University of Zurich[UZH]

Comparison of Stop-Move Detection Algorithms for GPS Data of Older Adults

GEO 511 Master's Thesis

Author

Elena Ebert

15-700-586

Supervised by

Dr. Eun-Kyeong Kim, Department of Geography, University of Zurich

Faculty representative

Prof. Dr. Robert Weibel, Department of Geography, University of Zurich

30.04.2020

Department of Geography, University of Zurich

Abstract

Due to the estimated aging demographic, promoting healthy aging in everyday life, particularly among older adults, has become an important agenda to alleviate the burden on the healthcare system and further society. It is known that daily mobility is the key to maintain physical and mental health of older adults through diverse activities including exercise, daily traveling, and social activities. Hence, it is crucial to assess the movement behaviors of older adults to draw conclusion about their state of health in daily life. Wearable location sensing technologies using GPS have enabled obtaining and analyzing the daily mobility data. Movement data analysis involves several procedures to transform sensor signals to meaningful information on mobility behaviors. As an early step, stop-move detection algorithms play a significant role in automatically extracting activity locations and moves from raw movement data.

While many algorithms have been proposed for stop-move detection, selecting one among the existing algorithms is still challenging. This is because input parameters and optimal thresholds vary with different algorithms as well as input data and are often difficult to determine without a comparative analysis. Such kind of algorithm comparison research is still an evolving topic in the field of movement analysis. In this context, the aim of this thesis is to review and compare stop-move detection algorithms for GPS data in order to analyse the mobility patterns of older adults from a spatial and temporal perspective. The data was collected from the interdisciplinary research project named *MObility, Activity and Social Interaction Study* (MOASIS), conducted by research teams at the University of Zurich.

To evaluate the suitability of three chosen criteria, that are 1) maximum parsimony of the algorithmic model, 2) ease of understanding, and 3) high performance, four algorithms were selected: two algorithms fully meeting the criteria (POSMIT and SOC), one algorithm partially meeting the criteria (MBGP algorithm), and one algorithm which does not correspond to them (CandidateStops). For each parameter of these algorithms, a vector of possible values was defined, which was then applied to the algorithm in every possible combination to find the optimal set of parameters. The algorithms were then compared based on F-measure, average number and variance of stops per day, robustness to noise as well as varying sampling rates and compactness of shape. Since the calculation of F-measure values requires ground truth, 90 days of data were manually labelled using an R Shiny Application. Furthermore, three scenarios were analysed out to find the algorithm that: 1) best handles noisy data, 2) works well with varying sampling rates, and 3) has the most compact shapes.

As a result, the MBGP algorithm performed the best for the sampled MOASIS data in the first and second scenario as well as regarding the average number of stops per day. The POSMIT algorithm suited the analysed MOASIS data the best in terms of the third scenario. The least suitable algorithm for the sampled MOASIS data was the CandidateStops algorithm. The result suggests that it is important to ensure that stop-move detection algorithms have at least a spatial and a temporal parameter.

This thesis set a base for further movement analysis at an individual level by analysing and proposing an algorithm that performs the sampled MOASIS data the best, which is the MBGP algorithm with the given setting: Dmax = 275 m, Tmax = 14400 s, and Tmin = 900 s. Since the F-measure values that could be calculated for the 90 test days were high, future research should mainly focus on generating more valid results instead of further increasing accuracy as well as including the interrater agreement to better assess the quality of the manually labelled ground truth.

Acknowledgements

First of all, I would like to thank my main advisor, Dr. Eun-Kyeong Kim, for the informative meetings and competent and helpful support during the whole process of writing the Master's Thesis. Many thanks as well to the faculty representative, Prof. Dr. Robert Weibel, for all the valuable feedback and to my two concept reviewers, Julia Villette Ardito and Dr. Peter Ranacher, for the constructive inputs.

I am further very grateful to my proofreader, Tim Waldburger, who has given me useful tips for the improvement of my work, as well as to my fellow roommate, Joseph Bailey, who proofread my English texts. I would also like to thank the MOASIS research teams for allowing me to use the MOASIS data for my thesis and Michelle Fillekes for making available the MBGP algorithm's R-Codes.

Finally, warmest thanks to my partner, friends, colleagues of work, and family for all the support, understanding, and encouragement during this project. I especially appreciated the coffee breaks and thematic exchange with Nathalie and the never-ending patience and comforting words of my partner. Tim, thank you for always being there for me!

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

B

BIRCH	Balanced Iterative Reducing and Clustering using Hierarchies

C

C-DBSCAN	Constrained Density-Based Spatial Clustering of Applications with Noise
CB-SMoT	Clustering-Based Stops and Moves of Trajectories
CLAR	Collaborative Location and Activity Recommendation

D

DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DB-SMoT	Direction-Based Stops and Moves of Trajectories
DCC	Direction Change Coefficient

E

EAR	Electronically Activated Recorder

G

GPS	Global Positioning System

H

HDOP	Horizontal Dilution of Precision

I

IMU	Inertial Measurement Unit

M

MI	Moment of Inertia
MOASIS	MObility, Activity and Social Interaction Study
MSN	Move-Stop-Noise

O

OPTICS	Ordering Points to Identify the Clustering Structure

P

P-DBSCAN	Pre-Processing and post-processing techniques that are used in combination with an unmodified version of DBSCAN
PIE	Point-of-Interest Extraction
POLOI	Polygon-of-Interest
POSMIT	Probability of Stops and Moves in Trajectories

S

SMoT	Stops and Moves of Trajectories
SOC	Sequence Oriented Clustering
STC-SMoT	SpatioTemporal Clustering-based Stops and Moves of Trajectories

T

TDBC	Time and Distance Based Clustering
TOSCA	TwO-Steps parameter free Clustering Algorithm
TraClus	Trajectory Clustering
TrajDBSCAN	Trajectory Density-Based Spatial Clustering of Applications with Noise

V

VDOP	Vertical Dilution of Precision

1 Introduction

An aging demographic was estimated by looking at the current development of the Swiss population (Swiss Confederation, 2016). To cope with the potential high demand of healthcare services, the Swiss healthcare system has to manage its capacities in order to deal with the older population (ibid.). More importantly, it is imperative to promote healthy aging (WHO, 2017). There are many aspects of healthy aging including physical/social activity, and cognitive well-being, as well as their interactions (Bereuter, Fillekes and Weibel, 2016). For instance, physical activity has a positive impact on the cognitive function of older adults[1] (Voelcker-Rehage, Godde and Staudinger, 2011). The real-life assessments using advanced sensors and mobile devices have enabled examining the relationship between spatial use, physical and social activity and cognitive well-being beyond laboratory settings (Bereuter, Fillekes and Weibel, 2016).

The interdisciplinary *MObility, Activity and Social Interaction Study (MOASIS)* research project was conducted at the University of Zurich. Over a period of 30 days, data on individual-level mobility, physical activity, and cognitive status from the everyday lives of older adults from German-speaking Switzerland were collected by using a wearable device, *uTrail*, equipped with mobile sensors (e.g., global positioning system (GPS), accelerometer) and an audio recorder (Bereuter, Fillekes and Weibel, 2016; Fillekes, Röcke, *et al.*, 2019).

Measuring the individual-level mobility in geographic space by GPS is useful to understand the healthy aging of older adults, as there is a relationship between the basic human need of physical movement and mobility. However, this need can often not be satisfied due to environmental barriers to mobility, which can lead to a decline in older adults' mobility (Rantanen, 2013). This also limits their social activity and prevents older adults from using public transport, which is mainly seen as main driver of physical activity (Chaix *et al.*, 2019). To investigate the influence of environmental barriers on mobility and thus on healthy ageing, GPS is often used as an efficient and cost-effective source of data (Hirsch *et al.*, 2014; Chaix *et al.*, 2019).

To extract meaningful information on daily mobility and activities from collected raw trajectory data (e.g., GPS), it is critical to design automatic pre-processing methods including outlier removal, stop-move detection, and semantic enrichment (Zheng, 2015). Particularly, stop-move detection algorithms are known to play a significant role in determining the number of activity locations and the degree of mobility (Fillekes, Kim, *et al.*, 2019; Fillekes, Röcke, *et al.*, 2019). Therefore, it is important to detect stops and moves in trajectories correctly, otherwise, the results of the analysis of the individual's mobility level could be distorted. Wrong outcomes (e.g., distorted travel distances), in turn,

[1] The term "older adults" describes people above retirement age (which is around 65 years in Switzerland).

would lead to the drawing of false conclusions about the older adults' state of health and could make planning processes inefficient or counterproductive.

Apart from health planning processes, detecting stops in the trajectories of older adults correctly is also useful in urban and facility planning as older adults have different mobility patterns than younger people. According to Marcum (2013), older adults spend less time travelling and communicating compared to younger people. Instead, they spend their time on individual activities, such as doing housework. The reason for this is that younger people tend to have bigger and more diverse social networks than older adults, as they usually live in multi-person households and travel more among people because of work or education (Marcum, 2013). When we know where older adults make stops, public places could be adapted to their needs such as building escalators instead of stairs in public areas, positioning defibrillators, creating green areas, or placing benches to recover on the way.

1.1 Problem Statement

Until now, discrepancies exist between the error-prone GPS data collected by smart phones and the requirement of accurate fine-scale estimates of trajectories (Li *et al.*, 2019). Therefore, the trajectory reconstruction from GPS data (e.g., stop-move detection) is of substantial importance in trajectory data analysis. Depending on the stop-move detection algorithm used, automatically detected stops and moves may vary, yielding various mobility patterns and performances in estimation accuracy. Hence, it is necessary to be aware of the impact of selected algorithms on the accuracy and uncertainties of mobility lifelines by the comparison and evaluation of those algorithms. However, there is a lack of systematic evaluation criteria and profound comparison of the different algorithms in literature.

Selecting a stop-move detection algorithm involves decision making on various aspects of the algorithm: 1) optimal parameter setting, 2) consideration of missing data, and 3) robustness of results to the input parameters.

First, different algorithms have different parameters and optimal thresholds. There is still need for more studies on finding the optimal thresholds of a stop-move detection algorithm. One example is the recent research of Fillekes, Kim, *et al.* (2019). They performed a sensitivity analysis on the parameter thresholds for the stop-move detection algorithm proposed by Montoliu, Blom and Gatica-Perez (2013).

Second, only few algorithms consider a missing data problem. Informed selection of stop-move detection algorithms optimized for the data could make research more reliable and reduce interpreting false correlations.

Third, according to Xiang, Gao and Wu (2016), the existing research on trajectory data has mainly focused on data management, data mining, and query techniques but neglects the impact of algorithms and parameters on the results (ibid.). Algorithms should be selected that are the least sensitive (i.e., more

robust)[2] to data size and input parameters, as well as most accurate. The low sensitivity to the input data size is important because, even within the same dataset, the data size of each individual or each day may vary due to behavioural fluctuations or sensor types or amount of time spent in-/outdoors, and overly sensitive algorithms will yield inconsistent results of detected stops and moves. Algorithms that are robust to the input parameters can potentially serve less experienced researchers even without profound background knowledge and it may simplify the pre-evaluation of the sensitivity of candidate algorithms thanks to expected consistency by nature.

1.2 Research Questions

The aim of this thesis is to review and compare stop-move detection algorithms for GPS data collected from the healthy aging research project to analyse the mobility patterns of older adults from a spatial and temporal perspective. The focus of this thesis is an analysis of stop-move detection algorithms based on a profound literature-based overview. Based on these findings, four algorithms are selected, compared, evaluated, and implemented in the programming language R. The goal is to find out which of these algorithms suit the specific MOASIS dataset the best. For doing so, it is important to compare the advantages and disadvantages of the algorithms and learn about their different input parameters. In the thesis, the following research questions (RQ) will be addressed at both conceptual and analytical levels:

1.2.1 RQs at a Conceptual Level

RQ1: What are the conceptual criteria to evaluate the stop-move detection algorithms?

This research question is a conceptual question and will mainly be answered through a literature review. According to Braune, Besecke and Kruse (2015), there will hardly be an algorithm that completely fulfils all of these criteria.

RQ2: Which evaluation measure should be used to compare the stop-move detection algorithms?

This question will be answered through a literature review as well. It is expected that sensitivity can be measured based on shape, precision, and the average number and variance of stops per day.

[2] Robustness meaning in this thesis: the algorithms' consistency is independent of the size of the input dataset and the selected parameters.

