based on the emission time of the respected radio wave. The latter contains information regarding the time span between emitting and receiving the signal. By multiplying the signal propagation time by the speed of light, the GNSS device can compute the distance between a certain satellite and itself [26]. In other words, the satellite serves as a reference point to determine the relative distance between itself and the device.
However, a single distance between a GNSS device and a satellite is not sufficient to establish the position of the GNSS device on the earth's surface. To determine a two-dimensional position via multilateration, the GNSS device needs to interpret the radio signals of at least three satellites

referred to as S_1, S_2 and S_3 [26]. The computed distance between the device and satellite S_1 serves as the radius of a circle and positions the device somewhere on earth. Likewise, the range between the device and satellite S_2 locates the device somewhere on earth. The resulting two spheres intersect which means that the device needs to be at one of the two intersection points. However, the device's internal clock might inhibit some offset compared to the internal atomic clocks of the satellites S_1 and S_2 [26]. This potentially results in ranging errors that negatively affect the accuracy of the positioning of the device. Namely, the two intersecting points could become two intersecting regions. Therefore, a third satellite S_3 is involved in the process to compensate for this clock offset [26]. By resolving the uncertainty in terms of time error via satellite S_3, we obtain accurate ranges between the device and the two satellites S_1 and S_2. Thereby, the range between the device and satellite S_3 creates a third sphere which can be used to determine the most likely point of the two intersection points. The resulting outcome represents the respective position of the device [26].
An advantage of using GNSS is the fact that each individual has unlimited free access to this technology anywhere on earth [26]. Thereby, consumer-grade GNSS chips are considered low-cost [25]. However, a potential downside of GNSS is its performance within smaller areas. Namely, a device equipped with a GNSS chip can provide positioning accuracies between 2-20 m [15, 16]. This accuracy range might potentially result in unreliable trajectories within smaller areas. Furthermore, inaccurate positioning might result from time-delayed signals being reflected from surfaces, a phenomenon called multipath [26]. Additionally, a radio wave might fade due to ionospheric scintillation [26].

2.2.2 Ultra-wideband

Another RTLS is UWB. Similarly to GNSS, UWB uses radio signals for positioning a device. According to Sahinoglu et al. [27] the difference between the upper frequency and the lower frequency of the radio signal needs to be at least 500 MHz in order to categorise a signal as UWB. A company named Pozyx designed two systems which utilise such large bandwidths for real-time positioning: a Creator Kit for small experimental projects and an Enterprise Kit for more demanding industrial projects [28]. Since we used the former system in this study, we will solely focus on the working principles of the Pozyx's Creator Kit [29] in the remainder of this section.
The Pozyx's Creator Kit includes five anchors, four developer tags and the necessary cables and adapters to provide the anchors as well as the tags with constant power [29]. Aside from the hardware part, Pozyx also delivers a cloud app named the Pozyx Creator Controller. The Pozyx Creator Controller enables the user to verify the hardware setup, create a local Cartesian coordinate system and visualise the real-time positioning of the active tags within the local coordinate system. Regarding the positioning protocol, the Creator Kit makes use of single-sided Two-Way Ranging (TWR) in combination with multilateration. [29]. The coordinates (x, y, z) of at least four anchors serve as reference points and form the local Cartesian coordinate system. In other words, the locations of active tags within this local Cartesian coordinate system are determined relative to the coordinates of the anchors [28].
For TWR, the communication between an active tag and at least four anchors plays an important role in defining the relative distances between them [30], represented in Figure 1:

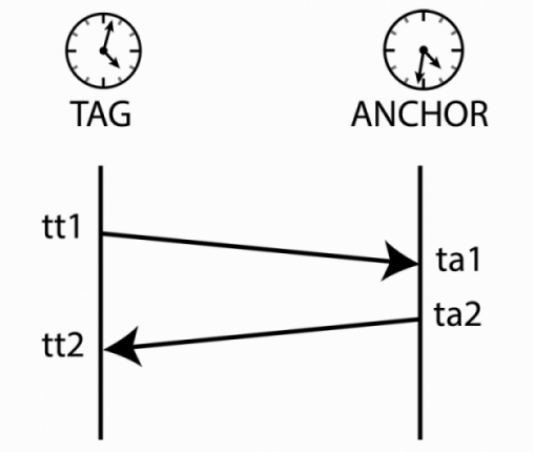


Figure 1: Two-way-ranging protocol between an active tag and an active anchor with asynchronous clocks that allows for computing the Time of Flight [28].

As becomes clear from Figure 1, an active tag initialises the communication by sending a message in the form of a radio signal, $tt1$. The anchor, in its turn, records the time of arrival, $ta1$ and replies by transmitting a message back to the active tag, $ta2$. This message includes the time of arrival of the received message and the time it took to process and respond. In the meantime, the tag keeps track of the time between sending the initial message and receiving a response, $tt2$. The information based on the two messages allows the active tag to compute the Time of Flight (ToF) in seconds, representing the time it took to transmit and receive a radio signal. Via the ToF, an active tag can calculate the relative distance between itself and the anchor in question via the following equation (1) [31]:

$$d = ToF \times q \tag{1}$$

where q represents the speed of light ($q = 299792458$ m/s), and d denotes the distance in meters. Furthermore, the clocks of the anchors do not need to be synchronised to successfully perform TWR [30].
In a two-dimensional local Cartesian coordinate system, the positioning of an active tag can be established via at least three other anchors [28]. Similarly to GNSS, the ranges between an active tag and three anchors represent the radii of three circles. The intersection of these three circles defines the position of the active tag, visualised in Figure 2:

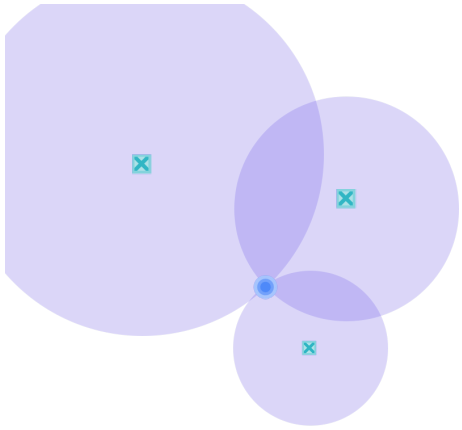


Figure 2: Determining the two-dimensional position of an active tag, the blue circle, by means of three active anchors represented by the green crosses via multilateration [28].

However, Pozyx [28] states that the determination of ToF is never perfect due to noise involved in the measurements. Therefore, the resulting three circles might not perfectly intersect at one point due to possible deviations of the radii. Pozyx's system solves this problem by selecting an intersection that yields the shortest radius for all involved anchors [28].
An advantage of using UWB for real-time positioning is its reported high accuracy of approximately 10-15 cm [17, 18, 19, 29, 32]. Furthermore, the UWB signals are to a certain degree robust against signal interference due to multipath [18, 28]. A drawback of UWB is the necessity for the hardware setup and constant power supply. Unlike GNSS, the anchors yield a relatively small coverage area for the positioning of an active tag. In order to cover larger areas, more anchors are needed which also increases the cost of this respective system. Thereby, the user capacity of this system is limited in the sense that it can only track up to five active tags within a local Cartesian coordinate system [29]. Another disadvantage of the Pozyx Creator Kit is that the positioning protocol involves single-sided TWR. Single-sided TWR might be affected by clock drift resulting in less accurate positioning. Potential clock drift could be mitigated via double-sided TWR [33, 30] in which both the active tag and the anchor send independent initial messages to each other. The time differences between the messages in terms of transmitting and receiving are averaged to obtain the ToF.

2.3 Classification Algorithms

In the next three subsections, we will discuss the following supervised machine learning algorithms for classification: Decision Tree, Random Forest and Support Vector Machine. We opted for these three algorithms because they are popular machine learning classifiers and can be used for, or in the case of a Support Vector Machine extended to, multi-class classification problems [34, 35, 36]. In addition, the classification stage in our proposed automated framework is based on the respective stage in the framework proposed by Etemad et al. [10] in which they used a Decision Tree and a Random Forest model.

2.3.1 Decision Tree

A Decision Tree (DT) is a supervised machine learning algorithm that can handle classification and regression tasks. The core idea of a DT is to perform splits on the feature set by making binary decision rules. In this respect, a DT aims to divide the data set into smaller groups referred to as nodes. The ultimate goal is to minimise the diversity within the leave nodes by making these splits [36]. Since it is a supervised algorithm, the learned model should be able to predict the target variable of an instance via the decision rules.
A split consists of partitioning a parent node into two child nodes. In the case of a binary decision problem, a pure split describes a feature that can flawlessly distinguish between the positive and negative labels [36]. Hence, a feature split should result in minimising the impurity in the two child nodes compared to the impurity in the parent node [36]. Several measures e.g., gini impurity and entropy, exist to interpret the quality of a split on the feature set [36].

2.3.2 Random Forest

In essence, a Random Forest (RF) uses an ensemble technique referred to as bagging [36]. Bagging is applied during the training phase to create multiple tree models based on different random subsets of the provided training set. In such a manner, bagging provides diversity among decision tree models. The final model can be constructed via majority voting or averaging over probability scores [36].
Analogously to a DT model, the trees generated in RF perform split on the feature set. However, the number of features considered for a specific split can alter depending on the task. For instance, Pedregosa et al. [37] argue, from an empirical standpoint, that a random sample with a size equal to the square root of the total number of features is an acceptable number of features for classification tasks.

2.3.3 Support Vector Machine

Support Vector Machine (SVM) is a classification algorithm that is inherently designed for binary classification tasks [36]. Given a linearly separable data set, an SVM tries to find a hyperplane which can differentiate between the two sets by maximising the margin [36]. Nevertheless, this machine learning model can be extended to be applicable in a multi-class setting via two heuristics: one-vs-one and one-vs-rest.
Both methods convert a multi-class classification task into several binary classification tasks. One-vs-one evaluates all possible combinations of pairs without repetition. More formally, b designating the number of binary classification tasks from splitting a multi-class classification task via one-vs-one is given by equation (2):

$$b = \frac{K \cdot (K-1)}{2} \tag{2}$$

where K refers to the number of classes. On the other hand, a one-vs-rest approach considers $b = K$ binary classification tasks. Therefore, a one-vs-rest method is often referred to as the winner takes it all strategy [38].

2.4 SMAC

The aforementioned classification algorithms in Section 2.3 have in common that their internal learning process regarding a classification task depends on hyperparameters [22, 39]. In other words, hyperparameters can guide this learning process of a model. Therefore, it is important to identify the optimal set of hyperparameter values that maximise the performance of a model [22, 39, 40]. This method is referred to as hyperparameter optimization (HPO) [22].
However, HPO can be a time-consuming task if one needs to manually analyze different combinations of hyperparameters within, for instance, a machine learning pipeline [22]. Additionally, one needs to have a thorough understanding regarding the impact of each hyperparameter on the model's performance and the respective ranges to pick values from [22]. Sequential Model Algorithm Configuration (SMAC) [23] can alleviate the human workload by automating the HPO process via Bayesian Optimization (BO). In more detail, the main component of SMAC consists of a surrogate model in the form of a RF [22]. Furthermore, SMAC inherently supports different hyperparameter values (e.g. integer, float and categorical). Thereby, conditionalities [22] give control over the activation of hyperparameters. In other words, there can exist a hierarchy among hyperparameters meaning that parent hyperparameters can activate or deactivate children hyperparameters.
On the other hand, there also exist automated machine learning (AutoML) frameworks which target more steps in the machine learning pipeline than only HPO. Apart from HPO, AutoML decides upon the possibility to perform certain preprocessing methods on the data set and chooses among a variety of machine learning algorithms [41]. Additionally, AutoML can construct ensemble models [41]. Auto-sklearn [41] is an example of an AutoML framework. Auto-sklearn samples a machine learning pipeline based on 15 classification algorithms and 14 preprocessing methods [41]. A drawback of using, for instance, Auto-sklearn is that these building blocks limit the range of options to create a machine learning pipeline. SMAC mitigates this limitation since its core functionality centres around performing automated HPO on an existing framework. In this respect, one can design a very specific pipeline configuration and utilise SMAC to optimise the performance of this pipeline via automated HPO.