1.2.2 RQs at an Analytical Level

RQ3: *To what extent do spatial and temporal parameters affect the results of detected stops and moves? How sensitive are the algorithms to the number of input data points, speed variation, temporal sampling interval, and data quality?*

The three types of parameters that use spatial, directional, and temporal traits of movements are used in most of the stop-move detection algorithms, as stated by Zadeh Monajjemi (2013). Previous studies have found that different algorithms perform differently to those input parameters and the number of input data points. Bermingham and Lee (2018) concluded that the probability-based algorithms' results are less sensitive to changes in the spatial parameter than grid-based and density-based algorithms are to their respective spatial parameters. The probability-based algorithms were also less sensitive to the number of input data points (ibid.). Hence, the following three hypotheses are drawn.

H3-1: The detection results of the probability-based algorithms are less sensitive to changes in spatial parameters for the specific MOASIS dataset than for grid- or density-based algorithms.

H3-2: The probability-based algorithms perform better than the other algorithms when the sampling rate is bigger than 1 s.

H3-3: The detection results of density-based algorithms using a minimum stop duration threshold are most sensitive to temporal input parameters as setting the optimal duration threshold is not trivial (Gong *et al.*, 2015).

RQ4: *How do spatial, temporal and geometric characteristics of detected stops and moves vary depending on the applied algorithms?*

Detected stops are often characterized as regions where the data points are spatially denser than during periods of movement (Nogueira, Martin and Andrade, 2017). This density and other characteristics of the detected stops change depending on the applied algorithm. Some algorithms only seek for core sequences of the stop cluster and have to fulfil many criteria in order to detect a stop, other algorithms adopt fewer or relaxed criteria to decide whether a data point belongs to the stop or move cluster (Nogueira, Braga and Martin, 2014; Xiang, Gao and Wu, 2016). This leads to the following hypotheses regarding selected algorithms and the characteristics of detected stops.

H4-1: The spatial extent varies between the applied algorithms with respect to shape, size, and spatial extent. A stop can be classified as an area or as multiple smaller points. Since a high resolution of detected stops is anticipated to occur, especially without post-processing the results, the algorithms are expected to find smaller stop points instead of one bigger area. Therefore, all algorithms will deliver more false positives than false negatives when compared to the ground truth.

H4-2: The density-based algorithms deliver most compact stops for the specific MOASIS dataset due to their ability to build clusters of contiguous data points that are close to each other in space and time (Bermingham and Lee, 2018).

H4-3: The least compact stops result from the geography-based algorithms as they use predefined geographic geometries to detect stops (Bermingham and Lee, 2018). Hence, it can be assumed that indoor movements, for instance, in a building will not be detected easily.

1.3 Thesis Structure

The main part of this thesis is divided into seven chapters. *Chapter 2* first explains the theoretical background and then presents the relevant literature for this project. The first part deals with the concept of trajectories and their properties (e.g., path characteristics) in general as well as with the concept of trajectory segmentation and the definitions of the basic terms (e.g., stop and moves). In the second part of this chapter, existing stop-move detection algorithms as well as their grouping schemes are described. *Chapter 3* presents the MOASIS data, which are the basis for this project. In *Chapter 4*, the methodological framework is presented in three steps and the procedure of this project is explained in detail. This chapter contains the methods that are necessary to evaluate the implemented stop-move detection algorithms introduced in *Chapter 2*. *Chapter 5* then presents the results of analysis and evaluation of the algorithms. In *Chapter 6*, the results are discussed in response to the research questions proposed in *Chapter 1*. *Chapter 7* draws a conclusion and further research approaches are proposed.

2 Theoretical Background and Framework

Individuals change their spatial location over time. This can be seen as a fundamental characteristic of life and is driven by processes that operate across multiple temporal and spatial scales (Nathan *et al.*, 2008). These processes can be explored through movement research to get a detailed view of individual movement (ibid.). Traditional methods for getting an insight into the movement behaviour of individuals (i.e., direct observations, interviews, and diaries) are put into question in terms of reliability, compliance, reproducibility, and recall. With the use of GPS devices, individuals' movements can be quantified with compensating the disadvantages of the traditional approaches (Vazquez-Prokopec *et al.*, 2009). GPS devices record the individuals' personal raw movement track data that typically consist of spatiotemporal points p = (x, y, t) (Parent *et al.*, 2013; Zheng, 2015; Zhang, Wang and Huang, 2019). Depending on the GPS device, additional data (e.g., speed, acceleration, rotation, and direction) could supplement the spatiotemporal point data (Parent *et al.*, 2013).

2.1 Trajectories

Since raw data usually includes outliers and noise, many applications are not interested in analysing the entire quantities of unprocessed raw data. Therefore, often only the meaningful movements are extracted through pre-processing (Parent *et al.*, 2013). According to Parent *et al.* (2013), these segments are called trajectories (see *Figure 2.1*). Each trajectory contains two specific positions of the movement track, meaning the *Begin* (individuals' first position) and *End* (individuals' last position) of the trajectory (ibid.).

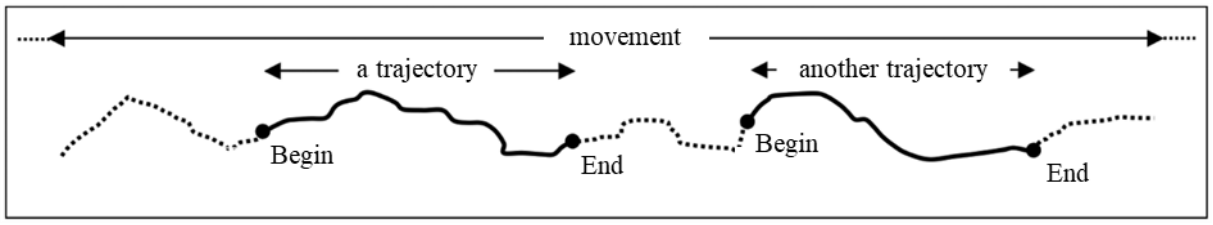


Figure 2.1 Movement track (dotted line) with extracted trajectories (from Parent *et al.*, 2013:4)

Bermingham (2018) refines the trajectory definition as stated in Parent *et al.* (2013) and distinguishes between spatial trajectories and spatiotemporal trajectories. Spatial trajectories contain the minimal amount of information that a movement track can have (i.e., a list of spatial entries, meaning spatial/geographic coordinates, *X* and *Y*) as shown in *Equation 2.1* (Bermingham, 2018).

$$T_{spatial} = \{X_1, Y_1\}, \ldots, \{X_n, Y_n\} \quad (2.1)$$

Alongside this spatial information, many other attributes are typically recorded quite often (ibid.). In many recordings, timestamps can be found. These trajectories, that incorporate spatial (*X* and *Y*) as well as temporal information (*t*) are called spatiotemporal trajectories (see *Equation 2.2*) (ibid.).

$$T_{spatiotemporal} = \{X_1, Y_1, t_1\}, \ldots, \{X_n, Y_n, t_n\} \quad (2.2)$$

However, spatiotemporal trajectories are complex structured data and are difficult to handle efficiently (Damiani and Hachem, 2017). Even simple operations (e.g., range and spatiotemporal join queries) are often computationally costly. Hence, the efficient access of spatial trajectory data needs to be further researched (ibid.). Especially when the knowledge granule is not the spatiotemporal coordinates itself, but rather the individuals' behaviour in time, efficient data handling is of importance. The extracted information encoded into a semantic form reduces the size of the dataset and makes accessing relevant information of the summarized spatial trajectory easier (ibid.). Resulting out of this approach, the notion semantic trajectory has become popular in recent times (Parent *et al.*, 2013; Damiani and Hachem, 2017). Semantic trajectories represent the time-varying behavioural information of single individuals (see *Figure 2.2*) (Damiani and Hachem, 2017).

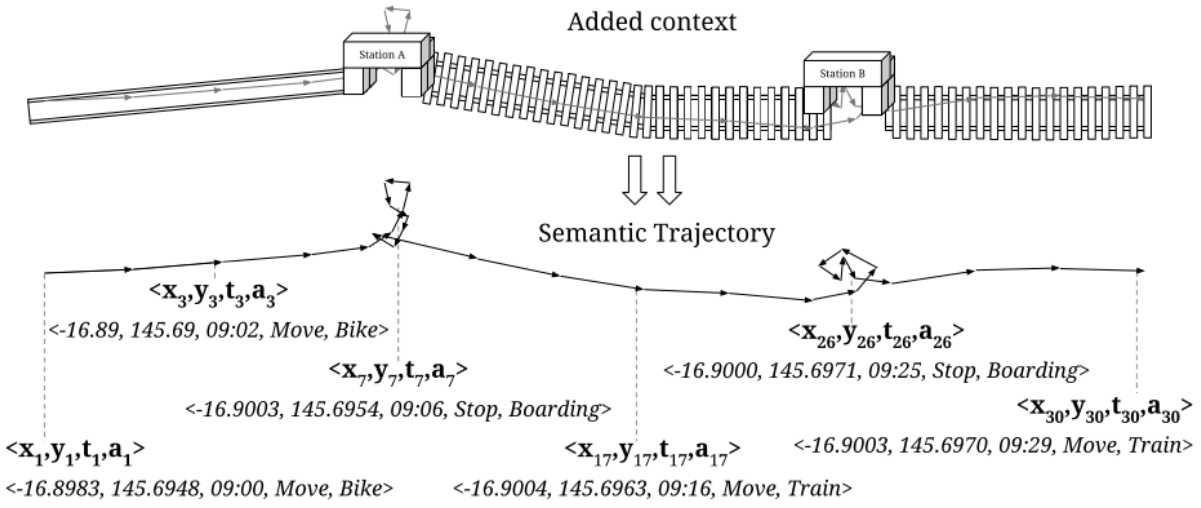


Figure 2.2 Example of a semantic trajectory (from Bermingham, 2018:3)

The concept of adding knowledge to trajectories is called semantic enrichment process (Parent *et al.*, 2013). Parent *et al.* (2013) define additional data that are attached to subparts of a trajectory or the trajectory as a whole, as annotations (*a*) (see *Equation 2.3*).

$$T_{semantic} = \{X_1, Y_1, t_1, a_1\}, \ldots, \{X_n, Y_n, t_n, a_n\} \quad (2.3)$$

Annotation values can be either an attribute value, a link to an object in the contextual data repository, or a complex value composed of attribute values and links to objects (Parent *et al.*, 2013). There are

several ways of capturing annotation values. They are usually captured by observers and sensors, computed from raw data, or extracted from contextual data (ibid.).

2.1.1 Stop Characteristics

A stop point stands for a visited location (*X*,*Y*) where a person arrived at a specific time (t_{start}) and left at another time (t_{end}) as *Equation 2.4* shows (Fu *et al.*, 2016).

$$P_{stop} = \{X, Y, t_{start}, t_{end}\} \tag{2.4}$$

Fu *et al.* (2016) distinguish between three types of stops as can be seen in *Figure 2.3*. The first type of stop is mostly the end or start point of each trajectory. Fu *et al.* (2016) argue that GPS data collection mostly starts from a location that a person leaves (e.g., leaving home) and ends in a position that is without a signal for a long time (e.g., arriving home). However, the end of a trajectory does not always have to indicate a stop. A trajectory may end up with other factors, such as a low battery (Fu *et al.*, 2016). The second type of stop is a spatiotemporal cluster of GPS points that appears when a person stays in one place for a long time without losing the GPS signal (ibid.). The third type of stop occurs when a person stays in a region where the GPS signal is lost due to buildings or barriers (ibid.).

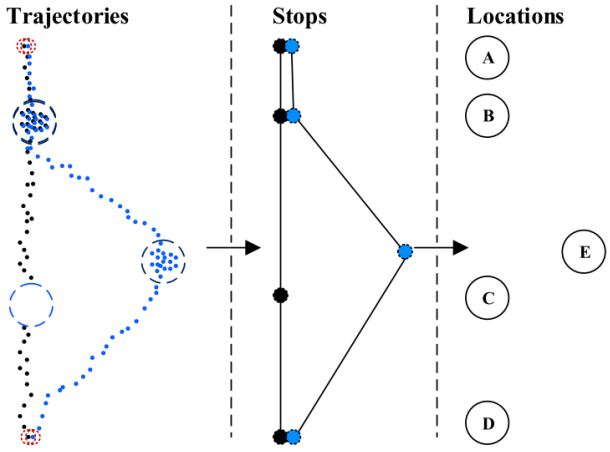


Figure 2.3 Stop and location extraction from trajectories (from Fu *et al.*, 2016:3)

When the stop points are clustered, a group of location points that are frequently visited by a person can be detected. Tran *et al.* (2011) distinguish between shared and personalized location points. Shared stops are location points where many different moving objects pass by and stay for a while such as public places. Personalized stops, on the other hand, are individual places where mostly only a few moving objects stop as, for example, the home of a tracked person (Tran *et al.*, 2011).