3 Related Work

In this chapter, we will discuss two studies related to predicting different modes of transportation and one study which focuses on clustering movement patterns. The proposed frameworks in the two studies about predicting transportation modes contain useful stages that we could adopt in our automated framework. The clustering of movement patterns study is relevant since the artificial created movement patterns can be mimicked by human-beings throughout our data collection sessions.

3.1 Prediction of Transportation Modes

Prediction of transportation modes via GPS trajectories is studied by Etemad et al. [10]. The authors propose a competitive framework which is capable of predicting different modes of transportation, demonstrated in Figure 3:

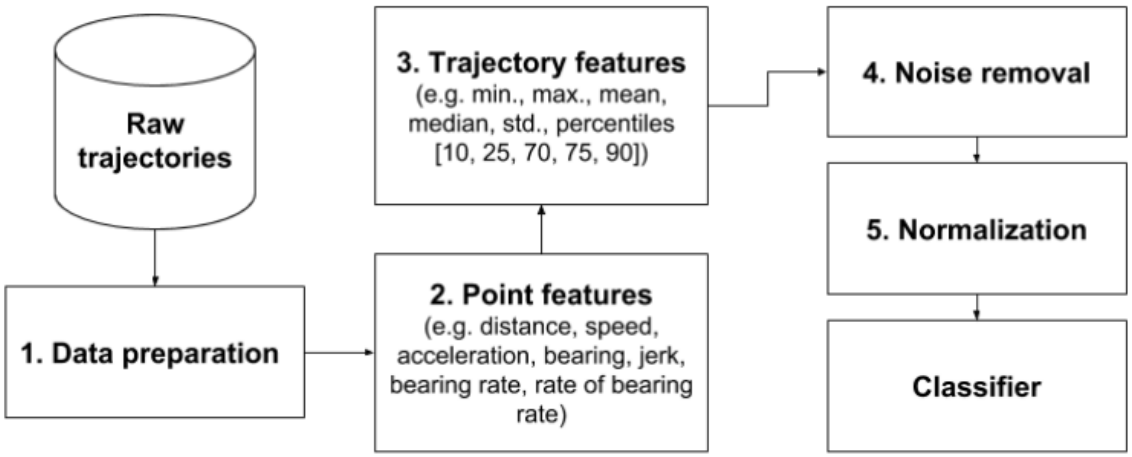


Figure 3: Proposed pipeline of Etemad et al. [10] to predict transportation modes based on GPS trajectories.

As we can see in Figure 3, the pre-processing phase of the pipeline consists of five distinct phases. Phases two and three are concerned with computing features based on GPS trajectories. In more detail, Etemad et al. [10] first compute the following point features per trajectory: distance, speed, acceleration, jerk, bearing, bearing rate and the rate of bearing. These features are grouped under point features since they are computed per pair of consecutive GPS records. Next, five global and five local trajectory features are computed for each of the seven point features. The global trajectory features include the minimum, maximum, mean, median and standard deviation and provide information about an entire trajectory. The local trajectory features describe local parts within a trajectory and comprise five percentiles (10, 25, 50, 75, 90). Thus, a single trajectory is defined by 7 point features × 5 global trajectory features × 5 local trajectory features resulting in a total of 70 features.
Etemad et al. [10] trained five classifiers: Decision Tree, Random Forest, Neural Network, Naive Bayes and Quadratic Discriminant Analysis on five sub-selections of the GeoLife GPS data set [4]. The authors obtained the highest results with the Random Forest model, giving an average accuracy

score of 93.36% and an average F_1 score of 93.16% based on the five data sets. Furthermore, this study highlights the advantage of applying noise removal in transportation mode prediction by reporting on performance augmentations between 3.36 and 29.04 for the accuracy score and 2.56 and 28.15 for the F_1 score.
Phases two and three in the proposed framework of Etemad et al. [10] are meaningful for our pipeline because they serve to extract features from raw trajectory records containing spatio-temporal information. However, this study lacks motivation for the used hyperparameter values throughout the framework. Performing automated HPO over the pipeline could result in higher classification performance. Additionally, the model could be more versatile with respect to predicting transportation modes through discrete GPS trajectories from other data sets.
The research of Dabiri & Heaslip [11] also addresses the identification of transportation modes by means of discrete GPS trajectories from the Geolife data set [4]. Their work differs from the work of Etemad et al. [10] since Dabiri & Heaslip utilise a Convolutional Neural Network (CNN) instead of traditional classification algorithms. Furthermore, the noise removal part within their pipeline can be divided into two sub-parts. The first part consists of detecting anomaly GPS records based on certain thresholds such as a maximum speed associated with each transportation mode and a minimum number of GPS points to form a discrete trajectory. The second phase involves smoothing each discrete trajectory via a Savitzky Golay Filter. Dabiri & Heaslip opted for the Savitzky Golay filter since this smoothing kernel does not impose a specific distribution of the signal and maintains the original shape of the signal after the smoothing process [11].
Dabiri & Heaslip [11] extracted the discrete GPS trajectories of 69 individuals from the GeoLife data set [4] representing five transportation modes: walk, bike, bus and train. Via an ensemble model, the authors achieved an accuracy score of 84.8%. The proposed smoothing kernel is interesting since the Savitzky Golay Filter manages to smooth a signal while maintaining the original shape [11]. However, this research does not justify the respective hyperparameter settings for the smoothing kernel. Including automated HPO for the Savitzky Golay Filter might prove beneficial for the classification performance of the ensemble model.

3.2 Clustering of Movement Patterns

Trajectory clustering via deep representation learning is presented in the work of Yao et al. [42]. Their trajectory clustering technique differentiates itself from other methods since it is capable of clustering similar discrete GPS trajectories occurring in different locations and at different time periods [42]. Per pair of two successive GPS records within a single discrete trajectory, the authors compute the average speed, change of speeds and the change of rate of turn (ROT). Next, six statistics including the mean, max, min, 25% quantile, 50% quantile and 75% quantile are computed over these three quantities which represent the features describing a discrete trajectory. Yao et al. [42] assessed their clustering framework on different data sets including a synthetic one. The synthetic data set consisted of three movement patterns: {*Straight, Circling, Bending*} and six combinations: {*Straight + Circling, Straight + Bending, Circling + Bending, Circling + Straight, Bending + Straight, Bending + Circling*}, each represented by 1000 instances [42]. Through a K-means algorithm with K set equal to 9 (i.e., the number of different clusters), the authors achieved an accuracy of 64.80% with a sequence-to-sequence auto-encoder composed of an LSTM.
The generated artificial movement patterns in this work resemble basic movement patterns that humans can mimic by walking. This means that we could collect similar movement patterns in our

study. Furthermore, the average speed, change of speeds and change of ROT are quantities that can be extracted from both GNSS and UWB trajectories. Since these three features describe the artificially generated movement patterns, we could adopt these features for the movement patterns collected in our study. Despite the assessment of different hyperparameters for the deep learning network, the authors did not report on tuning the hyperparameters within the pre-processing phase of their pipeline. In addition, Yao et al. [42] computed the average speed, change of speeds and change of ROT without converting the latitude and longitude to a Cartesian Coordinate system. Since latitude and longitude are often expressed in degrees concerning the earth's radius, we need to transform them into a linear distance unit before calculating the aforementioned quantities which assume a distance-based metric.

4 Methodology

In this chapter, we will present the general outline of the automated framework that we designed in the current study (see Figure 4). The following parts of the automated framework will be discussed in more detail: Raw Trajectories, Data Preparation, Two-point Features, Trajectory Features, Noise Removal and Normalization. Additionally, we will compare our work with the previously mentioned works.

4.1 Automated Framework

The automated framework that we designed in this study is represented in Figure 4:

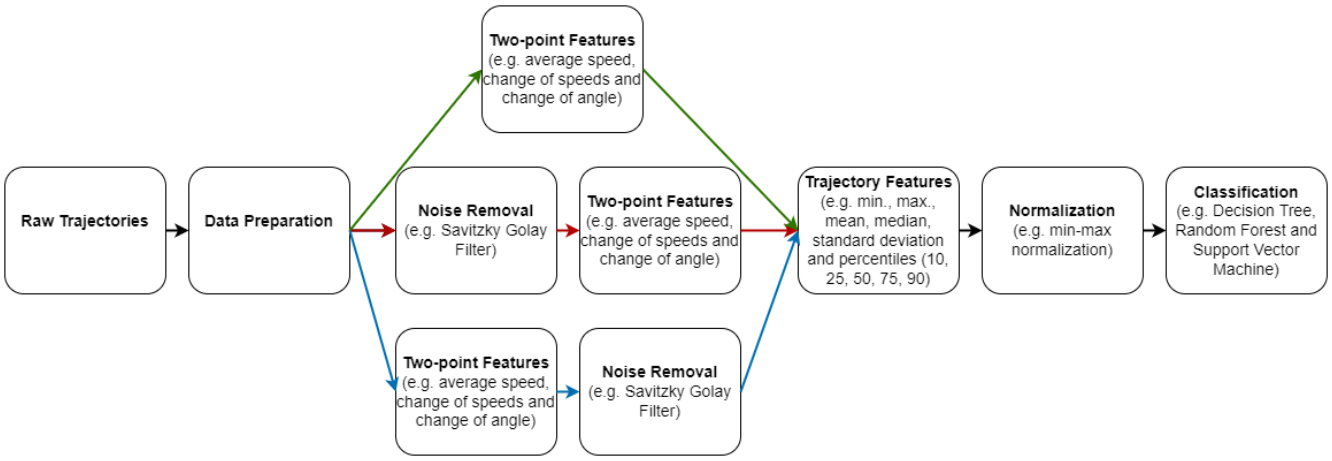


Figure 4: Automated framework which takes raw input data of both GNSS and UWB systems, pre-processes the spatio-temporal data and classifies them according to the correct patterns. The green line denotes no noise removal application, the red line indicates Savitzky Golay Filter on the location data and the blue line indicates Savitzky Golay Filter on the Two-point Features. Within this framework, we automatically optimize hyperparameters present in the Data Preparation, Noise Removal and Classification stages, adapted from [10].

As becomes apparent from Figure 4, the architecture of the automated framework is based on the proposed framework of Etemad et al. [10]. In more detail, we adopted the seven stages: Raw Trajectories, Data Preparation, Point Features, Trajectory Features, Noise Removal, Normalization and Classifier from Etemad et al. [10]. Since, from a high-level perspective, the framework of Etemad et al. [10] is capable of accepting raw discrete GPS trajectories as input, pre-processes the spatio-temporal data of each discrete trajectory and feeds the extracted features to a classifier, we utilised the seven building blocks of this framework as a basis for our automated framework. In other words, from a high-level perspective, our automated framework is expected to conduct the three previously mentioned tasks.
Furthermore, as a baseline, we included a pipeline configuration yielding no noise removal which is represented by the green line in Figure 4. However, Etemad et al. [10] stress on the importance of using noise removal within the framework to improve the performance. For this reason, we included a pipeline configuration consisting of applying a Savitzky Golay Filter on the raw location data

after which we compute the Two-point Features, denoted by the red line in Figure 4. The blue line in Figure 4 refers to a pipeline configuration consisting of extracting Two-point Features from the raw data followed by a Savitzky Golay Filter on the Two-point Features. Given that this specific sequence is present in the framework proposed by Dabiri & Heaslip [11], we included it in our framework to assess differences with other possible pipeline configurations. In the remaining subsections, we will explain each respective stage, except the Classification part, of the automated framework in more detail.

4.1.1 Raw Trajectories

The raw GNSS data per pattern recording is saved in a csv format file. This file contains information about the latitude and longitude with the corresponding timestamp in Epoch time. The raw UWB data of a single pattern recording in the form of time-ordered JSON packets are recorded into a text file. Each packet contains basic information about a single active tag. Since the Pozyx Creator Kit [29] demands at least two active tags (i.e., a master tag and a puppet tag), the mentioned text file contains packets from all active tags. The mentioned information per packet includes the ID of the tag, whether or not the tag was alive, whether or not the position of the tag could successfully be established and the timestamp in Epoch time. In case both alive and success evaluate to true, the basic information is supplemented with the latency, update rate and success rate. Furthermore, the Pozyx Creator Controller [28] allows for additional information in each packet. In our case, we added the coordinates $(x,\ y)$ of the active tags.

4.1.2 Data preparation

The GNSS data files, each one containing a single discrete trajectory, do not need any further preparation. However, the UWB data files containing the time-hierarchically structured packets of all active tags do require additional preparation to partition the successful packets according to the associated active tag. First, we distinguish between successful packets and unsuccessful packets. Each packet contains a key named *success* which indicates if the position of an active tag could be established. Given that the spatial information of an active puppet tag is crucial in our pipeline to extract the necessary features, we reject all packets in which the key *success* is equal to the Boolean value *False*. Second, we format the hierarchical JSON structure to a flat pandas DataFrame structure which allows for direct access to the raw spatio-temporal data. In addition, the pandas DataFrame is saved to a csv format file. Lastly, we split the created pandas DataFrame based on the value of *tagId* since we are only interested in the raw spatio-temporal data of the active puppet tag. Furthermore, in this part of the automated framework, we included a hyperparameter that accounts for partitioning a discrete trajectory into segmented trajectories.

4.1.3 Two-point Features

In the third step of our automated framework, we compute three Two-point Features: average speed, change of speeds and change of angle for each discrete trajectory. As explained in Section 3.2, these quantities can be computed for both GNSS and UWB trajectories and were used in the work of Yao et al. [42] to describe the artificial generated movement trajectories in terms of features.

Therefore, we decided to use these three Two-point Features. Each record within a discrete GPS trajectory contains the following information: latitude, longitude, and timestamp. Based on the location and timestamp of each record, we can compute the average speed via:

$$s_r = \frac{haversine((lat, lon)_{r+1}, (lat, lon)_r)}{t_{r+1} - t_r} \tag{3}$$

in which s represents the average speed, lat denotes the latitude, lon stands for longitude, $haversine$ determines the circular distance between two consecutive records, t describes the timestamp, $r \in [2, R]$ indicates a single record of a discrete trajectory and R represents all records ranging from $[1, R]$ describing a discrete trajectory. The average speed for $r = 1$ is set equal to 0. In equation 3, the haversine function[2] returns the circular distance between two points represented by latitude and longitude on earth. We computed the angle between two consecutive records by:

$$a_r = arctan2(haversine(lat_{r+1}, lat_r), haversine(lon_{r+1}, lon_r)) \tag{4}$$

in which a denotes the angle and $r \in [2, R]$. The angle for $r = 1$ is set to 0. In equation 4, we used the arctan2 since the latitude ranges between -90 and 90 degrees and the longitude ranges between -180 and 180 degrees.

As for each discrete UWB trajectory, a single record contains the position of an activate tag in (x, y) coordinates and a corresponding timestamp. We computed the average speed according to the following equation:

$$(x, y)_r = \sqrt{(x_{r+1} - x_r)^2 + (y_{r+1} - y_r)^2} \tag{5}$$

$$s_r = \frac{(x, y)_r}{t_{r+1} - t_r} \tag{6}$$

in which x denotes the x coordinate, y describes the y coordinate and $r = [2, R]$. Again, we set $r = 1$ equal to 0. Regarding the angle, we applied the following equation:

$$a_r = arctan2((y_{r+1} - y_r), (x_{r+1} - x_r)) \tag{7}$$

in which $r \in [2, R]$. Again, we set the angle at $r = 1$ equal to 0. Given that we used the arctan2 to compute the angle for the GPS record, we used the arctan2 on the UWB records for the sake of consistency.

Regarding the three Two-points features: average speed, change of speed and change of angle, we computed the average speed for the discrete GPS trajectories with formula 3 and for the discrete UWB trajectories with formulas 5 and 6. For both the discrete GPS and UWB trajectories, we computed the change of speeds for two consecutive records by:

$$s_{\Delta_r} = s_r - s_{r-1} \tag{8}$$

[2] https://pypi.org/project/haversine/

in which s_Δ denotes the change of speed for $r \in [2, R]$ records. In addition, we calculated the change of angle for two succeeding records via:

$$a_{\Delta_r} = a_r - a_{r-1} \tag{9}$$

in which a_Δ represents the change of angle for $r \in [2, R]$. For both s_{Δ_r} and a_{Δ_r}, we set $r = 1$ to 0.

4.1.4 Noise removal

Given that both RTLS systems work with electromagnetic waves to establish the positioning, it might occur that the positioning is not accurate due to e.g., possible interference with conductors, air humidity, multipath interference, ionospheric scintillation, time-drifting and battery levels. Therefore, a discrete trajectory risks containing records with incorrect position information, making a discrete trajectory incoherent. For this reason, we decided to include a noise removal filter in our automated framework. The automated framework contains three options for the application of the noise removal filter: no noise removal, noise removal on the raw location data and noise removal on the computed Two-point Features according to the presented framework in the research of Dabiri & Heaslip [11].
The noise removal algorithm that we decided to use is called the Savitzky Golay Filter [43]. The Savitzky Golay Filter uses a sliding window in combination with a convolution process [44]. This filter targets a subset of data points defined by the target point plus the points left of the target point within the given window size and the points right of the target point within the given window size. Therefore, the length of the symmetric sliding window is equal to $2W + 1$ [44], where M represents the window size. For each such target point, a polynomial is fit onto this subset of data points. As a final step, the target point gets substituted for the computed outcome of the polynomial within the interval of the target point [44]. Two crucial hyperparameters of the Savitzky Golay Filter are the size of the sliding window and the degree of the fitted polynomial. Given that a symmetric sliding window of length $2W + 1$ is used to derive the coefficients of a polynomial [44], the size of the window needs to be odd. Regarding the order of the polynomial, it is restricted in the sense that it needs to be lower than the size of the sliding-window.
Another promising smoothing filter is the Kalman Filter [45]. However, according to Kennedy [46], the Kalman Filter can only find an optimal solution if the noise within a signal is normally distributed. If this assumption is violated, the solution is no longer optimal [46]. Given that the shape of the underlying noise distribution of the raw spatio-temporal data or a computed discrete trajectory feature is unknown [11], the Kalman Filter might not give optimal results in terms of noise removal application. The Savitzky Golay Filter mitigates this issue since it does not assume a certain distribution of the noise of a signal [11, 46]. Other advantages are that the pattern of the raw signal will be conserved in the smoothed signal after applying the Savitzky Golay Filter [11] and it yields a low computational complexity [46].

4.1.5 Trajectory features

The framework of Etemad et al. [10] makes a distinction between global trajectory features and local trajectory features. Global trajectory features denote values that yield information about

an entire trajectory. Local trajectory features, on the other hand, describe only a subpart of a trajectory. Following the approach of Etemad et al. [10], we compute the following global trajectory features for every Two-point feature: minimum, maximum, mean, median and standard deviation. As for the local trajectory features, we calculate five different percentiles (10, 25, 50, 75, 90) for each point feature.
As described in Section 4.1.3, we compute three Two-point Features: average speed, change of speed and change of angle. For these three Two-point Features, we compute five global trajectory features and five local trajectory features. Thus, this gives us a total of $3 \times (5 + 5) = 30$ features representing a single discrete trajectory.

4.1.6 Normalization

Given that the generated features differ in range from each other, we need to bring them to an equal range. This can be achieved via min-max normalization, following the framework of Etemad et al. [10]. Min-max normalization transforms a feature set into the range $[0, 1]$ by mapping the lowest value to 0, the highest value to 1 and the remaining values between 0 and 1. In this way, the relationship between the original feature set values is preserved [47].

4.2 Comparison with Other Works

In Chapter 1, we mentioned that, to the best of our knowledge, there is no study that compares GNSS versus UWB technology in a pattern mining task. Nevertheless, the study of Yao et al. [42] most closely resembles our current study since they assessed a clustering framework on an artificially generated data set consisting of basic movement patterns. In our current study, we adopted some of the basic movement patterns for the data collection sessions and included the three features: average speed, change of speeds and change of angle, from their proposed framework in our automated framework. Furthermore, we were inspired by the proposed frameworks by Etemad et al. [10] and Dabiri & Heaslip [11]. In this respect, we utilized the different stages of the framework of Etemad et al. [10] as the basic building blocks for our automated framework and included the Savitzky Golay Filter adressed in the framework of Dabiri & Heaslip within our optional noise removal stage.
However, the three studies do only consider GNSS technology whereas our study is focused on both GNSS and UWB technologies. Apart from the difference in RTLS technologies, there is a general lack in these studies regarding the motivation of hyperparameter values through different stages within the frameworks. In our study, we introduce SMAC to perform automated HPO on the different stages that contain hyperparameters. A final difference is that the application of our study is focused on classifying human movement patterns through GNSS and UWB technology. Thus, by building upon the previously mentioned studies [42, 10, 11], we propose a versatile framework which accepts both GNSS and UWB trajectories and classifies them according to the correct movement patterns. Thereby, we automate the HPO through the use of SMAC.

5 Experiments

In this chapter, we will first focus on the data collection session which took place under controlled conditions. Second, we will explain in more detail the used performance metrics serving to assess the respective performance of each pipeline configuration. Third, we will clarify the experimental settings including the bootstrapping protocol. Finally, we will report and interpret the obtained results via our conducted experiments. The source code from this study is available on GitHub[3].

5.1 Data Collection

During the multiple data collection sessions under controlled conditions, we made use of a GNSS positioning system and a UWB positioning system. For the GNSS positioning system, we used a Samsung Galaxy Tab A7 (2020) SM-T500 64GB tablet [48]. Regarding the UWB positioning system, we utilised the Pozyx Creator Kit [29]. In the remaining subsections, we will discuss the data collection procedure and the resulting two data sets in more detail.

5.1.1 Locations

The data collection, divided over six sessions, took place at two different locations. The first location was a playground situated in a neighbourhood in Leiden. At this location, we used a rectangular area marked by four trees. We chose this particular area since there were no playground structures or other objects within or near this area that could be considered as conductors (e.g. metal or water) for the electromagnetic waves of the Pozyx system [28]. Thereby, the lack of playground structures made this area less attractive for children, which minimised the possibility of disruptions during the data collection. The second location was a private garden. Similarly to the first location, we went for a rectangular space that could be identified through the general layout of the garden. The only object present within this area was a wooden table on which we placed the laptop, including the master tag. Since wood can be considered as an insulator for the signals of the Pozyx system [28], any possible interference with the table should have resulted in a negligible impact on the positioning of an active tag. Given that the garden was exclusively accessible to us, the factor of possible disruptions during the data collection was minimised.