Furthermore, there is differentiation between generic stops and concrete stops in terms of the stop characteristics; this is useful because determining the suitable stop size often is difficult, as the

granularity (i.e., spatial scale) of the stops can differ (ibid.). Generic stops can be defined as more general stops – e.g., a shopping mall as opposed to the individual stores a person visited (ibid.). In such an example, concrete stops are the individual shops or restaurants in the shopping centre, indicating more specific and small-scale stops. Tran *et al.* (2011) state that most applications require the knowledge about concrete stops instead of generic ones. The authors therefore suggest splitting large stops into smaller, more concrete ones. As *Figure 2.4* shows, this idea can be visualised through a tree-based stop hierarchy.

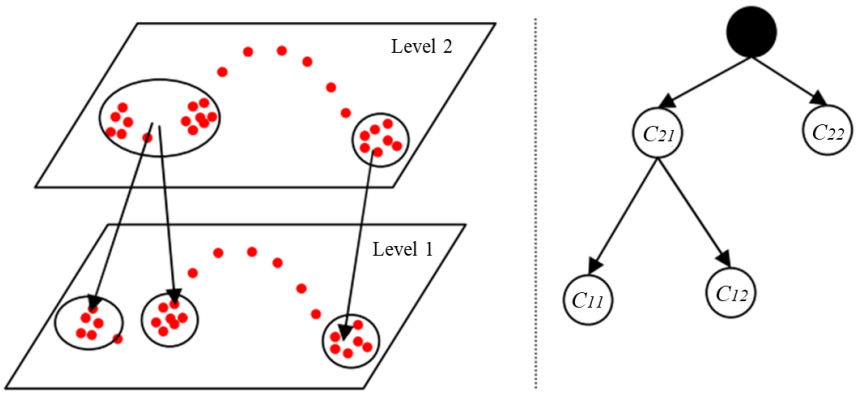


Figure 2.4 Concept of hierarchical stops (from Tran *et al.*, 2011:2)

As seen in *Figure 2.4*, the stops in level 2 are more generic ones and contain more aggregated information on the potential stops, whereas the stops in lower level 1 are in small size and contain spatially higher-resolution information on the stops.

Besides the spatial extent of a stop, a stop can also be characterised by its level of motion. A stop (i.e., nearly absence of motion) is characterized by a velocity less than 1 m/s (Nogueira, Braga and Martin, 2014). This threshold may serve as a good approximation, but it has to be adapted depending on the mode of transportation (ibid.).

To not only get information on the stops themselves but also on how they are connected and in what frequency they are visited, it is also important to take into account the parts in between two stops: the moves (Fu *et al.*, 2016).

2.1.2 Movement Patterns

In general, moving objects can be defined as entities whose geometric attributes or positions change over time. Nogueira, Braga and Martin (2014) characterize steady human motion with velocity greater than 1 m/s and acceleration values between -0.3 m/s^2 and 0.3 m/s^2. If the velocity increases and acceleration is less than -0.3 m/s^2, negative acceleration happens whereas positive acceleration happens when the acceleration is greater than 0.3 m/s^2 at a higher velocity than 1 m/s. Hence, movement is a

change of position when considered as physical phenomenon (ibid.). Dodge, Weibel and Lautenschütz (2008) propose a conceptual framework of movement as visualised in *Table 2.1*. There are three major groups of movement parameters: primitive parameters, primary derivates, and secondary derivates (Dodge, Weibel and Lautenschütz, 2008). The primary derivates and secondary derivates can be summarized under the generic term derived parameters. The parameters have spatial, temporal, and spatiotemporal dimensions (ibid.).

As *Table 2.1* shows, the temporal dimension of primitive parameters consists of instance, a point in time, and interval, a temporal sampling rate. Several derived parameters can be defined from the primitive parameters. A direct function of position are distance, direction, and spatial extent which can be assigned to the group of primary derivatives (ibid.). Duration consists of at least one time interval and is defined as a period of time in which a movement is observed. The space and time dimensions can be directly derived from spatial positions and time instances and consist of speed and velocity (ibid.). The speed is defined as "rate of change of the object's position" and the velocity is defined as "rate of change of position and direction" (Dodge, Weibel and Lautenschütz, 2008:3). Higher-order movement parameters grouped into the secondary derivates can be derived from primary derivates. Besides defining movement parameters in an absolute sense, defining them in a relative sense (i.e., an object's movement relative to the movement of otherer moving objects) is also preferable when at least two moving objects are to be analysed (ibid.).

Table 2.1 Parameters of movement (modified from Dodge, Weibel and Lautenschütz, 2008:243)

Parameters / Dimension	Primitive	Primary derivatives	Secondary derivatives
Spatial	Position (x,y)	Distance $f(posn)$	Spatial distribution $f(dist)$
		Direction $f(posn)$	Change of direction $f(dir)$
		Spatial Extent $f(posn)$	Sinuosity $f(dist)$
Temporal	Instance (t)	Duration $f(t)$	Temporal distribution
	Interval (t)	Travel time $f(t)$	Change of duration $f(dur)$
Spatiotemporal (x,y,t)	-	Speed $f(x,y,t)$	Acceleration $f(speed)$
		Velocity $f(x,y,t)$	Approaching rate

In terms of the nature of movement, human beings share some parallels with, but also show discrepancies to others. According to Marcum (2013), older adults have smaller, less diverse, and more family-centric networks than younger people. These movements can take place at different temporal and spatial scales (Dodge, Weibel and Lautenschütz, 2008). In movement pattern analysis, scale plays an important role. The spatial scale of a movement can range from a very local scale (e.g., movements at home) to a global scale (e.g., movements with airplanes). The temporal scale can range from a very

short-term behaviour to a long-term one (ibid.). Marcum (2013) emphasises that time use especially differs between older and younger people. As older adults need more privacy in order to perform personal care activities, they tend to spend less time with others (Marcum, 2013). Furthermore, older adults might have fewer work and family obligations and are therefore in a different structural position than younger people. This means that older adults have more leisure time and usually cannot sleep through the night. Consequently, they need more time to rest during the day, whereas younger people have more daily routines (e.g., time at work) and have less time for daytime naps (ibid.). However, studies showed that older adults should stay in contact with others as it is important for staying healthy. In other aspects, time use does not change with age as a work-by-day, sleep-by-night pattern is what people tend to strive for (ibid.).

As the cycle of daily life (e.g., wake up in the morning, eat, brushing teeth) becomes standardized over time, it is still part of the time budgets of older adults, although some factors (e.g., work) lead to age-associated changes (ibid.). Hence, movement patterns of older adults tend to concentrate mostly around their home location, as the top three leisure activities for older people are television watching, reading, and reflecting alone (ibid.).

2.2 Trajectory Pre-processing

In many scenarios, such as trajectory clustering and classification, trajectories need to be divided into segments for further processes (Zheng, 2015). This fundamental step of many trajectory data mining tasks is called trajectory pre-processing (see *Figure 2.5*).

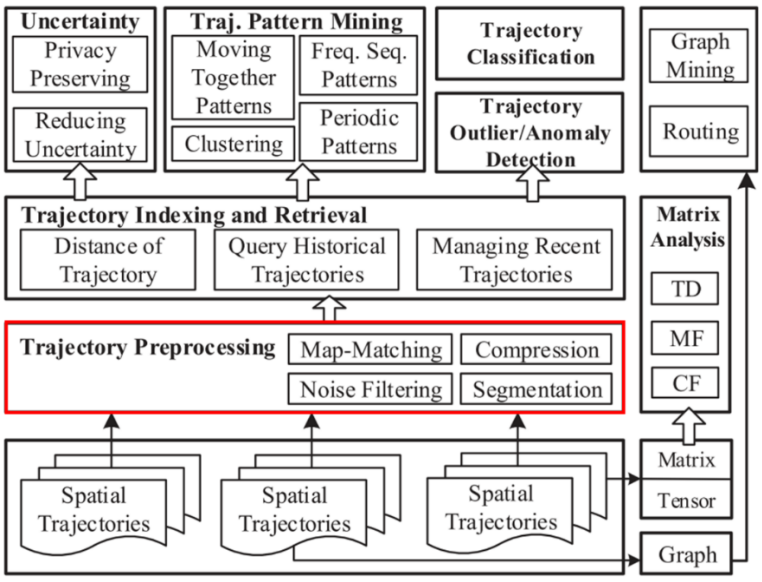


Figure 2.5 Trajectory data mining (modified from Zheng, 2015:2)

Trajectory pre-processing includes (amongst others) four parts: Map-Matching, Compression, Noise Filtering, and Segmentation (Zheng, 2015).

2.2.1 Map-Matching

The process that converts a sequence of raw latitude and longitude coordinates to a sequence of road segments is called map-matching (Zheng, 2015). Map-matching methods can be classified based on the additional information used, or the range of sampling points considered in a trajectory (ibid.).

The first classifying approach using the additional information can be divided into four groups: geometric, topological, probabilistic, and other advanced techniques (ibid.). Geometric map-matching algorithms take into account the shape of a road network's individual connections (e.g., adapting a GPS point to the nearest road). Topological algorithms consider the road network's connectivity of a road network (ibid.). To deal with noisy and low-sampled trajectories, probabilistic algorithms explicitly pay attention to GPS noise and consider several possible paths through the road network to find the best one (ibid.). There are more advanced map-matching algorithms that consider both the road network's topology and the trajectory data's noise. These algorithms find a sequence of road sections that simultaneously approximate the noisy trajectory data and form a route through the road network (ibid.).

The second classifying approach, the range of the sampling points considered, can be divided into two categories: local/incremental and global methods (ibid.). The local/incremental algorithms follow a greedy strategy to sequentially extend the solution from an already adapted part. Based on distance and orientation similarity, these methods try to find a local optimal point (ibid.). The local/incremental methods run very efficiently, often used in online applications. However, if the sampling rate of a trajectory is low, the matching accuracy deteriorates. As an alternative, global methods aim at matching an entire trajectory with a road network, e.g., by considering the successors and predecessors of a point (ibid.). Global methods are more accurate, but less efficient than local methods, and therefore usually used for offline tasks, meaning where entire trajectories have already been generated (e.g., mining of common trajectory patterns) (ibid.).

This map-matching approach is especially important for research concerning traffic flow, guiding a vehicle's navigation, or predicting where a vehicle is going (ibid.). Therefore, knowing about which road a vehicle was on is important. However, parallel roads, overpasses, and spurs can make map-matching a complex problem (Krumm, 2011 in Zheng, 2015).

2.2.2 Compression

With GPS-equipped devices, timestamped geographical coordinates can be recorded every second for a moving object. Unfortunately, that costs a lot of battery power and needs a lot of data storage. Also, many applications do not really require such spatial accuracy (Zheng, 2015). To solve this problem, two categories of trajectory compression strategies based on the shape of a trajectory have been proposed: offline compression and online compression (ibid.). These strategies aim to reduce the size of a trajectory without compromising the precision of the new data representation (Lee and Krumm, 2011 in

Zheng, 2015). Offline compression, or batch mode, reduces the size of the trajectory after the trajectory has been generated completely. Online compression compresses a trajectory immediately as an object moves (Zheng, 2015).

2.2.3 Noise Filtering

Due to sensor noise and other factors (e.g., poor GPS signals in urban environments), spatial trajectories are never entirely accurate (Zheng, 2015). Sometimes the error is acceptable because it can be fixed with map-matching methods (e.g., some GPS points do not perfectly fit the road the vehicle drove on). In other scenarios, the error of a noise point is too large, because, for example, it is several hundred meters away from its actual position. In this case, useful information such as driving speed cannot be derived (ibid.). Therefore, such noise points have to be filtered out of the trajectories before a mining task can be performed. The existing methods can be divided into the three main categories: mean (or median) filter, Kalman and particle filters, and heuristic-based outlier detection (ibid.).

The mean (or median) filter in a way smooths the trajectory as it replaces each value with the mean (or median) of a fixed amount of data points (Zheng, 2015). It can be seen as a sliding window. For handling extreme errors, the median filter seems to be more robust than the mean filter (ibid.).

For the Kalman filter method, the data are expressed as a state space model (Park *et al.*, 2019). To this model, probabilistic estimations are applied in order to estimate the sensor values. The Kalman filter requires a measurement noise variance as input parameter (ibid.). However, as the sensor data are rather complex, acquiring the detailed noise information is difficult and can lead to a poor filtering performance (ibid.).