5.1.2 Placement of the Anchors

As described in Section 2.2.2, the Pozyx Creator Kit needs at least four active anchors with a constant power supply [28]. Therefore, we connected them to power banks. Regarding the placement of these anchors, Pozyx recommends considering the following rules of thumb [28]:

1. “Place the anchors high and in line of sight of the tag(s)”.
2. “Spread the anchors around the area, never all on a straight line”.
3. “Keep the distance between two anchors in a range of 2 - 20 meters”.
4. “Keep the anchors at least 20 cm away from metal”.

[3] https://github.com/rlaanen/uwbtrajectorypatterns

5. “Place the anchors vertically with the antenna at the top or bottom”.

Given the first and fifth rule, we decided to attach the anchors to trees at the playground and at wooden fences in the garden, always with the antenna pointing at the sky. As explained in Section 5.1.1, there were no objects that could disrupt the line of sight between anchors. In this respect, we also ensured that the anchors were not in direct proximity to any form of metal material, conforming to the fourth rule. The four anchors were sufficiently spread around each area by forming rectangular areas. As for the third rule, we manually measured the distances between the anchors to confirm that the distance between a set of anchors was between 2 and 20 meters. For the manual measurements, we used a Bosch GLM 500 Professional laser distance meter which yields an accuracy of ± 3.0 mm with a potential additional deviation of ± 0.15 mm/m due to potential low reflectivity of the target caused, for instance, by the sun. The use of a laser distance meter for the anchor setup is also recommenced by Pozyx [28].
Since the anchors serve as reference points for the location determination of active tags, it is essential to know their exact locations in terms of *(x, y, z)* coordinates. In our experiment, we are only interested in the positions of active tags in a 2-dimensional Cartesian coordinate space (i.e.,, (x, y) coordinates). Therefore, we placed all four anchors at a similar height of approximately 1500 mm (i.e.,, the z coordinate of each anchor is approximately 1500 mm) measured from the ground until the middle point of the anchor with the laser distance meter.
Regarding the (x, y) coordinates, the Pozyx Creator Controller provides an inbuilt auto-calibration function for determining these two coordinates for every anchor in a local Cartesian coordinate system. However, it often resulted in a mirrored and flipped representation of the real-world setup. Additionally, this function required a manual determination of an origin by clicking on one of the four corners of the area which resulted in an inaccurate origin. To mitigate these obstacles in combination with the Pozyx's recommendation to conduct the anchor setup with a laser distance meter [28], we decided to manually measure each side of the quadrilateral areas with the above-mentioned laser distance meter. By assigning the (x, y) coordinates of one of the anchors to (0, 0) respectively (i.e., placing this anchor at the origin of the area), we were able to compute the remaining (x, y) coordinates of the other three anchors. The coordinates of each anchor can be entered in the Pozyx Creator Controller from which the system understands the relative dimensions of the area.
Thus, during the first data collection session at each location, we accurately measured the distances between the anchors with a laser distance meter. Then, we made a high-level floor plan including these measurements. The location-dependent floor plans made it easy to re-build the local coordinate system at the corresponding location for a new session. Nevertheless, we consistently measured the height during each new session with the laser distance meter.

5.1.3 Tablet and Tag settings

The tablet representing the GNSS positioning technology contains a chip that can receive radio signals from GPS, Glonass, Beidou and Galileo satellite positioning systems [48]. The tablet operates on its build-in battery power. Furthermore, the tablet uses Wi-Fi and Bluetooth signals to obtain better positional accuracy. However, during the six data collection sessions, the tablet was not connected to a Wi-Fi or Bluetooth signal. Additionally, we utilised the Sensor Logger app[4] which

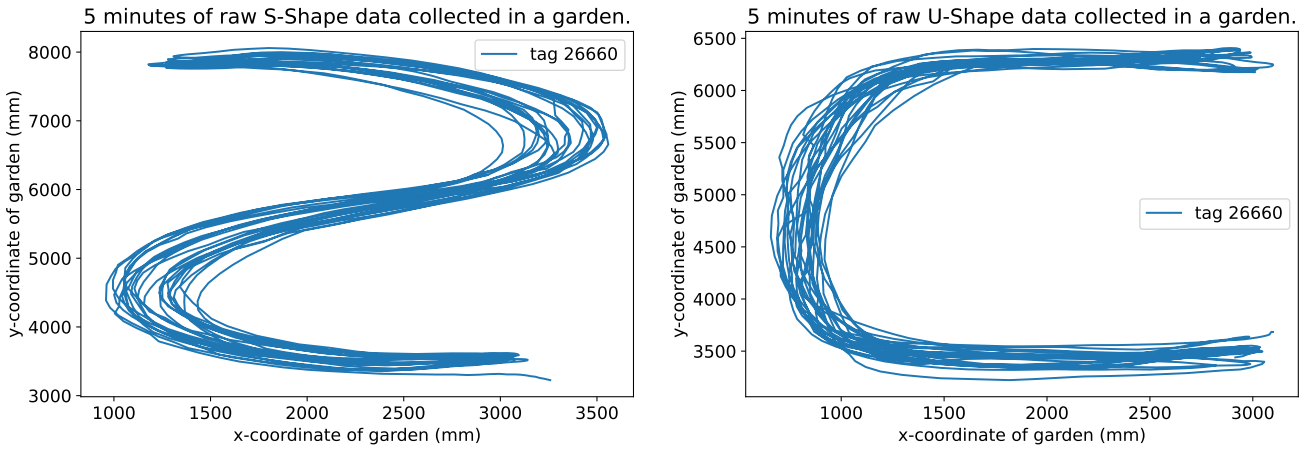
[4] https://www.tszheichoi.com/sensorlogger

exclusively logs the positional data from the tablet (i.e., (*lon*, *lat*)) based on the GNSS chip. In more detail, the Sensor Logger app recorded data from the GNSS chip with a sample rate of 1 Hz. The positioning data (*lat*, *lon*) including the corresponding timestamp from the tablet was automatically logged to a csv format file.
As for the Pozyx Creator Kit [29], each session included four anchors powered via power banks, a master tag connected to the laptop and one active puppet also connected to a power bank. We decided to follow the advice of Pozyx and kept the default UWB settings which yield UWB channel 5, a data bitrate of 110 kbps, a pulse repetition frequency of 64 MHz a preamble length of 1024 and a Tx gain of 11.5 dB [29]. However, we did change the following positional settings [29]: First, we opted for a precision ranging protocol instead of a fast ranging protocol. The precision ranging protocol assures the importance of accurate positioning of the active tags at the expense of the update rate. Second, we could choose between none, low pass, moving average and moving median to optimise the position estimates of the active tags. Regarding this filter, we went for the moving average since it provides smoother trajectories [29]. Last, we could set the filter strength ranging from 0, less delay, but more jittery position estimates, to 15, less jittery position estimates, but more delay [29]. This option was set to 7 in order to maintain a balance between the maximum possible update rate in combination with fewer fluctuations in terms of position estimates. Throughout the six data collection sessions, the Pozyx Creator Kit [29] yielded an average sampling rate of 5.91 Hz per active tag in combination with the aforementioned settings. To log the data from the active puppet tag, we ran a python script in the background. The script contains the paho-mqtt [5] module which establishes a communication with the Pozyx Creator Controller to get the information packages containing the position (x, y) with the corresponding timestamp.

5.1.4 Movement Patterns

As explained in Section 3.2, Yao et al. [42] used the following set of basic movement patterns: { *Straight*, *Circling*, *Bending*}. In our experiment, we adopted the *Straight* and *Circling* patterns and added two other movement patterns to this set: *S-Shape* and *U-Shape*, represented in Figure 5:

[5] https://pypi.org/project/paho-mqtt/



(a) *S-Shape*.

(b) *U-Shape*.

Figure 5: The left figure denotes 5 minutes of *S-Shape* and the right figure represents 5 minutes of U-Shape. Both discrete trajectories have been collected through UWB technology.

Thus, the entire set of movement patterns consists of: {*Straight*, *Circling*, *S-Shape*, *U-Shape*}. To create a baseline indication for these movement patterns, we placed several plastic boxes within the rectangular areas. For example, by placing two boxes in front of each other with some distance between them, we could walk a straight line.

5.1.5 Collecting Movement Patterns

Until this point, we discussed the collected set of movement patterns at two different locations, the approach regarding the positioning of the anchors and the configurations of both positioning systems. In the remaining part of this section, we will concentrate on the execution of collecting movement pattern data under controlled conditions.
During a session, we proceeded as follows: As soon as the script on the laptop, as well as the logging app, were started, one person would commence forming one of the four patterns continuously for five minutes. This person would hold the tablet and the active puppet tag including the power bank. Both were held at the same orientation throughout all the sessions. After 5 minutes, the script would automatically stop logging data from the active tags (i.e., master tag and puppet tag). Regarding the app on the tablet, a timer is displayed which keeps track of the logging time in seconds. Hence, after 300 seconds, the person that walked manually stopped the app from logging data from the tablet. We repeated a single pattern multiple times before proceeding to the next pattern.
Following this controlled protocol throughout six data collection sessions, we collected trajectory data from four individuals. The walked patterns per participant are demonstrated in Table 1:

Table 1: Distribution of the collected pre-defined movement patterns per participant. Each individual discrete trajectory per movement pattern contains 5 minutes of recorded data.

Participant	Number of Sessions	*Straight*	*Circling*	*S-Shape*	*U-Shape*	Total
1	3	7	12	18	17	54
2	1	2	3	2	3	10
3	1	5	5	5	5	20
4	1	5	5	5	5	20

Thus, the movement patterns of the four participants form two data sets with each 104 5-minute-long discrete trajectories consisting of 19 *Straight* patterns, 25 *Circling* patterns, 30 *S-Shape* patterns and 30 *U-Shape* patterns.

5.1.6 Data Sets

As described in the previous section, we obtained two data sets, one with a total of 104 5-minute-long discrete GNSS trajectories and another one with 104 5-minute-long discrete UWB trajectories both consisting of four different movement patterns. However, walking continuously a certain movement pattern for 5 minutes is implausible in a real-life setting. Therefore, we decided to include a hyperparameter in the data preparation stage that accounts for partitioning each 5-minute discrete trajectory into segmented trajectories. For instance, dividing each 5-minute-long discrete trajectory into segmented trajectories of approximately 1 minute gives 95 *Straight* patterns, 125 *Circling* patterns, 150 *S-Shape* patterns and 150 *U-Shape* patterns per positioning technology. A potential consequence of this partitioning protocol is that it might create multiple similar-looking segmented trajectories. In this respect, a drawback might be to obtain near similar instances which could potentially cause the classification model to overfit on the training data set.

5.2 Performance Metrics

Since the performance of a trained model on the test data set serves as an indication of how acceptable a certain pipeline configuration is, it is important to assess several performance metrics against each other. In the case of a binary classification task, we can efficiently describe the model's performance on the test data set in the form of a confusion matrix [36]. Within a confusion matrix, we consider the following four groups: the correctly classified positives, True Positives (TP), the correctly classified negatives, True Negatives (TN), the positives classified as negative, False Negatives (FN), and the negatives classified as positive, False Positives (FP). These four groups allow for explicit performance metrics such as the accuracy, recall, sensitivity, and the Area Under the ROC curve [49]. However, not all metrics are suitable for a multi-class classification task. Therefore, we will consider three metrics capable of evaluating the model's performance in a multi-class setting in the next three subsections.