The heuristic-based outlier detection methods do not aim to replace the trajectory values. Instead, they directly remove the noise points from the trajectories (Zheng, 2015). As an example, removing speed outliers in GPS data of older adults by setting a speed-threshold that must not exceed the speed value of the fastest high-speed train (e.g., 330 km/h in Germany), which is used by Fillekes, Kim, *et al.* (2019).

2.2.4 Segmentation

Through segmentation, the computational complexity can be reduced. Segmentation further allows us to mine richer knowledge from which we can learn more than from an entire trajectory (Zheng, 2015). The segmentation methods can be grouped into two categories: attribute-driven and pattern-driven (Damiani and Hachem, 2017).

2.2.4.1 Attribute-Driven Segmentation

Attribute-driven segmentation partitions a spatial trajectory in a minimum number of segments. The goal is that movements inside this trajectory are nearly uniform with respect to some condition on

movement attributes (Damiani and Hachem, 2017). The segmentation criteria can be divided into the three categories: application specific, monotone, and non-monotone (ibid.).

Application specific criteria are criteria which have been selected according to the requirements of a specific task (ibid.). Instead of defining criteria on an ad-hoc basis, monotone criteria are defined in a more general model. A monotone criterion is granted, if any sub-trajectory that is based on a given criterion (e.g., speed < 50 km/h) of a sub-trajectory is sure to follow the given criterion (ibid.). Segmentation techniques meeting monotone criteria can be computed efficiently. A non-monotone criterion is granted, if a further criterion is added to build a sub-trajectory (e.g., speed < 50 km/h for at least 5 hours) and it cannot be assumed that every sub-trajectory of the sub-trajectory fulfils this second criterion (ibid.). Due to outliers causing incorrect breakpoints not all criteria of practical interest can be implemented using monotone segmentation criteria. Therefore, a theoretical framework based on non-monotone segmentation criteria has been introduced by Aronov *et al.* (2015). However, the trajectory segmentation using non-monotone criteria is more computationally complex (Aronov *et al.*, 2015; Damiani and Hachem, 2017).

2.2.4.2 Pattern-Driven Segmentation

Pattern-driven segmentation methods are usually based on machine learning (Damiani and Hachem, 2017). Particularly, the unsupervised segmentation methods including clustering are part of the pattern-driven segmentation. Clustering based segmentation techniques are often used for the detection of specific behaviours or patterns that can either be concentrated on specific domains (e.g., human mobility) or be generic (e.g., stop-move patterns) (ibid.). Stop and move patterns can be seen as an abstraction of an object's mobility behaviour; in other words, stops and moves are essential for understanding the moving object's mobility because it repeatedly stays somewhere for a while before moving somewhere further. Thus, stop-move pattern detection is essential in this thesis.

2.3 Stop-Move Detection Algorithms

Stop-move detection algorithms automatically identify stops and moves in trajectories and can have the following input parameters: spatial threshold, temporal threshold, directional threshold, geographic places, minimum number of points, and non-attribute related thresholds (Bermingham, 2018; Bermingham and Lee, 2018). Depending on the applied algorithm, the single or multiple parameters can be used. Further, there are multiple ways of grouping these algorithms. The algorithms can be grouped according to either core algorithmic procedures to detect stops or presumptions/requirements for the task and input data to apply the algorithm.

2.3.1 Classification Scheme

Stop-move detection algorithms can be grouped into three categories in terms of characteristics of algorithms and predefined geographic objects: density-based, geography-based, and probability-based algorithms (Bermingham and Lee, 2018). In terms of how the geographic space is partitioned into subspaces (i.e., different geographic shapes), stop-move detection algorithms can be classified into grid-based and clustering-based (including density-based, centre-based, and hierarchical-based) algorithms (Guidotti, Trasarti and Nanni, 2015). *Table 2.2* gives an overview of the algorithms that are classified according to two criteria: 1) core procedural characteristics of the algorithm and 2) the method of partitioning space into subspaces. The algorithms in the table are linked to their respective description.

Table 2.2 Algorithm grouping scheme

		partitioning of space into subspaces			
		grid-based	clustering-based		
			density-based	centre-based	hierarchical-based
characteristics of algorithms	density-based	Grid-growing clustering algorithm, Velocity-based trajectory structure	CandidateStops, DB-SMoT, Novel kernel-based algorithm, MBGP, MSN, Simple stop detection algorithm, STC-SMoT	C-DBSCAN, DBSCAN, OPTICS, P-DBSCAN, SeqScan, SOC, StopFinder, TraClus	TDBC, TOSCA, TrajDBSCAN
	geography-based		CB-SMoT, SMoT	PIE	SMoT+
	probability-based			POSMIT	

Regarding the criterion based on core procedural characteristics, algorithms can be classified into 1) density-based, 2) geography-based, and 3) probability-based algorithms.

First, the density-based algorithms automatically build clusters of contiguous data points that are spatiotemporally next to each other. The non-clustered data points are labelled as moving, the clustered data points as stopping events of the trajectory (Bermingham and Lee, 2018).

Second, the geography-based algorithms search for a subsequence of connecting trajectory entries that match with predefined geographic geometries (e.g., buildings, footpaths) to discover stops (ibid.). The other recordings outside those predefined geometries are not reasonable stop candidates and therefore not considered as stops. Further, a stop is only valid if the subsequence of connecting trajectory entries matches with the geometry for a minimum user-specified duration (ibid.).

Third, probability-based stop-move detection algorithms were recently proposed by Bermingham and Lee (2018) as an additional category. These algorithms calculate the probability that the person truly stopped at a given data point.

Based on an algorithm's partitioning method for a study region, algorithms are categorized into 1) grid-based and 2) clustering-based algorithms. First, the grid-based algorithms divide the space into cells of same size and aggregate the data points according to their density (Guidotti, Trasarti and Nanni, 2015).

Second, the clustering-based algorithms can be further divided into centre-based clustering methods, hierarchical clustering methods, and density-based clustering methods (ibid.). Centre-based clustering methods often split the entire set of location points into several groups that are loosely connected although they tend to correctly identify subgroups of observations that should belong to the same location (ibid.). In an opposite way, hierarchical-based clustering methods tend to spot the loose connections well but are not good at distinguishing those that are actually boundaries with other clusters (ibid.).

2.3.2 Algorithm overview

In literature, there exist numerous different algorithms. Some algorithms have many input parameters allowing more flexibility and fine-tuning, while adding more complexity. Other algorithms have a few input parameters which makes them not only tuneable, but also simpler to adjust.

Therefore, *Table 2.3* gives an overview of the algorithms grouped in *Table 2.2*. Algorithms 1-5 are based on the SMoT algorithm, algorithms 6-11 are based on the DBSCAN algorithm, algorithms 12-13 are based on OPTICS, algorithms 14-23 are variations of the algorithms mentioned in 1-13, and algorithms 24-25 are grid-based algorithms.

Table 2.3 Overview of stop-move detection algorithms and their parameters

#	**Name of stop-move detection Algorithm (Abbreviation)** *Algorithm's input parameters* Description of the algorithm
1	**Stops and Moves of Trajectories (SMoT)** *T, set of geometries, t_{min}* The SMoT algorithm verifies for each point of a GPS trajectory if it intersects the geometry of a manually selected candidate stop (e.g., buildings) (Alvares *et al.*, 2007). If the duration of the intersection is at least equal to a given temporal threshold (t_{min}), the intersected candidate stop is considered as a stop (ibid.). However, stops that are not in the manually selected candidate stops are not detected (ibid.).
2	**Direction-Based Stops and Moves of Trajectories (DB-SMoT)** *T, t_{min}, dir_{min}, tol_{max}* The DB-SMoT algorithm considers the direction as main threshold to find the clusters (dir_{min}) (Rocha *et al.*, 2010). The algorithm further uses the parameters t_{min} and tol_{max}. The minimal amount of time to generate a cluster is represented by t_{min} whereas tol_{max} is the maximal tolerance to evaluate the variation of the direction (ibid.). The algorithm calculates the direction variation between every two points. In case the direction does not variate between two points, the maximal tolerance is checked in order to identify whether the point was noise or if the direction change has ended (ibid.). The points with enough direction variation are added to the cluster if they pass the minimal time duration constraint (ibid.).
3	**Clustering-Based Stops and Moves of Trajectories (CB-SMoT)** *T, t_{min}, area, set of geometries* The CB-SMoT algorithm is a clustering method that is based on the speed variation of the trajectory. First, the trajectory sample points are evaluated and then clustered in places where the trajectory speed is lower than a given speed threshold for a minimal amount of time (t_{min}) (Rocha *et al.*, 2010). However, the pairwise distance between two sample points should not be greater than a minimum distance (Bermingham and Lee, 2018). This minimum distance is a relative parameter calculated from the mean and standard deviation of distances between adjacent points in a trajectory (Palma *et al.*, 2008). As an input, a value between 0 and 1 has to be set in order to compute the minimum distance parameter. This parameter (*area*) is the approximate proportion of points that generate potential stops in relation to the total amount of points (ibid.). Secondly, the clusters are matched with a set of relevant geographic places defined by the user. This algorithm is useful for applications, such as traffic management, where speed plays a major role (Rocha *et al.*, 2010).

4	**SpatioTemporal Clustering-based Stops and Moves of Trajectories (STC-SMoT)** *T, SC,* t_{min} The STC-SMoT detects spatiotemporal clusters regardless of the density (Hwang *et al.*, 2018). It checks for the minimum stop duration (t_{min}) between spatiotemporal neighbours (ibid.). The set of points need to be temporally close and in a specific distance (*Eps*). Most of the cluster-based approaches select *Eps* independently of the recording interval of a GPS trajectory (ibid.). However, *Eps* should be small when the data is collected in small intervals (high frequency), and *Eps* should be large otherwise (i.e., large interval/low frequency). To solve this problem, *Eps* is calculated out of the product between *SC* and the interval, whereas *SC* is the constant that takes the relationship between interval and *Eps* (ibid.). In order to remove noise, the noise point is replaced by the most frequent value via a sliding window containing the three consecutive data points (ibid.). The STC-SMoT works well with sparse trajectory data.
5	**Stops and Moves of Trajectories+ (SMoT+)** *T, IS, H* SMoT+ can determine the relationships between hierarchies. If site A contains site B and point C is located at B, the algorithm recognises that C also has to be located at A (Moreno *et al.*, 2014). The aforementioned SMoT algorithm does not take into account possible overlapping or nesting of geographic locations (ibid.). Using hierarchies, stops in nested regions would better express the semantics of the trajectories. Therefore, the SMoT+ algorithm should find stops in nested regions, such as regions that occur in other regions (ibid.). Finally, the total time of the stop per site is calculated. As input the algorithm uses a knowledge base that represents a hierarchy (*H*) of containment relationships between every interesting site from the set of interesting sites (*IS*) (ibid.).
6	**Density-Based Spatial Clustering of Applications with Noise (DBSCAN)** *T, Eps,* p_{min} With the DBSCAN algorithm data objects are clustered based on density (Karami and Johansson, 2014). It builds clusters, meaning groups the points into regions in which the objects are densely distributed (ibid.). These dense regions are separated by regions where the object density is low. The algorithm is able to discriminate noise as well as to find clusters with arbitrary shape (ibid.). In the case of stop-move detection, the noise points are considered as moves (Gong *et al.*, 2015). Besides the trajectory sample points (*T*), DBSCAN only requires two input parameters, the radius of the cluster (*Eps*) and the minimum number of points required in the cluster (p_{min}) (Karami and Johansson, 2014). If points have more neighbours than p_{min}, they are considered to be a core point of the cluster (Schubert *et al.*, 2017). However, it is not easy to determine suitable values for p_{min} and *Eps* (Karami and Johansson, 2014).

7	**Trajectory Density-Based Spatial Clustering of Applications with Noise (TrajDBSCAN)** *T, Eps,* t_{min} The TrajDBSCAN algorithm forms hierarchies of stop types: generic stops at large regions like public places, concrete stops at specific locations, personalised stops where only an individual stops, and shared stops where many individuals stop (Bermingham, 2018). The main goal of TrajDBSCAN is to find clusters in which all points inside are smaller than a given radius (*Eps*) and the duration staying at this place is greater than t_{min} (Tran *et al.*, 2011). In contrast to the DBSCAN algorithm, TrajDBSCAN defines a temporal linear neighbourhood with a core point that is determined based on a minimal stop time (t_{min}) instead of using a minimal number of points (p_{min}) (Gong *et al.*, 2015).
8	**Constrained Density-Based Spatial Clustering of Applications with Noise (C-DBSCAN)** *T, Eps,* p_{min}, DCC_{AP}, PCT_{AP} DBSCAN builds clusters without considering the temporal sequence of the points (Gong *et al.*, 2015). Consequently, the stop cluster could contain moving points or stops that happened to a later point in time. Furthermore, when the data is collected in small intervals, the DBSCAN algorithm fails to detect objects that are moving with a small speed (ibid.). In order to avoid the two errors, the C-DBSCAN algorithm fulfils two additional constraints. First of all, all points in a cluster should succeed one after another. In case of a sudden increase, the cluster will be divided into two different clusters (ibid.). If the p_{min} condition for each cluster is not fulfilled, the points will be labelled as moving. Otherwise, the potential stop cluster will be tested further. The percentage of abnormal points in a cluster (PCT_{AP}) should not exceed a user-defined threshold (ibid.). Additionally, points in a stop cluster should have an even distribution of direction changes. Therefore, the direction change coefficient (*DCC*) should nearly always have a different value from 1. Points with a *DCC* value close to 1 can be assumed to represent movement (ibid.). These points are called abnormal points. DCC_{AP} is used to imply the approximation to 1 (ibid.). If both constraints are fulfilled, the clusters are marked as stops (ibid.).

9	**Pre-Processing and post-processing techniques that are used in combination with an unmodified version of DBSCAN (P-DBSCAN)** *T, Eps, p_{min}, speed threshold, G* The actual stop-move detection algorithm of this approach is based on the simple DBSCAN algorithm (Bermingham, 2018). Pre- and post-processing methods should refine the algorithm's outcome. The pre-processing methods include the removal of outliers based on too high-speed entries and the interpolation between recording gaps within the trajectory (ibid.). For post-processing, the values are sampled from an odd-numbered fixed-size sliding window with grid cell size (*G*) and the entry labels within the window are homogenized based on the majority label (ibid.).
10	**SeqScan** *T, Eps, p_{min}, t_{min}* SeqScan tries to solve the problem of noise in cluster-based methods (Damiani and Hachem, 2017). It works well for trajectories of low-sampling rate GPS points and works based on the three input parameters distance (*Eps*), minimum points (p_{min}), and minimal time span (t_{min}) (Damiani, Issa and Cagnacci, 2014). The algorithm goes through three phases. First of all, it searches for minimal stay regions. In the search phase DBSCAN is used for clustering the data (ibid.). Then, the stay regions are expanded and if a new minimal stay region is found, the active stay region is finally closed. A stay region cannot be expanded anymore once it is closed (ibid.).
11	**Trajectory Clustering (TraClus)** *T, L, l_{min}, Eps* The TraClus algorithm consists of two phases 1) partitioning and 2) grouping (Lee, Han and Whang, 2007). For the first phase, the partitioning phase, a formal trajectory partitioning algorithm based on the minimum description length principle is developed (ibid.). Therefore, the first phase of the algorithm only needs the trajectory sample points (*T*) as input. The second phase, the grouping phase, is done by a density-based line-segment clustering algorithm. This algorithm is based on DBSCAN, but using line segments (*L*) instead of points (ibid.).

12	**Ordering Points to Identify the Clustering Structure (OPTICS)**	*T, Eps,* p_{min}
	In contrast to DBSCAN, OPTICS can handle clusters with different densities (Guidotti, Trasarti and Nanni, 2015). After ordering the points with respect to the distance function (i.e., spatially adjacent points follow each other closely), a value for each point can be determined (ibid.). This is the density value that a point needs in order to be accepted in a cluster. If these distances are plotted, the clusters form valleys and can thus be identified (ibid.). OPTICS orders the points by starting at any unprocessed point, determining the neighbours in the distance (*Eps*) and remembering them in a priority queue according to their best reachability distance so far (Ankerst *et al.*, 1999). The point with the smallest reachability distance is always next in the order. By processing a new point, the reachability distances of the unprocessed points can improve (ibid.). OPTICS processes a detected cluster completely by sorting this priority queue before continuing with the next cluster (ibid). Further, the user has to define the minimum number of points (p_{min}) in the *Eps*-environment to determine the centre point of a cluster.	
13	**StopFinder**	*T*
	The StopFinder algorithm by Zimmermann, Kirste and Spiliopoulou (2009) is based on OPTICS. Their algorithm takes into account the speed of motion and the temporal neighbourhood of the data points based on elapsed time. Additionally, the approach includes a visualization utility that portrays both the motion and the duration of a stop (Zimmermann, Kirste and Spiliopoulou, 2009). The spatial parameter (*Eps*) and the temporal parameter (t_{min}) are estimated by the algorithm for specifying the spatial neighbourhoods. The values are approximated by averaging the respective values over the number of data points (ibid.).	
14	**Simple stop detection algorithm**	*T,* d_{max}, t_{min}
	The Simple stop detection algorithm searches for sub-trajectories where the distance between two points does not exceed the user-defined spatial threshold (d_{max}) and does not fall below a user-specified temporal threshold (t_{min}) (Bermingham, 2018). This requires a very precise parameter setting by the user. The points that fulfil these two constraints are clustered as stops; the others labelled as moves.	

15	**Novel kernel-based algorithm**	*T, Eps, t_{min}*
	The algorithm proposed by Thierry, Chaix and Kestens (2013) uses the GPS points to build a kernel density surface instead of analysing the points sequentially. Local maxima are extracted from the kernel density surface and classified as periods of stay, meaning stops (Thierry, Chaix and Kestens, 2013). The algorithm requires one spatial and one temporal parameter. With the spatial parameter (*Eps*) the user defines the kernel bandwidth and with the temporal parameter (t_{min}) the user defines the minimum duration of a stop (ibid.). According to the authors, the smoothing effect of the kernel density estimations should reduce the susceptibility to noise.	
16	**Sequence Oriented Clustering (SOC)**	*T, Eps, Tau, t_{min}, t_{min}*
	The SOC algorithm allows to identify noise points in a stop sequence and to classify them as part of the stop. The algorithm computes a reachability distance for each point by scanning the trajectory in chronological order (Xiang, Gao and Wu, 2016). The reachability distance is defined by a spatial parameter (*Eps*) and a temporal parameter (*Tau*) (ibid.). The algorithm then extracts continuous points whose reachability distance does not exceed a given threshold (*Eps*). Stops are then computed from *Eps*-reachability sequences (ibid.). Furthermore, a minimum duration (t_{min}) for a stop and a move, respectively has to be defined by the user. During the process, sequences which are separated by noise points but spatially and temporally close are merged (ibid.). Finally, false positive stops are removed from the results (ibid.).	
17	**TwO-Steps parameter free Clustering Algorithm (TOSCA)**	*T, cut-threshold*
	TOSCA takes small clusters obtained from centre-based methods and tries to aggregate them via an iterative procedure like single linkage hierarchical clustering based on their medoids (Guidotti, Trasarti and Nanni, 2015). The aggregation is stopped when the effort of merging two clusters is much larger than the previous merges. However, determining the cut-threshold, meaning the precise moment, is a non-trivial issue (ibid.).	

18	**CandidateStops**	*T*, v_{min}
	The CandidateStops algorithm is a statistical method that works by considering the sampling rate standard z-score (Nogueira, Braga and Martin, 2014). The metric indicates how far the sample is from the mean of the whole dataset. With this value, the candidate stops are identified based on the analysis of sampling rate and speed values (ibid.). Good position accuracy has to be assumed when applying this algorithm. For each point, the time difference between the current and previous point is considered. When the standard score of each element of different time is calculated, moments where the interval of time between the points lasts longer than normal can be determined (ibid.). Because a long-time gap could happen due to GPS signal loss, the speed (v_{min}) has to be considered as well. A decreasing speed can be assumed in case of a stop, whereas the speed stays the same during a GPS signal loss (ibid.).	
19	**Stop detection algorithm by Montoliu, Blom and Gatica-Perez (MBGP)**	*T*, t_{min}, t_{max}, d_{max}
	The MBGP algorithm groups the points of the GPS trajectory using a time-based clustering technique. The parameter t_{min} represents the minimum duration between the points in order to count as a stop (Montoliu, Blom and Gatica-Perez, 2013). Further, the distance between the first GPS point of a stop cluster and the last GPS point of this cluster is measured. The diameter of this stop cluster must not be greater than a maximum distance (d_{max}) (ibid.). Finally, t_{max} represents the maximum allowed time gap between two recorded location points to be considered as a part of the same stop cluster.	
20	**Point-of-Interest Extraction (PIE)**	*T*, i_{max}, *Eps*, t_{min}, dir_{min}, π_{max}, *set of geometries*, P_{POLOI}
	According to De Graaff, De By and De Keulen (2016) the PIE algorithm detects stop sequences, that are geographical regions where a user stayed for a minimal time (t_{min}) within a distance of *Eps*. The algorithm filters out the points that are too imprecise to do further calculations with (i.e., points where the inaccuracy value exceeds the accuracy threshold (i_{max})). Then, the stay points (i.e., centroids of stop sequences) are extracted and the direction change between the stay points is determined (De Graaff, De By and De Keulen, 2016). For that, the user has to define a minimum value for direction change (dir_{min}) (ibid.). With this it can be determined whether any stay point is really a visited spot or caused due to natural behaviour (e.g., traffic jam). Afterwards, the selected points are projected onto nearby polygons (a set of all existing polygons is also an input of the algorithm) by considering the maximum projection distance (π_{max}). Polygons-of-interest (POLOIs) are added to the result set of visited polygons. Due to the fact that the heading change is taken into account, the stay points that were on a relatively straight path with respect to the previous and next stop points are filtered out (ibid.).	

21	**Move-Stop-Noise (MSN)**	$T_{distance,\ duration,\ speed,\ and\ turning\ angle}$, v, dir_{min}, Eps, t_{max}, j
	The MSN algorithm is a statistical method for stop, move, and noise detection and requires the distance, duration, speed, and turning angle between each pair of points as input parameters (Nogueira, Martin and Andrade, 2017). The turning angle can be described as the angle formed by three neighbouring points. Further input parameters are the modified z-score limits for distance (*Eps*), duration (t_{min}), and speed (v) (ibid.). To improve noise detection, a minimum rotation angle (dir_{min}) can be set, since it is assumed that a moving object does not rotate at small angles (ibid.). To make the algorithm more robust against outliers, a random uniform jitter (j) can be used (ibid.). First, the MSN algorithm identifies potential noise points, meaning points with relatively long distances compared to the median distance of all pairs of sequential trajectory points (ibid.). Furthermore, the turning angles are verified in order to filter out the noise points. After identifying and removing potential noise, potential stops are labelled (ibid.).	
22	**Time and Distance Based Clustering (TDBC)**	T, Eps, t_{max}
	The TDBC algorithm counters the problem that of most of the present algorithms have (i.e., some shortcomings due to the influence of GPS signal loss and data drift) (Fu *et al.*, 2016). According to the authors, the algorithm has a better adaption for individual trajectory when determining stops. As input parameters TDBC uses a time threshold (t_{max}) and a distance (*Eps*) (ibid.). Fu *et al.* (2016) distinguish between three types of points as already mentioned and visualised in *Figure 2.3*. The algorithm itself performs a case distinction to find the three types of points. First of all, the algorithm checks for stops of the first type. If they belong to type one, a stop is directly classified (ibid.). Afterwards, the distance from a cluster to the next point is determined. If the distance is smaller than the threshold, the point is added to the cluster (ibid.). As soon as the distance and time exceed the given thresholds, the cluster is closed and defined as stop of the second type. In case the distance is smaller and the time duration greater than the given thresholds, there occurred a gap in the trajectory and a stop of type three is identified (ibid.). The remaining points are classified as moves.	

23	**Probability of Stops and Moves in Trajectories (POSMIT)**	*T*, h_i, h_d, *e*
	The POSMIT algorithm calculates the probability that a point is actually stopping (Bermingham, 2018). Since each point is assigned with a stop probability, the user must specify a minimum stop probability (*e*). All points that are greater than or equal to this stop probability are classified as stops (ibid.). This allows the user to filter out ambiguous points. This way, the probability of error during classification can be kept to a minimum (ibid.). POSMIT does not define stops over a minimum duration but weights the number of indices between the current entry and any neighbouring entry for the calculation of the stop probability (the less the weighting, the larger the number of indices) (ibid.). This can be realised with the search bandwidth (h_i). With the stop variance (h_d) users can define how strict they want to be when calculating the stop probability (ibid.). Due to the noise in the GPS data and its system-induced inaccuracy, a certain spatial shift occurs between the points. Therefore, the parameter h_d has a considerable impact on the size of the stop cluster (ibid.). If h_d is chosen close to zero, this means that only points with the same geographical position would be assigned to the stop cluster (ibid.).	
24	**Velocity-based trajectory structure**	*T*
	According to Yan *et al.* (2010) the velocity-based trajectory structure algorithm first smooths the trajectories and reduces the outliers by velocity thresholds that are determined due to domain knowledge (i.e., speed of walking, driving bike, driving car). The speed threshold used for finding stops, is determined through an algorithm (Nogueira, Martin and Andrade, 2017). This speed threshold is used in combination with a function that considers the moving object's average speed and the average speed of other moving objects. For this calculation the space is divided into a grid and each cell gets an average speed value (Yan *et al.*, 2010). Because the algorithm requires non-robust average speed measures, a correct identification of stops could be difficult in case there is a large speed range in a trajectory. In order to reduce the number of input parameters, there was an effort to set speed thresholds dynamically, however the stop duration is still a user defined absolute metric value (ibid.).	