5.2.1 Accuracy Score

The accuracy score reflects the test instances that are correctly classified by a model divided by the total amount of test instances [36]. The accuracy score can be calculated via the following

equation [36]:

$$acc = \frac{1}{|Te|} \sum_{x \in Te} I[\hat{c}(x) = c(x)] \tag{10}$$

in which Te represents the number of test instances, x is a test instance within Te, c denotes a mapping between instance x and the target value, $\hat{c}$ is a mapping between instance x and the predicted value. Furthermore, I represents an indicator function which evaluates to 1 if the comparison evaluates to true and 0 otherwise [36]. In this respect, the accuracy score lies in the interval [0,1].
This method can also be extended for multiclass classification by taking the sum over all groups represented in Equation (11), adapted from Flach [36]:

$$acc = \frac{1}{|Te|} \sum_{k=1}^{K} \sum_{x:g(x)=k} I[\hat{c}(x) = c(x)] \tag{11}$$

where k represents a class, and $g(x)$ is a mapping from the instance to the associated class. However, the accuracy score can be a biased measure of model performance due to, for instance, class imbalance [49, 50]. Namely, the accuracy score might provide a misleading performance of the model if we are interested in the minority class of an unequally distributed data set. Furthermore, in a multi-class setting, the accuracy metric is incapable of incorporating the number of misclassifications between classes [49].

5.2.2 F_1 Score

The F_1 or F-measure score represents the harmonic mean of precision and recall [36]. Taking a binary classification task as an example, the precision can be calculated according to Equation (12) and the recall can be computed via Equation (13):

$$precision = \frac{TP}{(TP + FP)} \tag{12}$$

$$recall = \frac{TP}{(TP + FN)} \tag{13}$$

Subsequently, we can express the F_1 score according to Equation (14), adapted from Flach [36]:

$$F_1 = 2 \cdot \frac{(precision \cdot recall)}{(precision + recall)} \tag{14}$$

From this equation, it becomes apparent that the F_1 score is dependent on both the precision and recall. This implies that both the precision and recall should be high if we want to achieve a high F_1 score. The F_1 metric provides a score in the interval [0,1]. Furthermore, the F_1 score is not affected by the TNs since these are not considered in both precision and recall [36]. Thus, for tasks in which we are indifferent to TNs, we can use the F_1 score. Thereby, the inconsideration of the TNs raises another aspect of the F_1 score. Namely, class swapping can result in different outcomes [50].
By extending the F_1 metric to a multi-class setting, we can obtain an F_1 score via, for instance, micro- or macro-averaging [51]. Micro-averaging computes the precision and recall for each class and accumulates them by taking into account the original proportions per class [51]. Subsequently,

it computes the global F_1 score via these averaged precision and recall scores. On the other hand, macro-averaging considers the average of precision and recall on a per-class basis and, thus, computes the F_1 score per class [51]. To obtain the global F_1 score it averages over the per-class F_1 scores. Hence, the macro F_1 score is robust against class imbalance [51].

5.2.3 Matthew's Correlation Coefficient

Matthew's Correlation Coefficient (MCC) was formally introduced by Matthew [52]. From a statistical point of view, the MCC expresses the linear relationship between two binary variables via discretisation of the Pearson's correlation coefficient [49]. In assessing a machine learning model within a binary classification setting, the MCC denotes a correlation coefficient between the predicted and observed classifications computed via the following equation [50]:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}} \tag{15}$$

From this equation, we can infer that the MCC score reflects the entire confusion matrix in a single score. Hence, a good score can be achieved via a classifier that is capable of correctly classifying positive as well as negative instances [50]. Regarding the interpretation of the MCC score, -1 denotes a perfect misclassifier, 0 means a no-skill classifier and 1 means a perfect classifier. Furthermore, in comparison to the F_1 score, MCC is not affected by class swapping [50]. Thereby, Chicco & Giuseppe [50] demonstrate that MCC reflects a more reliable score regarding the classification capabilities of a trained model compared to the F_1 score. As explained in Section 5.2.2, the F_1 metric, being composed of the precision and recall, ignores the number of TNs. In this respect, the F_1 cannot account for the relationship between positive and negative instances in terms of their respective proportions. Therefore, the F_1 risks to represent an unrealistic performance of the model in case of an imbalanced data set [50]. On the contrary, a model classifying the majority of the positive and negative test instances correctly regardless of their respective proportion in the test set will result in a relatively high MCC score [50]. Thus, Chicco & Giuseppe [50] advise utilising the MCC instead of the F_1 or accuracy scores in binary classification tasks.
Gorodkin [53] extended this metric to multi-class classification tasks. The MCC score for a multi-class setting can be computed according to Equation (16), adapted from [37]:

$$MCC = \frac{e \cdot Te - \sum_k^K p_k \cdot v_k}{\sqrt{(Te^2 - \sum_k^K {p_k}^2) \cdot (Te^2 - \sum_k^K {v_k}^2)}} \tag{16}$$

in which e represents the total number of correctly predicted instances, v_k denotes the true occurrences of class k and p_k describes the number of times class k was predicted [37]. According to Grandini et al. [54], the formula explicitly reflects the importance of correctly classifying the instances of each class in order to obtain a high MCC score.

5.3 Experiment Settings

As previously discussed in Chapter 4, the data preparation, noise removal and classifier parts contain hyperparameters. In the data preparation phase, we implemented a hyperparameter that decides upon the partitioning size of the original 5-minute-long discrete trajectories. For instance, a

value of 5 divides each 5-minute-long discrete trajectory into 5 segmented trajectories of each one minute. If the value is equal to 1, the original discrete trajectory will be considered in the next stages of the automated framework. The following table describes the partitioning hyperparameter present in the data preparation stage of each pipeline configuration:

Table 2: Hyperparameter responsible for partitioning the original instances in the preparation step of the pipeline.

Method	**Hyperparameter**	**Range**	**Type**
partitioning instances	split	[1, 10]	Uniform Integer

An optional part of our pipeline is to perform noise removal via the Savitzky Golay Filter on the raw location data or on the computed Two-point Features. As discussed in Section 4.1.4, the Savitzky Golay Filter depends on an odd window size and a polynomial order. Table 3 demonstrates these two hyperparameters involved in the Savitzky Golay Filter:

Table 3: Hyperparameters of the Savitzky Golay Filter in the noise removal step of the pipeline.

Method	**Hyperparameter**	**Range**	**Type**
Savitzky Golay Filter	window_length	[1, 3, ..., 27, 29]	Categorical
	polyorder	[1, 10]	Uniform Integer

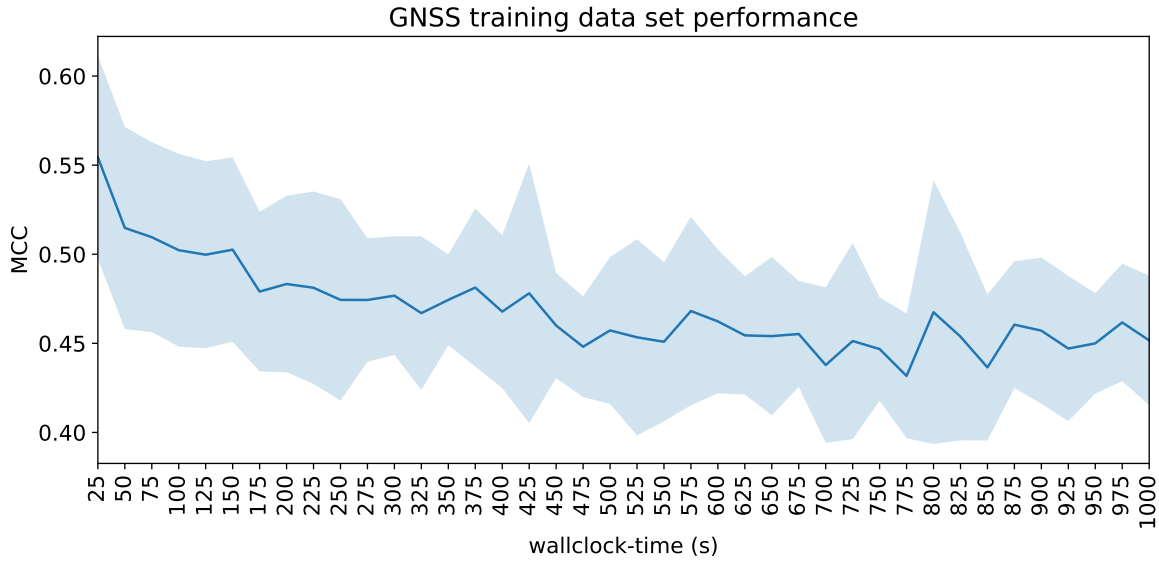
Given that the respective window size of the Savitzky Golay Filter needs to be odd, we opted for a Categorical type of hyperparameter since a Uniform Integer would cause errors every time an even value is introduced as window size. The final step of each pipeline configuration consists of applying a classification model. We considered three classification algorithms: Decision Tree, Random Forest and Support Vector Machine. The hyperparameters of each respective model are described in Table 4:

Table 4: Three classification models with their corresponding hyperparameter ranges.

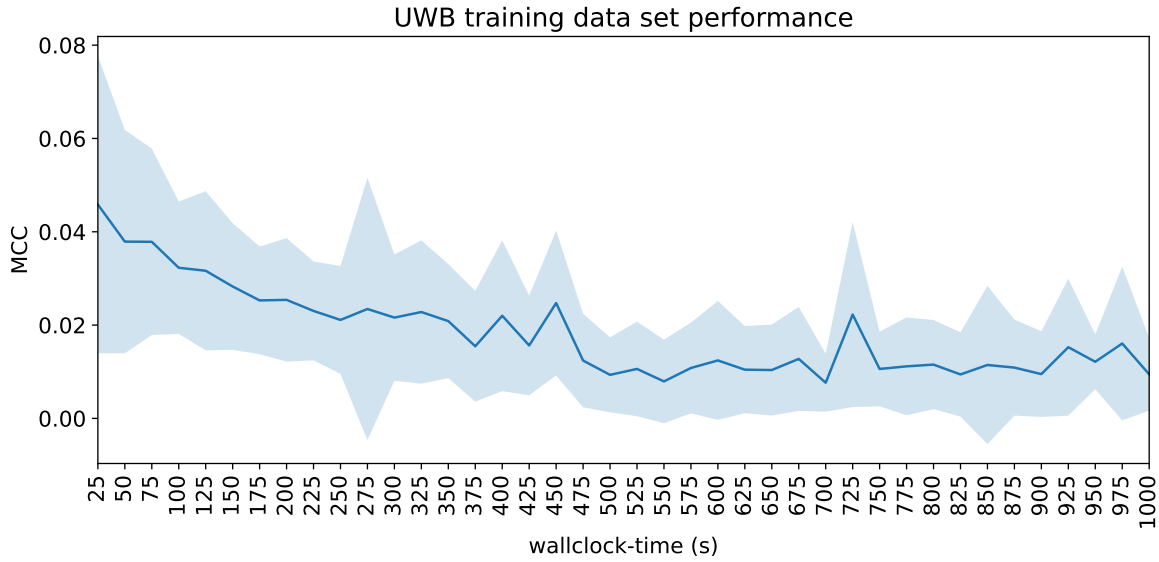
Model	**Hyperparameter**	**Range**	**type**
Decision Tree	max_depth	[5, 50]	Uniform Integer
	min_samples_leaf	[1, 10]	Uniform Integer
	min_samples_split	[2, 10]	Uniform Integer
	criterion	["gini", "entropy"]	Categorical
Random Forest	n_estimators	[5, 100]	Uniform Integer
	min_samples_leaf	[1, 10]	Uniform Integer
	max_depth	[5, 50]	Uniform Integer
	min_samples_split	[2, 10]	Uniform Integer
	criterion	["gini", "entropy"]	Categorical
Support Vector Machine	C	[0.1, 100]	Uniform Float
	kernel	["linear", "poly", "rbf", "sigmoid"]	Categorical

Different hyperparameter settings can have different effects on the performance of the pipeline. Therefore, we decided to automate the HPO process via SMAC [23]. Although SMAC automates the HPO, it remains a time-consuming task [22]. Thereby, setting no seed in the SMAC object makes the optimization non-deterministic which can result in different incumbent outcomes (i.e., sets of hyperparameter values) for different runs of the same pipeline configuration [23]. For these reasons, we decided to first assess the relationship between the wallclock-time and the MCC score of a model. Second, we implemented an adaptation of the bootstrapping protocol proposed in the work of Wang et al. [55].