25	**Grid-growing clustering algorithm**	*T, G*
	The grid-growing clustering algorithm performs clustering operations on the cells of the partitioned data space (Zhao *et al.*, 2015). As the data are looked at in smaller packages, the algorithm is much more time efficient than general clustering algorithms (ibid). In the first part of the grid-growing algorithm, a grid structure (*G*) containing *n* rows and *n* columns is generated. Each data point is assigned to the corresponding grid cell based on its location. Afterwards, a region growing on the grid structure is performed (ibid.). This results in a certain number of groups. After selecting *n* number of seeds, regions are grown to adjacent points and *K* number of clusters are formed. From the *K* number of clusters, the clustering partitions (*P)* are gained (ibid.). Points that are do not correspond to a cluster are registered as noise, and in the case of stop-move detection, classified as moves.	

Parameter legend:
area = approximate proportion of points that generate potential stops in relation to the total amount of points,
DCC_{AP} = direction change coefficient of abnormal points,
dir_{min} = minimal direction change,
Eps = geographical distance,
H = Hierarchy of containments among the sites from *IS*,
h_i = index search bandwidth,
j = random uniform jitter,
l_{min} = minimal lines,
p_{min} = minimal number of points,
SC = constant that takes the relationship between Interval and *Eps*,
Tau = time duration that defines a core sequence,
t_{min} = minimal amount of time,
v = speed,
d_{max} = maximal distance,
e = minimum stop probability,
G = grid cell size,
h_d = stop variance,
IS = Set of interesting sites,
L = set of line segments,
PCT_{AP} = percentage of abnormal points,
P_{POLOI} = subset of set of geometries containing POLOIs,
T = trajectory sample points,
t_{max} = maximal amount of time,
tol_{max} = maximal tolerance,
v_{min} = minimal speed

An analysis and a table that show the basis on which criteria the algorithms used for this thesis were selected can be found in *Section 4.1*.

2.3.3 Algorithm Selection Criteria

While working with algorithms, it is important to understand the basic concepts of algorithm design and modelling. The first step of designing algorithms is to get a profound knowledge about the problem before planning a solution (Kant, 1985). Finding appropriate criteria that an algorithm should meet is not trivial. Therefore, there are several approaches of criteria selection. One example is that conditions for being a suitable algorithm can be condensed at a higher conceptual level as 1) maximum parsimony of the algorithmic model, 2) ease of understanding, and 3) high performance.

At the first level, the principle of parsimony shows that an increasing of the number of parameters in a model leads to a decreasing bias but an increasing variance (Posada and Buckley, 2004). Examples for high parsimony are: minimal number of input parameters, kind of input parameters, minimal number of input datasets, and mathematical simplicity.

At the second level, it is important that the algorithms are easy to understand. This means that they are conceptually intuitive and easy to visualize in mind. According to Xiao (2016), algorithms that use fewer input parameters are easier to explain and understand.

The third level of a suitable algorithm covers the aspect of high performance. High performance and efficient algorithms are especially important when a considerable number of input datasets has to be handled (Wegener, 1989). Aspects that cover the criteria of high performance are, for example, accuracy, precision, small computational complexity in time and space (e.g., minimizing energy used), robustness to noise and outliers, and input data resolution (Ovaska and Sztandera, 2002; Xiao, 2016; Dongarra, Grigori and Higham, 2020). However, the efficiency or speed of an algorithm is always determined by the problem to solve too (Xiao, 2016). Xiao (2016) suggests getting an overview of the computational steps that are needed to run the algorithm, as external factors such as computer memory could influence the running time.

2.4 Algorithm Evaluation

For algorithm evaluation, evaluation metrics are often employed (Fedorchuk and Lamiroy, 2017). Besides the more general approaches, such as comparison of the algorithm's running time or output evaluation (i.e., in case of stop-move detection algorithms the average number or variance of detected stops), algorithms can be evaluated with probabilistic metrics, through sensitivity analysis, or shape measures (MacEachren, 1985; Silberholz and Golden, 2010; Fedorchuk and Lamiroy, 2017; Bermingham, 2018).

2.4.1 Probabilistic Metrics

Collecting ground truth and using it for evaluating or further training methods, is very convenient in current research. However, it is often rather costly to obtain and sometimes impossible to get (Lamiroy and Sun, 2013). Generally, ground truth collection requires human intervention and needs to be validated somehow. Hence, the ground-truthing process is costly, error prone, and difficult to scale (ibid.).

Algorithms with binary outputs (i.e., stop or move of the stop-move detection algorithms) can be evaluated without ground truth as proposed by Fedorchuk and Lamiroy (2017). The main idea of this approach is that the standard ground truth will be replaced with a probability-based formula (Fedorchuk and Lamiroy, 2017). The statistical equivalent of the ground truth is an array expressing the probability $P(\delta_i)$ of class affiliation $\triangle^+$ for each data item δ_i (see *Equation 2.5*) (ibid.).

$$P(\delta_i) = \sum_{k=1..s} \frac{S_k(\delta_i)}{s} \tag{2.5}$$

$S_n(\delta_i) \in \{0,1\}$ signifies the classification result of data item δ_i by classifier S_n and s the number of classifiers (ibid.).

Fedorchuk and Lamiroy (2017) describe three ways of how classifiers can be compared and evaluated using probabilistic metrics: F-measure, Negative Rate Metric, and Normalized Cross Correlation.

2.4.1.1 F-measure

In order to get the correspondent probabilistic equivalent of the F-measure, probabilistic equivalents of precision (*Pr*) (i.e., amount of relevant selected items, *Equation 2.6*) and recall (*Rc*) (i.e., amount of selected relevant items, *Equation 2.7*) are calculated and then their harmonic mean is computed (Fedorchuk and Lamiroy, 2017). The higher the value of this metric, the better the classifier.

$$Pr(S_k) = \frac{\sum_{1..d} P(\delta_i) S_k(\delta_i)}{\sum_{1..d} S_k(\delta_i)} \qquad (2.6)$$

$$Rc(S_k) = \frac{\sum_{1..d} P(\delta_i) S_k(\delta_i)}{\sum_{1..d} P(\delta_i)} \qquad (2.7)$$

In case there is ground truth available, the F-measure, precision, and recall can also be determined with regard to true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) (Lamiroy and Sun, 2013).

The following confusion matrix in *Figure 2.6* visualises the context of true and false positives, and false negatives.

Ground Truth \ Algorithm classification	✓ (stop)	x (no stop)
✓ (stop)	Stop detected by algorithm and present in ground truth (TP)	Stop not detected by algorithm but present in ground truth (FN)
x (no stop)	Stop detected by algorithm but not present in ground truth (FP)	No stop detected by the algorithm and not present in ground truth (TN)

Figure 2.6 Exemplary confusion matrix (modified from Fillekes, Kim, *et al.*, 2019:23)

Out of this confusion matrix, the following equations for precision (*Equation 2.8*), recall (*Equation 2.9*), and F-measure (*Equation 2.10*), can be determined (Fillekes, Kim, *et al.*, 2019):

$$Pr = \frac{TP}{TP + FP} \qquad (2.8)$$

$$Rc = \frac{TP}{TP + FN} \qquad (2.9)$$

$$F = \frac{2(Pr * Rc)}{Pr + Rc} \qquad (2.10)$$

2.4.1.2 Negative Rate Metric

The Negative Rate Metric is a function that is based on the probabilistic equivalents of false negative, false positive, true positive and true negative values. Out of the negative rate of false positive (NR_{FP}) and false negative values (NR_{FN}), the average (i.e., Negative Rate Metric) is calculated and visualized in *Equations 2.11* and *Equation 2.12* (Fedorchuk and Lamiroy, 2017). The lower the value of this metric, the better the classifier.

$$NR_{FP}(S_k) = \frac{\sum_{1..d}(1 - P(\delta_i))S_k(\delta_i)}{\sum_{1..d} P(\delta_i)} \qquad (2.11) \qquad NR_{FN}(S_k) = 1 - \frac{\sum_{1..d} P(\delta_i)S_k(\delta_i)}{\sum_{1..d} P(\delta_i)} \qquad (2.12)$$

2.4.1.3 Normalized Cross Correlation

The Normalized Cross Correlation (see *Equation 2.13*) is the normalized correlation that is calculated between the probability that the data items δ_i (given the majority voting $P(\delta)$) belong to class $\Delta+$ and the result given by classifier S_k (Fedorchuk and Lamiroy, 2017). The higher the value of this metric, the better the correlation between the two arrays.

$$\mathrm{NCC} = 1 - \frac{\sum_{1..d}(S_k(\delta_i) - \overline{S_k})(P(\delta_i) - \overline{P_\delta})}{\sqrt{\sum_{1..d}(S_k(\delta_i) - \overline{S_k})^2 \sum_{1..d}(P(\delta_i) - \overline{P_\delta})^2}} \qquad (2.13)$$

2.4.1.4 Matthew's Correlation Coefficient

Bermingham (2018) introduces the Matthew's Correlation Coefficient for measuring the binary classification accuracy (see *Equation 2.14*). Opposite to precision and recall, the Coefficient represents the accuracy in a single value. It also considers all quadrants of the confusion matrix covering the true and false observations contrary to the F-measures (Bermingham, 2018).

$$\mathrm{MCC} = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \qquad (2.14)$$

2.4.2 Sensitivity Analysis

To measure the sensitivity of algorithms, Bermingham and Lee (2018) proposed evaluating the algorithms according to sampling rate robustness and general classification effectiveness. The sampling rate robustness can be measured by varying the sampling rate of the input data 1) regularly or 2) randomly. Regular sampling rate means that data points are removed from a single trajectory with equal intervals (i.e., considering every third, every fourth, … , every *n*-th data point) (Bermingham, 2018). The sampling rate can be selected randomly by using Poisson sampling (Ohlsson, 1998). The approach assumes that the Poisson distribution describes the random mechanism to generate the sampling data (ibid.).

The classification effectiveness can be compared by varying the different input parameters while the others stay the same (Bermingham, 2018). The effect of each parameter on the result can be investigated.

Furthermore, the robustness to noise (i.e., creating degraded data with added noise) is also a common approach to test the sensitivity of an algorithm (Newson and Krumm, 2009). In order to test the robustness to noise there are approaches that increase the proportion of noise in a dataset from 0 – 90 % and analyse how the algorithms perform differently (Domingues *et al.*, 2018). One approach for simulating GPS noise could be to add Gaussian noise to the data using a distribution with mean value of 0 (Bösche *et al.*, 2013).

A second approach is the house noise model introduced in Zhang, Liu and Song (2015). This function has a shape of a pitched house roof and noise can be applied to the data at a level between 0 and 1. A minimum level of 0 means that no noise is applied whereas a level of 1 means that the original sample will be changed to other values having a probability of 1 (ibid.).

2.4.3 Shape Measures

Algorithms' stops can also be evaluated through shape measures. Li, Goodchild and Church (2013) proposed an evaluation method that measures the stops' shape compactness, which is a numerical quantity symbolizing the degree to which a shape is compact. Compactness can be seen as an indicator of homogeneity within units (MacEachren, 1985). Li, Goodchild and Church's (2013) approach is a trapezium-based one to compute compactness based on the concept of the moment of inertia (*MI*) (i.e., the second moment of an area about a point (*p*) on the shape).

With this approach the compactness value (MI shape index (C_{MI})) can be determined and compared (Li, Goodchild and Church, 2013). The MI shape index is the circles' *MI* about its centre to the shape's *MI* about its centroid as *Equation 2.15* shows. Thereby, the values range between 0 and 1 whereas larger values indicate more compact shapes (ibid.). In the equation below, the area of the shape is *A*.

$$C_{MI} = \frac{A^2}{2\pi I_g} \tag{2.15}$$

Equation 2.16 for I_g which is needed for completing equation C_{MI} is shown below.

$$I_g = \sum_{i=1}^{n} I_{i\_T} + d_{i\_T,g}^2 A_{i\_T} + I_{i\_R} + d_{i\_R,g}^2 A_{i\_R} \tag{2.16}$$

In this equation, $d_{i\_T,g}$ and $d_{i\_R,g}$ are the distances between the centroids of the rectangle and triangle pieces of trapezium (*i*) (ibid.). The equation basically calculates the area, centroid, and *MI* of the whole polygon (ibid.).

Ebdon (1985)'s compactness measures that count as rather robust measures compare the shape of a polygon with the shape of a circle as a circle counts as the most compact shape (see *Equation 2.17*) (in Kitchin and Tate, 2000). The closer to 1 the shape index value, the more compact the shape.

$$f_{comp} = \frac{4A}{\pi D^2} \tag{2.17}$$

Equation 2.17 uses the smallest enclosing circle's diameter (*D*) that is calculated by finding the longest axis across the polygon (Kitchin and Tate, 2000). In contrast to the following compactness measure, the longest axis across the polygon is allowed to leave the polygon.

The next compactness measure proposed by Ebdon (1985) includes the longest axis of the shape without leaving the polygon (*L*) as *Equation 2.18* shows (in Kitchin and Tate, 2000).

$$f_{comp} = \frac{4A}{\pi L^2} \tag{2.18}$$

The last measure divides the radius of the smallest enclosing circle (R_1) by the radius of the circle with the same areas the shape (R_2) (see *Equation 2.19*) (Ebdon, 1985 in Kitchin and Tate, 2000).

$$f_{comp} = \frac{\mathrm{R}_1}{\mathrm{R}_2} \qquad where \qquad \mathrm{R}_1 = \sqrt{\frac{A}{\pi}} \qquad and \qquad \mathrm{R}_2 = 2D \tag{2.19}$$

3 Data

3.1 MOASIS Data

This thesis works with the GPS and Inertial Measurement Unit (IMU) datasets collected for the MOASIS project in 2018. There are data available from 152 participants (four of which are omitted for further evaluation due to broken devices or lack of data) that were collected during a 30-day study period. As of the time of study, the participants' age was over the retirement age and they were living in German-speaking Switzerland. The device that was used to collect the data is called *uTrail* and equipped with three sensors (see *Figure 3.1*). The GPS, IMU, and Electronically Activated Recorder (EAR) sensors cover the aspects of healthy aging: spatial and physical activity, and social interaction (Bereuter, Fillekes and Weibel, 2016).

	Sensor	Variable	Sampling rate
Spatial mobility	GPS	timestamp, latitude, longitude	1 / sec
Physical activity	IMU	timestamp, acceleration (x,y,z)	3 / sec
Social interaction	EAR	timestamp, sound sample	1 / 12.5 min

Figure 3.1 uTrail mobile Sensor used for data collection in the MOASIS study (modified from Bereuter, Fillekes and Weibel, 2016:2)

The EAR sensor data is not relevant for this thesis. The stops and moves were determined with the data collected from the GPS sensor that collected the GPS data points with a sampling rate of 1 point per second (ibid.). On average over 600’000 GPS data points were collected per participant.

The IMU sensor is a 3-axis accelerometer and its data were used in combination with the GPS data for manual ground truth labelling (see *Section 3.2* for further information).

After the first two weeks of data collection, the participants had to change their *uTrail* so that the data were able to be downloaded from the device and that no whole participants got lost for study purposes in case a *uTrail* broke. In that unfortune case, at least half the data of the study period could have been secured through this intermediate session. Hence, for the GPS and for the IMU data there exist two .csv-files per participant that contain the first half and the second half of the study period’s data, respectively.

In each GPS file, each column contains a timestamp (date and time), geographic coordinates (i.e., longitude and latitude), the number of satellites, an altitude, Vertical Dilution of Precision (VDOP), Horizontal Dilution of Precision (HDOP), recording and speed. Relevant for this thesis are the columns timestamp, geographic coordinates (i.e., longitude and latitude), and speed.

Each IMU file has the following columns: timestamp (date and time), the acceleration in mg (for each direction X, Y, Z), and the Magnetic flux density in mGauss (for each direction X, Y, Z). Relevant for ground truth labelling are the columns timestamp and acceleration in mg (for each direction X, Y, Z).

3.2 Ground Truth Labelling

In order to find the most suitable threshold combination per algorithm (one before and another after post-processing[3]), some ground truth had to be labelled manually using an R-Shiny Application implemented by Oliver Burkhard in 2017[4] as no actual ground truth information is available. Without labelling ground truth manually, the algorithms' outcomes could not be validated. The Shiny Application needs one .csv-file containing IMU and one .csv-file containing GPS information as input. It then visualises the trajectory and the user can click through the trajectory and gets information on the acceleration and speed of each point in the trajectory. With this information and visual monitoring, stops can be determined, labelled and downloaded (see *Figure 3.2* as an example for the Shiny Application's Graphical User Interface. The red arrows indicate that data are visualised for a user specified timeframe).

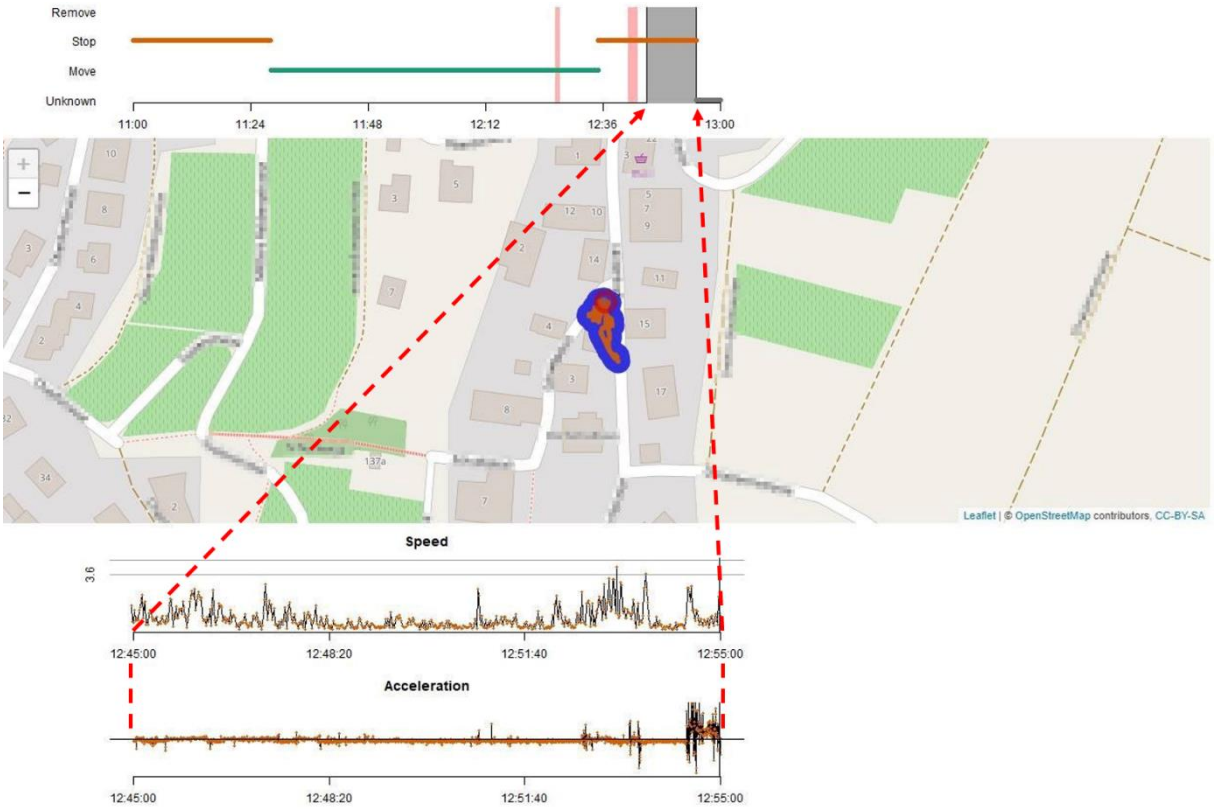


Figure 3.2 Graphical User Interface of Shiny Application by Burkhard (2017)

[3] In the further course of this thesis, the best threshold combinations for the results with and without post-processing are meant when talking about the optimal threshold values or similar.
[4] Check out the following link for detailed information on the R-Shiny Application: https://github.com/o1i/mode_tagger

The shiny application works with the three attributes speed, acceleration and time. Based on *Chapter 2*, data points were considered as stops that met the following criteria:

- velocity less than 1 m/s (Nogueira, Braga and Martin, 2014)
- acceleration between -0.3 m/s^2 and 0.3 m/s^2 (Nogueira, Braga and Martin, 2014)[5]
- stop duration at least 10 minutes (see *Section 4.3.1.2* for comparison)

In order to be able to label ground truth manually, sample datasets have to be selected.

3.3 Sample Data

The aim is to obtain datasets with as many data points as possible since there is a greater chance for detecting obvious stops while selecting ground truth manually. Datasets with many data points are further important as the sampling rate is going to be modified in order to test the sampling rate robustness. Furthermore, many data points could indicate that people were more likely to leave their house. Therefore, all data frames dated with Sundays are selected and the thirty data frames with the highest number of data points are chosen as sample datasets as people tend to cover highest daily distances on Sundays (Swiss Confederation BFS, 2012). Covering highest daily distances on Sundays means additionally, that people perform most of their outdoor leisure activities on these day (ibid.). Therefore, the chance of finding obvious stops while labelling ground truth manually increases. In order to prevent bias and covering some weekdays as well, thirty data frames with the highest number of data points of Tuesday and Thursday, respectively, are selected. With this approach, the diversity of sample datasets is given since some participants tend to take similar routes each day. In other words, while covering weekdays as well as weekends and including various participants' data by choosing the thirty datasets with the most data points per day, the chance of getting similar routes decreases.

[5] As it can be assumed, that during a stop period whether a positive nor negative acceleration takes place, the definition of steady human motion mentioned in Nogueira, Braga and Martin (2014) with an acceleration value range of -0.3m/s^2 and 0.3m/s^2 was taken.

4 Methodology

The methodological approaches of the Master's thesis can be divided into three parts: 1) conceptual framework and algorithm selection, 2) algorithm implementation, and 3) comparison and evaluation of the algorithms (see *Figure 4.1*).