The wallclock-time denotes the time between starting the SMAC optimization and terminating it in seconds. In this respect, we could analyze how much wallclock-time SMAC needs to find near-optimal hyperparameter settings for a specific pipeline configuration. To perform this, we first split the entire data set into a training and test set (i.e., training:testing, 67:33). Then, we performed 15 independent optimization runs for the pipeline configuration yielding the most hyperparameters (i.e., noise removal in combination with an RF model). We decided to increment the wallclock-time with steps of 25 seconds. The averaged MCC scores on the training data set for different wallclock-time values for both the GNSS and UWB pipelines are represented in Figures 6a and 6b respectively:



(a) GNSS



(b) UWB

Figure 6: A visualization of the relationship between the wallclock-time and MCC score of a pipeline configuration with noise removal application on the raw location data and a RF model as classifier. We independently sampled 15 training performances per wallclock-time from SMAC and averaged them for both the GNSS and UWB setting. The dark blue line represent the mean and the light blue area reflects the standard deviation.

Based on Figures 6a and 6b, we decided to implement a wallclock-time of 500 seconds in the bootstrapping protocol.
The adapted bootstrapping protocol [55] works as follows: First, we split the entire data set into a training and test set (i.e., training:testing, 67:33). Second, we utilised SMAC to optimise a single

pipeline configuration on the training data set via stratified 10-fold cross-validation. We opted for stratified k-fold cross-validation because it partitions the training data set in folds, each containing approximately the same distribution of class labels as in the training data set [37]. Given that SMAC targets HPO as a minimization problem, we aggregated over the 10 resulting performances and subtracted the outcome from 1 as an indicator of the respective pipeline configuration performance. Furthermore, we utilised Matthew's Correlation Coefficient [52] to measure the performance of a classification model during the HPO. We chose Matthew's Correlation Coefficient since a high-performance score is correlated with the number of correctly predicted instances [54] and it is robust against potential performance bias from an imbalanced data set [50, 54, 37]. In this fashion, we let SMAC optimise a single pipeline setting 15 times on the training data set. Then, 5 incumbents were randomly sampled from the 15 generated ones. Of these 5 incumbents, the one yielding the lowest optimization value served as the final hyperparameter configuration. Next, we repeated this process 50 times for a single pipeline configuration. Subsequently, we retrained 50 models according to the 50 produced incumbents on the training data set. Finally, we evaluated the 50 models on the test set and averaged over the 50 scores.

5.4 Results

Implementation-wise, we conducted our experiments in Python 3.9.7[6] together with version 1.0.2 of scikit-learn[7], version 1.7.1 of SciPy[8] and version 1.3.3 of SMAC[9]. Via SMAC, we performed automated HPO on each pipeline configuration to compare the performances against each other. Thereby, the bootstrapping method ensures that we conduct a fair comparison between the different pipeline configurations (see Figure 4.1). In the next section, we will first describe the results for the GNSS technology and the UWB separately. Then, we will compare the two technologies according to the best pipeline configuration found via SMAC.

5.4.1 GNSS Results

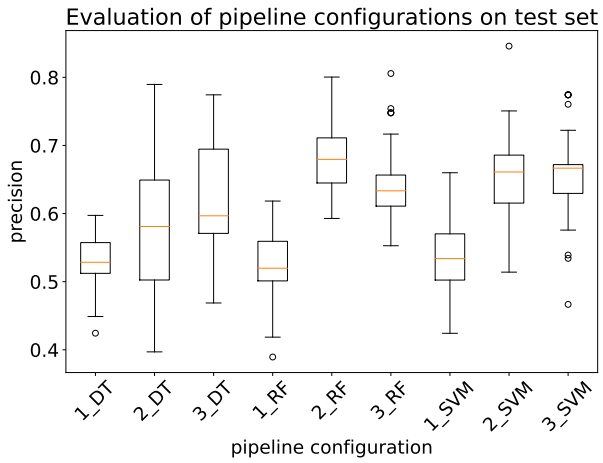
Given our different pipeline settings, we are interested in three variations regarding the noise removal part (i.e., no noise removal, noise removal on raw location data and noise removal on Two-point Features) as well as three different options in terms of classification algorithms (i.e., Decision Tree, Random Forest and Support Vector Machine) (see Figure 4.1). This yields a total of 9 pipeline configurations for the GNSS technology. Each pipeline configuration was assessed via the above-mentioned bootstrapping protocol in Section 5.3. The averaged scores on the test data set reflected in macro-precision, macro-recall, macro-F_1 and MCC which are based on the 50 best-found hyperparameter settings are demonstrated in Table 5:

[6] https://www.python.org
[7] https://scikit-learn.org
[8] https://docs.scipy.org
[9] https://github.com/automl/SMAC3

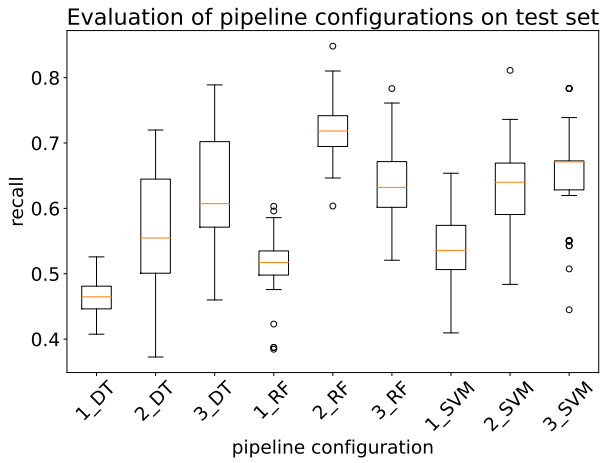
Table 5: GNSS results on the test data set for each pipeline configuration. The precision, recall and F_1 scores are calculated on a macro basis.

		Metric			
Model	**Noise Removal**	precision	recall	F_1	MCC
DT	no noise removal	52.75±0.036	46.23±0.027	47.90±0.028	25.36±0.040
	noise removal on raw location data	59.33±0.099	57.40±0.082	56.46±0.090	41.93±0.114
	noise removal on Two-point Features	63.01±0.086	63.30±0.094	62.23±0.093	48.78±0.122
RF	no noise removal	52.72±0.050	51.50±0.046	51.50±0.047	33.17±0.061
	noise removal on raw location data	**68.19±0.048***	**72.06±0.044***	**69.02±0.047***	**57.05±0.061**
	noise removal on Two-point Features	64.35±0.051	64.37±0.58	63.62±0.054	50.19±0.075
SVM	no noise removal	53.48±0.050	53.12±0.052	53.03±0.051	34.58±0.064
	noise removal on raw location data	65.34±0.061	62.86±0.060	63.23±0.060	48.92±0.082
	noise removal on Two-point Features	65.77±0.060	65.60±0.070	65.16±0.064	52.63±0.090

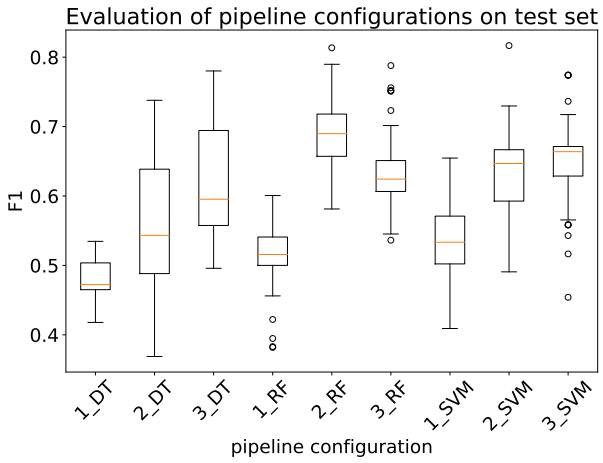
To determine if a certain pipeline configuration gives better performance than another, we performed a statistical significance test proposed in the work of Wang et al. [56]. First, we checked if the underlying distribution of each set of 50 performance scores was normally distributed. By performing an Anderson-Darling test [57], we concluded that the sets were not sampled from a Gaussian distribution. Given that the distributions are extracted from the same population, we decided to utilise the non-parametric Wilcoxon signed-rank test [58]. Regarding the significance level, we used the standard significance level of 0.05. Hence, in Table 5, the performance score with an additional asterisk indicates that this particular pipeline configuration gives statistically better results than the other configurations within the same performance metric. Thereby, bold-faced scores represent the highest performance per score metric.
Based on Table 5, the pipeline configuration yielding noise removal on the raw location data in combination with an RF model gives statistically higher results for the precision, recall and F_1 scores than the other possible pipeline combinations. As for the MCC score, the highest result was obtained via performing noise removal on the raw location data with an RF model. However, this result did not prove statistically significant. A visualization of the distributions of the 50 scores per pipeline configuration is represented in Figure 7:



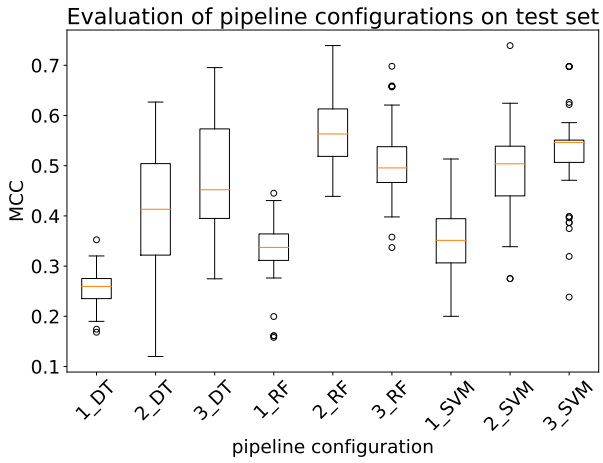
(a) Precision score distributions



(b) Recall score distributions



(c) F_1 score distributions



(d) MCC score distributions

Figure 7: A visualization of the distribution of scores per pipeline configuration for the GNSS data set divided over the four score metrics. The precision, recall and F_1 scores are computed on a macro basis. On the x-axis, 1 indicates no noise removal application, 2 refers to noise removal on the raw location data and 3 is noise removal on the Two-point Features.

As becomes clear from Figure 7, the highest performance over all four considered score metrics was achieved with a pipeline configuration yielding noise removal on the raw location data in combination with an RF model. Furthermore, the advantage of applying the Savitzky Golay Filter on the raw location data or the Two-point Features in terms of acquiring a higher performance score is reflected in Figure 7. For all four score metrics, the lowest scores were obtained with pipeline configurations yielding no noise removal application.

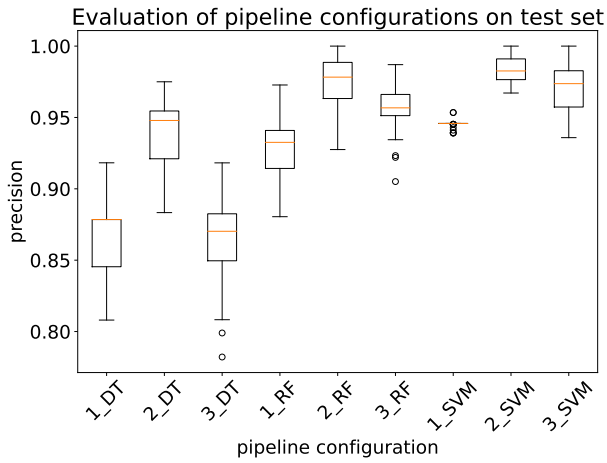
5.4.2 UWB SMAC optimiser Results

Analogously to the GNSS experiments, we assessed 9 pipeline configurations on the collected UWB data. The Anderson-Darling test [57] confirmed that the sets per pipeline configuration were not sampled from a Gaussian Distribution. The averaged test data set results over the 50 independent retrained models are reported in Table 6 in which bold-faced scores again indicate the highest performance and additional asterisks refer to the pipeline configuration that performs statistically better than other pipeline configurations per metric:

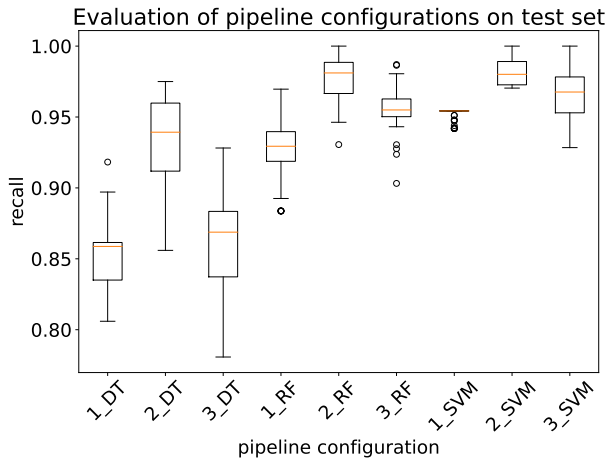
Table 6: UWB results on the test data set for each pipeline configuration. The precision, recall and F_1 scores are calculated on a macro basis.

		Metric			
Model	**N.R. Application**	precision	recall	F_1	MCC
DT	no noise removal	86.78±0.024	85.72±0.022	85.72±0.024	80.77±0.028
	noise removal on raw location data	93.70±0.025	93.25±0.030	93.22±0.029	91.05±0.037
	noise removal on Two-point Features	86.51±0.031	86.22±0.034	86.12±0.033	81.41±0.044
RF	no noise removal	92.61±0.020	92.47±0.020	92.34±0.021	89.70±0.025
	noise removal on raw location data	97.51±0.016	97.73±0.015	97.57±0.016	96.71±0.022
	noise removal on Two-point Features	95.76±0.015	95.60±0.015	95.63±0.015	94.15±0.020
SVM	no noise removal	94.55±0.002	95.21±0.004	94.74±0.003	93.10±0.004
	noise removal on raw location data	**98.21±0.008**	**98.04±0.008**	**98.09±0.008**	**97.59±0.009**
	noise removal on Two-point Features	96.88±0.021	96.57±0.022	96.67±0.021	95.73±0.027

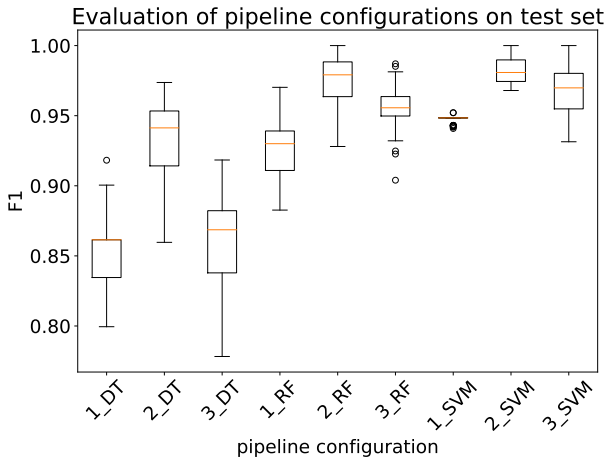
Based on Table 6, we can conclude that the highest performances for each score metric are achieved by applying noise removal on the raw location data in combination with an SVM model. However, this particular pipeline configuration is not statistically the one with the best performance. The lack of a statistically significant result is probably due to the fact that the pipeline configuration consisting of noise removal on the raw location data in combination with an RF model gives very similar results for all four score metrics. Figure 8 visualises the distribution of each pipeline configuration partitioned according to the four used performance metrics:



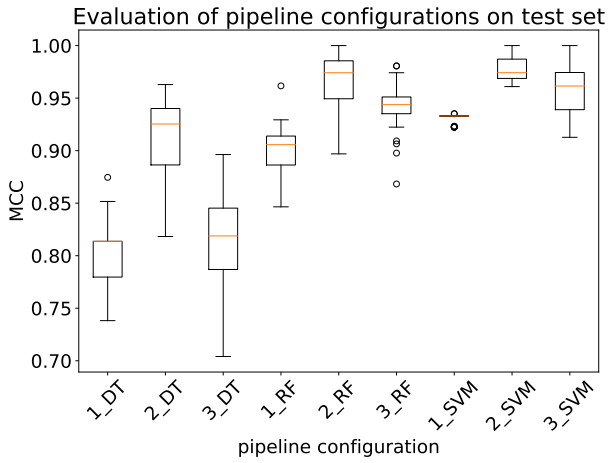
(a) Precision score distributions



(b) Recall score distributions



(c) F_1 score distributions



(d) MCC score distributions

Figure 8: A visualization of the distribution of scores per pipeline configuration for the UWB data set divided over the four score metrics. The precision, recall and F_1 scores are computed on a macro basis. On the x-axis, 1 indicates no noise removal application, 2 refers to noise removal on the raw location data and 3 is noise removal on the Two-point Features.

Figure 6 shows that, for all four metrics, the pipeline configuration with noise removal on the raw location data in combination with an RF model and the pipeline configuration with noise removal on the raw location data in combination with an SVM model closely related to each other in terms of the performance score. Furthermore, the optional noise removal application proves, in almost all cases, beneficial for the final performance. An exception is present for the pipeline configuration yielding noise removal on the Two-point Features in combination with an DT model assessed via precision.

5.4.3 GNSS Versus UWB Comparison

From Table 5, we concluded that the pipeline configuration containing noise removal on the raw location data in combination with an RF model gave the highest performance results for all four performance metrics. Thereby, this specific pipeline proved to achieve statistically higher results than other configurations for the precision, recall and F_1 score. Regarding the UWB results in Table 6, the pipeline configuration yielding noise removal on the raw location data with an SVM model achieved the highest results across the four performance metrics. However, in comparison to other pipeline configurations, this particular pipeline did not achieve statistical higher performance than other pipelines.

Therefore, we decided to compare the GNSS versus the UWB technology on the pipelines that achieved the highest results according to the significance test approach in the work of Wang et al. [56]. As already discussed in Sections 5.4.1 and 5.4.2, the sampled results per pipeline configuration are not normally distributed. Additionally, the sets that we want to compare are not sampled from the same population. For these reasons, we opted for the non-parametric Mann-Whitney U test [59] with a standard confidence level of 95%.

Regarding the hypothesis test on comparing the pipeline yielding noise removal on the raw location data in combination with an RF model for the GNSS and the pipeline containing noise removal on the raw location data with an SVM model for the UWB, we observe a positive effect for the UWB with a significance level of 99% for the precision. As for the recall, an identical positive effect for the UWB can be observed yielding a significance level of 99%. Regarding the, F_1 score, a positive effect can be observed in favor of the UWB with a significance level of 99%. Likewise for the MCC metric, a positive effect with a significance level of 99% is present for the UWB. Thus, the UWB pipeline statistically outperforms the GNSS pipeline across all four score metrics.

6 Conclusion and Further Research

In this final chapter, we will first give a brief overview of our study including the conclusions. Next, we will provide possible future directions as well as additional information about the data collection sessions.

6.1 Conclusion

In this study, we designed and evaluated an automated framework which is capable of correctly classifying pre-defined human movement patterns collected with GNSS and UWB technology. The automated framework can handle raw spatio-temporal data from both the GNSS and UWB systems, pre-processes this raw data and classifies them according to the correct patterns. The automated framework allows for three options in terms of the noise removal stage: no noise removal, Savitzky Golay Filter on the raw location data or Savitzky Golay Filter on the computed Two-point Features as well as three options for classification algorithms: Decision Tree, Random Forest and Support Vector Machine. Under controlled conditions, we collected a total of 104 5-minute-long discrete GNSS and a total of 104 5-minute-long discrete UWB trajectories consisting of four movement patterns. Based on these collected discrete trajectories, we tested the 9 different pipeline configurations of our automated framework via a bootstrapping protocol to assess which one yielded the best performance. For the GNSS technology, we achieved the highest performance with a pipeline configuration consisting of noise removal applied on the raw location data with a Random Forest model. Regarding the UWB technology, the pipeline configuration with noise removal applied on the raw location data in combination with a Support Vector Machine model gave the highest performance. We compared these two pipeline configurations through hypothesis testing and observed, based on a significance level of 99%, that UWB achieves significantly better performance than GPS at classifying pre-defined movement patterns in areas of approximately 100m^2.

6.2 Future directions

In our thesis, we collected data of four basic pre-defined movement patterns: *Straight*, *Circling*, *S-Shape* and *U-Shape*. In a future research project, it would be interesting to include more complex movement patterns or to create combinations of the four above-mentioned movement patterns as is done in the work of Yao et al. [42]. Adding more complex movement patterns which are even more interesting and challenging could make the pipeline configuration more adaptable to the real-life complex movement patterns of human beings.

Furthermore, we collected movement patterns under controlled conditions which allowed us to correctly label each trajectory. However, a movement pattern data set might not always yield labels or one might simply not have access to the ground truth to correctly label the trajectories. To mitigate this issue, it might be useful to integrate clustering algorithms [42] instead of classification algorithms in the final step of our proposed framework.

Feature engineering plays a crucial role in our framework. However, one might wonder if feature engineering is necessary to correctly classify pre-defined movement patterns. Unfortunately, this direction demands more raw data than we collected in this study. Nevertheless, it would be interesting to investigate this point in a future study.

Regarding the Two-point Features we computed per trajectory, one might remark that especially the average speed and change of speeds vary among individuals. In other words, an adult can walk a pattern with a different speed than a child and hence the average speed and change of speeds of a trajectory become representative of a certain subgroup. Ultimately, we would like to achieve a pipeline that is capable of correctly classifying the movement patterns independent of a person's speed. To circumvent the fluctuation in terms of the sampling rate resulting from different speeds, we could normalise the trajectories time-wise.

6.3 System failures and weather conditions

Before collecting the set of pre-defined movement patterns, we conducted four orientation sessions with the Pozyx Creator Kit [29] in order to become acquainted with the system and its functionalities. As described in Section 2.2.2, the Pozyx Creator Kit contains five anchors. During the second orientation session, one of the five anchors failed for an unknown reason. Therefore, we decided to exclude this specific anchor during the data collection sessions. Hence, we only utilised anchors 0X6842, 0X685F, 0X686C and 0X6865 as reference points for the localisation of the active puppet tag.
Furthermore, we informally investigated the capacities of the Pozyx Creator Kit [29] in terms of its positioning range. According to Pozyx [29], the Creator Kit offers a typical range of 30 m. However, we observed that the system can handle a range of around 10m in an outdoor setup instead of a range of 30m. This observation emphasises the need for a sufficient amount of anchors if one desires to cover a larger area than approximately 100m^2. Hence, a drawback of the UWB system is that it requires a significant number of anchors to cover larger areas than 100m^2. Thereby, the costs for such a setup accumulate with every extra necessary anchor.
Aside from the four orientation sessions, we performed a total of eight data collection sessions. During the setup of two sessions, it started raining. Given that the Pozyx Creator Kit [29], the tablet and the laptop are not IP-57 certified, we had to abruptly cancel the data collection. Furthermore, at the beginning of one session, the battery of the tablet turned out to be empty. Nevertheless, we collected movement patterns via the Pozyx system during this session. To maintain a fair comparison between the two positioning technologies, we eventually decided to exclude this session from the data set. Additionally, we excluded a single straight and circling line from a session since the tablet did not record the positional data.

References

[1] J. Bian, D. Tian, Y. Tang, and D. Tao, "A survey on trajectory clustering analysis," *arXiv preprint arXiv:1802.06971*, 2018.

[2] Y. Zheng, "Trajectory data mining: an overview," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 6, no. 3, pp. 1–41, 2015.

[3] G. Yuan, P. Sun, J. Zhao, D. Li, and C. Wang, "A review of moving object trajectory clustering algorithms," *Artificial Intelligence Review*, vol. 47, no. 1, pp. 123–144, 2017.

[4] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma, "Understanding mobility based on gps data," in *Proceedings of the 10th international conference on Ubiquitous computing*, pp. 312–321, 2008.

[5] Q. Gao, F. Zhang, F. Yao, A. Li, L. Mei, and F. Zhou, "Adversarial mobility learning for human trajectory classification," *IEEE Access*, vol. 8, pp. 20563–20576, 2020.

[6] A.-C. Petre, C. Chilipirea, M. Baratchi, C. Dobre, and M. van Steen, "Chapter 14 - wifi tracking of pedestrian behavior," in *Smart Sensors Networks* (F. Xhafa, F.-Y. Leu, and L.-L. Hung, eds.), Intelligent Data-Centric Systems, pp. 309–337, Academic Press, 2017.

[7] F. Zhou, R. Yin, K. Zhang, G. Trajcevski, T. Zhong, and J. Wu, "Adversarial point-of-interest recommendation," in *The world wide web conference*, pp. 3462–34618, 2019.

[8] M. Baratchi, N. Meratnia, and P. J. Havinga, "Recognition of periodic behavioral patterns from streaming mobility data," in *International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*, pp. 102–115, Springer, 2013.

[9] D.-H. Shih, M.-H. Shih, D. C. Yen, and J.-H. Hsu, "Personal mobility pattern mining and anomaly detection in the gps era," *Cartography and Geographic Information Science*, vol. 43, no. 1, pp. 55–67, 2016.

[10] M. Etemad, A. Soares Júnior, and S. Matwin, "Predicting transportation modes of gps trajectories using feature engineering and noise removal," in *Canadian conference on artificial intelligence*, pp. 259–264, Springer, 2018.

[11] S. Dabiri and K. Heaslip, "Inferring transportation modes from gps trajectories using a convolutional neural network," *Transportation research part C: emerging technologies*, vol. 86, pp. 360–371, 2018.

[12] M. Baratchi, N. Meratnia, and P. Havinga, "On the use of mobility data for discovery and description of social ties," in *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*, pp. 1229–1236, 2013.

[13] B. M. Heravi, J. L. Gibson, S. Hailes, and D. Skuse, "Playground social interaction analysis using bespoke wearable sensors for tracking and motion capture," in *Proceedings of the 5th International Conference on Movement and Computing*, pp. 1–8, 2018.

[14] Y. Luo, D. Irvin, N. Buss, A. Gutierrez, and B. Rous, "Outdoor playground localization system for tracking young children using ubisense sensor network," in *2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pp. 7–12, IEEE, 2020.

[15] P. Dabove and V. Di Pietra, "Towards high accuracy gnss real-time positioning with smartphones," *Advances in Space Research*, vol. 63, no. 1, pp. 94–102, 2019.

[16] J. Tomaštík Jr, J. Tomaštík Sr, Š. Saloň, and R. Piroh, "Horizontal accuracy and applicability of smartphone gnss positioning in forests," *Forestry: An International Journal of Forest Research*, vol. 90, no. 2, pp. 187–198, 2017.

[17] M. R. Mahfouz, C. Zhang, B. C. Merkl, M. J. Kuhn, and A. E. Fathy, "Investigation of high-accuracy indoor 3-d positioning using uwb technology," *IEEE Transactions on Microwave Theory and Techniques*, vol. 56, no. 6, pp. 1316–1330, 2008.

[18] M. Kuhn, C. Zhang, B. Merkl, D. Yang, Y. Wang, M. Mahfouz, and A. Fathy, "High accuracy uwb localization in dense indoor environments," in *2008 IEEE International Conference on Ultra-Wideband*, vol. 2, pp. 129–132, IEEE, 2008.

[19] K.-M. Mimoune, I. Ahriz, and J. Guillory, "Evaluation and improvement of localization algorithms based on uwb pozyx system," in *2019 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pp. 1–5, IEEE, 2019.

[20] A. Waqar, I. Ahmad, D. Habibi, and Q. V. Phung, "Analysis of gps and uwb positioning system for athlete tracking," *Measurement: Sensors*, vol. 14, p. 100036, 2021.

[21] A. Bastida Castillo, C. D. Gómez Carmona, E. De la Cruz Sánchez, and J. Pino Ortega, "Accuracy, intra-and inter-unit reliability, and comparison between gps and uwb-based position-tracking systems used for time–motion analyses in soccer," *European journal of sport science*, vol. 18, no. 4, pp. 450–457, 2018.

[22] M. Feurer and F. Hutter, "Hyperparameter optimization," pp. 3–38.

[23] M. Lindauer, K. Eggensperger, M. Feurer, A. Biedenkapp, D. Deng, C. Benjamins, T. Ruhkopf, R. Sass, and F. Hutter, "Smac3: A versatile bayesian optimization package for hyperparameter optimization," in *ArXiv: 2109.09831*, 2021.

[24] C. Renso, S. Spaccapietra, and E. Zimányi, *Mobility data*. Cambridge University Press, 2013.

[25] X. Zhang, X. Tao, F. Zhu, X. Shi, and F. Wang, "Quality assessment of gnss observations from an android n smartphone and positioning performance analysis using time-differenced filtering approach," *Gps Solutions*, vol. 22, no. 3, pp. 1–11, 2018.

[26] E. D. Kaplan and C. Hegarty, *Understanding GPS/GNSS: principles and applications*. Artech house, 2017.

[27] Z. Sahinoglu, S. Gezici, and I. Guvenc, "Ultra-wideband positioning systems," *Cambridge, New York*, 2008.

[28] Pozyx, "Pozyx." https://www.pozyx.io, 2022.

[29] Pozyx, "Pozyx creator kit." https://www.pozyx.io/creator, 2022.

[30] M. Ridolfi, S. Van de Velde, H. Steendam, and E. De Poorter, "Analysis of the scalability of uwb indoor localization solutions for high user densities," *Sensors*, vol. 18, no. 6, p. 1875, 2018.

[31] M. Malajner, D. Gleich, and P. Planinsic, "Soil type characterization for moisture estimation using machine learning and uwb-time of flight measurements," *Measurement*, vol. 146, pp. 537–543, 2019.

[32] M. R. Mahfouz, A. E. Fathy, M. J. Kuhn, and Y. Wang, "Recent trends and advances in uwb positioning," in *2009 IEEE MTT-S International Microwave Workshop on Wireless Sensing, Local Positioning, and RFID*, pp. 1–4, IEEE, 2009.

[33] H. Rainer, "Symetric double sided - two way ranging," Contribution 802.15-05-0334-00-004a to the IEEE 802.15.4a Ranging Subcommittee June 2005.

[34] M. Guia, R. R. Silva, and J. Bernardino, "Comparison of naïve bayes, support vector machine, decision trees and random forest on sentiment analysis.," *KDIR*, vol. 1, pp. 525–531, 2019.

[35] G. Bonaccorso, *Machine Learning Algorithms: Popular algorithms for data science and machine learning*. Packt Publishing Ltd, 2018.

[36] P. Flach, *Machine learning: the art and science of algorithms that make sense of data*. Cambridge university press, 2012.

[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[38] M. Pal, "Multiclass approaches for support vector machine based land cover classification," *arXiv preprint arXiv:0802.2411*, 2008.

[39] M. Claesen and B. De Moor, "Hyperparameter search in machine learning," *arXiv preprint arXiv:1502.02127*, 2015.

[40] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization.," *Journal of machine learning research*, vol. 13, no. 2, 2012.

[41] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," *Advances in neural information processing systems*, vol. 28, 2015.

[42] D. Yao, C. Zhang, Z. Zhu, Q. Hu, Z. Wang, J. Huang, and J. Bi, "Learning deep representation for trajectory clustering," *Expert Systems*, vol. 35, no. 2, p. e12252, 2018.

[43] A. Savitzky and M. J. Golay, "Smoothing and differentiation of data by simplified least squares procedures.," *Analytical chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964.

[44] R. W. Schafer, "What is a savitzky-golay filter?[lecture notes]," *IEEE Signal processing magazine*, vol. 28, no. 4, pp. 111–117, 2011.

[45] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.

[46] H. L. Kennedy, "Improving the frequency response of savitzky-golay filters via colored-noise models," *Digital Signal Processing*, vol. 102, p. 102743, 2020.

[47] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[48] Samsung, "Samsung Galaxy Tab A7 Lite." https://www.samsung.com/nl/tablets/galaxy-tab-a/galaxy-tab-a7-lite-lte-gray-32gb-sm-t225nzaaeub/, 2022.

[49] R. Delgado and X.-A. Tibau, "Why cohen's kappa should be avoided as performance measure in classification," *PloS one*, vol. 14, no. 9, p. e0222916, 2019.

[50] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation," *BMC genomics*, vol. 21, no. 1, pp. 1–13, 2020.

[51] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information processing & management*, vol. 45, no. 4, pp. 427–437, 2009.

[52] B. W. Matthews, "Comparison of the predicted and observed secondary structure of t4 phage lysozyme," *Biochimica et Biophysica Acta (BBA)-Protein Structure*, vol. 405, no. 2, pp. 442–451, 1975.

[53] J. Gorodkin, "Comparing two k-category assignments by a k-category correlation coefficient," *Computational biology and chemistry*, vol. 28, no. 5-6, pp. 367–374, 2004.

[54] M. Grandini, E. Bagli, and G. Visani, "Metrics for multi-class classification: an overview," *arXiv preprint arXiv:2008.05756*, 2020.

[55] C. Wang, T. Bäck, H. H. Hoos, M. Baratchi, S. Limmer, and M. Olhofer, "Automated machine learning for short-term electric load forecasting," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 314–321, IEEE, 2019.

[56] C. Wang, M. Baratchi, T. Bäck, H. H. Hoos, S. Limmer, and O. Markus, "Towards time-series-specific feature engineering in automated machine learning frameworks for time-series forecasting," *IEEE*, under review.

[57] T. W. Anderson and D. A. Darling, "Asymptotic theory of certain" goodness of fit" criteria based on stochastic processes," *The annals of mathematical statistics*, pp. 193–212, 1952.

[58] F. Wilcoxon, "Individual comparisons by ranking methods," in *Breakthroughs in statistics*, pp. 196–202, Springer, 1992.

[59] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, pp. 50–60, 1947.