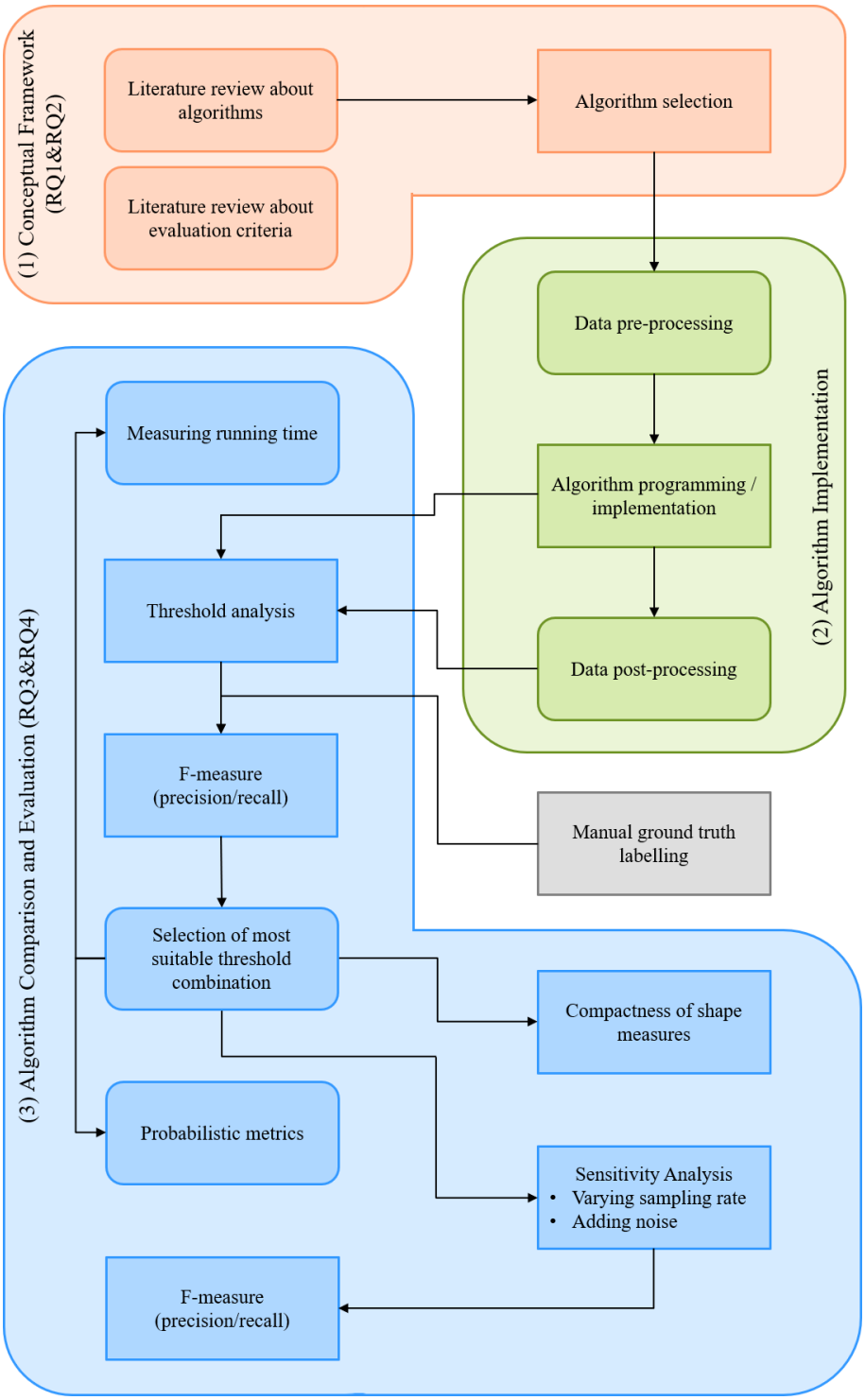


Figure 4.1 Overview of methodological approaches

The first and second research questions (RQ1 and RQ2) will be answered in the first methodological part and the third and fourth research questions (RQ3 and RQ4) in the last part. The second part focuses on working with the data and algorithms.

4.1 Conceptual Framework and Algorithm Selection

Based on *Chapter 2*, criteria could be worked out in order to select four algorithms that were going to be applied or implemented with the programming language R. *Table 4.1* provides an overview of the criteria the algorithms should ideally meet: 1) maximum parsimony of the algorithmic model, 2) ease of understanding, and 3) high performance.

1) Maximum parsimony

- The algorithms' number of input parameter should be minimal, but at least 3.
- The algorithms require at least a definition of a spatial and a temporal parameter (e.g., distance or minimum time, respectively).

2) Ease of understanding

- The algorithms' parameter setting should be at a non-expert level. This means that the user has an intuition about the meaning of the parameters, and no considerable mathematical background is necessary to understand how the parameters work.
- The algorithms should not use predefined geographic objects.

3) High performance

- The algorithms should be robust against outliers and noise.
- The algorithms should deliver reliable outcomes independently of the trajectory data sampling rate.
- The algorithms should not be computationally complex in time and space.
- The algorithms should detect indoor stops.

Table 4.1 Algorithm comparison based on criteria

	Criterion Category	(1) Maximum parsimony		(2) Ease of understanding		(3) High performance			
	Criterion / Algorithm	few, but at least 3 parameters	contains a spatial and temporal input	no expert-level parameter setting	no use of geographic objects	resilience to GPS noise	works well with sparse trajectory	not computationally complex	detects indoor stops
SMoT-Based	CB-SMoT	4	x						
	DB-SMoT	4	x	x	x				
	SMoT	3	x	x					
	SMoT+	3	x	x		x	x		
	STC-SMoT	3	x	x	x	x			
DBSCAN-Based	C-DBSCAN	5			x	x	x	x	
	DBSCAN	3			x	(x)		x	
	P-DBSCAN	5			x	(x)		x	
	SeqScan	4	x		x	x	x	x	
	TraClus	4			x	(x)		x	
	TrajDBSCAN	3	x		x	(x)	x	x	
OPTICS-Based	OPTICS	3			x	x	x	x	
	StopFinder	1		x	x	x		x	
Grid-Based	Grid-growing clustering algorithm	2			x			x	
	Velocity-based trajectory structure	1			x	x			
Others	CandidateStops	2		x	x				
	MSN	6	x		x	x	x		
	Novel kernel-based algorithm	3	x		x	x	x		
	PIE	8	x			x	x		x
	POSMIT	4	(x)	x	x	x	x	(x)	
	Simple stop detection algorithm	3	x		x			x	
	SOC	5	x		x	x	x	x	x
	MBGP	4	x	x	x			x	
	TDBC	3	x		x		x	x	
	TOSCA	2			x		x		

As already shown with the criteria listed above, the conditions for being a suitable algorithm can be condensed at a higher conceptual level as 1) maximum parsimony of the algorithmic model (e.g., minimal number of input parameters, containing a spatial and a temporal parameter), 2) ease of understanding (e.g., conceptually intuitive/easy to visualize in mind and no use of additional input variables such as geographic objects), and 3) high performance (e.g., computational complexity in time and space, robustness to noise, robustness to input data resolution, and detecting indoor stops).

At the first level, a maximum parsimony of the algorithmic model can be reached by minimising the number of input parameters. As the algorithms determine stops and moves in spatiotemporal trajectories, it would be good for comparing the algorithms on a spatiotemporal basis if the algorithms had a spatial and a temporal parameter (i.e., parameters that limit the stop duration and spatial extent or shape of potential stops). These two parameters combined with a directional parameter are commonly used in stop-move detection algorithms (Zadeh Monajjemi, 2013). The POSMIT algorithm does not have a specific temporal input parameter, such as minimum duration because it calculates the stop probability of each individual data point (Bermingham, 2018). POSMIT can therefore find very granular stops without taking a temporal parameter into account (i.e., stops containing sparse number of data points) (ibid.). However, Bermingham (2018) points out that because trajectories are recorded as time-ordered sequences of entries, the index search bandwidth (h_i) indirectly integrates the temporal dimension of the data as well. For this reason, POSMIT is marked as (x) in *Table 4.1* that it should contain a spatial and a temporal parameter.

Besides a spatial and a temporal parameter, the trajectories themselves (i.e., containing data points with latitude, longitude, date, time, and speed values) are a prerequisite input parameter so that the minimum number of input parameters should be at least three. It can be assumed that the more input parameters are included, the more detailed the knowledge for the input parameters algorithm has to be. This would contradict the principle of high parsimony because more input parameters would increase complexity (Posada and Buckley, 2004).

At the second level, the algorithm should also meet the criterion that it can be easily understood. This means that non-expert researchers can intuitively understand how the parameters function in the algorithm and set suitable parameters for their purpose of analysis. The parameters' influence on the selected stop should be mentally visualisable and no huge mathematical background should be necessary to understand how the parameters work. Karami and Johansson (2014) state that it is not easy to determine suitable values for p_{min} and *Eps* that are necessary for the DBSCAN-based algorithms and OPTICS. Algorithms' authors, Bermingham (2018) as an example, emphasize that their developed algorithm (in this case, POSMIT) is highly suitable for non-expert researchers because all of the parameters can either be estimated/calculated by the algorithm or set by the researcher.

The next criterion is that the algorithms should not require geographic objects as additional input data. Bermingham and Lee (2018) point out that a disadvantage of geographic-based algorithms is that using them is very time consuming: first, performing spatial queries throughout the algorithm to find the data points that enter the geographic geometries and second, selecting the specific candidate places, especially when they are labelled manually (Bermingham and Lee, 2018). Moreover, a spatial database that contains spatial geometries has to be available and stops that are not covered by the selected geometries will not be detected (ibid.).

At the third level, the algorithms should deliver a high performance. One of the most important criteria is that an algorithm is resilient (i.e., robust) against noise and outliers. When not explicitly stated, the corresponding column in *Table 4.1* was left empty. Some DBSCAN-based algorithms are marked as (x) in *Table 4.1* for that criterion. The DBSCAN algorithm and its extensions are developed to handle applications with noise; they are specialized for detecting clusters with irregular shapes. However, some DBSCAN-based algorithms such as SeqScan are further improved to handle noise in cluster-based methods (Damiani and Hachem, 2017). Hence, the improved algorithms were marked with the x in *Table 4.1*. SOC, for example, is able to accept noise points as part of a stop instead of marking them as noise, when detected in a trajectory stop (Xiang, Gao and Wu, 2016).

Furthermore, the algorithms should deliver consistent results independently of the number of data points. It is common that human trajectory data relying on location sensing mobile devices have missing parts in data, due to GPS signal loss or compliance issues (Fillekes, Kim, *et al.*, 2019). Although the MOASIS data have many data points available, a significant number of participant datasets show incomplete records and contain noise. However, Das and Winter (2016) state that most DBSCAN algorithms do not perform well with sparse trajectory data (e.g., Hwang *et al.*, 2018). Additionally, when clusters' density varies locally or different clusters have different densities, DBSCAN performs poorly at differentiating between them (Braune, Besecke and Kruse, 2015). As with the resilience to noise criterion, when not explicitly stated that the algorithm performs well with sparse trajectory data, the corresponding cells in *Table 4.1* were left empty.

By looking at the computational complexity, the SMoT algorithms are regarded rather complex and time consuming because they have to access an external database with geographic objects (Bermingham and Lee, 2018). According to Xiang, Gao and Wu (2016), the DBSCAN-based algorithms, the OPTICS-based algorithms, and the SOC algorithm have the same computational complexity that is $O(n*log\ n)$, where n is the number of data points. However, SOC requires less computation than DBSCAN and OPTICS because after detecting a core sequence, it only checks the further points belonging to that core sequence (Xiang, Gao and Wu, 2016). POSMIT was marked as (x) in *Table 4.1* for this criterion because depending on the parameter value h_i it outperforms the SMoT-based

algorithms; it was compared to by Bermingham (2018). For POSMIT, the computational complexity is $O(nh_i)$, and for CB-SMoT, it is $O(n)$ (ibid).

Since the thesis focuses on detecting stops and moves out of mobility patterns of older adults who are more sedentary than active, an algorithm's ability to detect indoor stops is critical (see *Section 2.1.2*, for further explanation). Only the PIE and SOC algorithm focus on detecting indoor stops. According to De Graaff, De By and De Keulen (2016), the PIE algorithm was developed in particular for research concerning urban indoor trajectory assessment (De Graaff, De By and De Keulen, 2016). Xiang, Gao and Wu (2016)'s algorithm is capable of detecting indoor stops. The authors compared their SOC algorithm to other algorithms, showing that SOC outperformed the other algorithms regarding indoor stop detection.

4.1.1 Algorithm Selection

Based on the number of criteria met out of the analysis illustrated and summarized in the previous section and *Table 4.1*, the following four algorithms were selected for further analysis: CandidateStops, MBGP, POSMIT, and SOC.

The evaluation results in *Table 4.1* show that the SOC and POSMIT algorithms meet most of the criteria (seven out of eight). The authors of both algorithms claim that the algorithms performed best compared to other stop-move detection algorithms such as CB-SMoT. According to Bermingham (2018), POSMIT showed the best overall classification effectiveness and was least sensitive to changes in the spatial parameter. Xiang, Gao and Wu (2016) pointed out that SOC was more robust to noise than the other algorithms. However, the performance of SOC and POSMIT was not tested against each other. Furthermore, two other algorithms were chosen to compare them against POSMIT and SOC: the MBGP algorithm with a moderate level of evaluation (five out of eight criteria were met) and the CandidateStops algorithm with a low level (two out of eight criteria were met). The MBGP algorithm is commonly used and was applied to another similar dataset of older adults by Fillekes, Kim, *et al.* (2019) but not to MOASIS data. It would be interesting to compare if the threshold analysis results are consistent over different study data. Unlike the other three chosen algorithms, CandidateStops takes speed into account (Nogueira, Braga and Martin, 2014). In case this algorithm should deliver good outcomes, the selection criteria for the algorithm used above would have to be reconsidered. As algorithm comparisons were frequently done with the SMoT algorithms, not all stops are detected depending on the manually defined candidate stops, and their criteria of geographic objects is a time-consuming approach, they were not selected for further analysis.

4.1.2 Source Code Availability

The availability of algorithm's source code to the public was checked because it increases the reproducibility and replicability of analysis. At the time the analysis for this thesis was conducted, DBSCAN and OPTICS algorithms were implemented as functions in R and could be easily called. The R codes for the MBGP algorithm were borrowed from Fillekes, Kim, *et al.* (2019). The SMoT, CB-SMoT, and DB-SMoT algorithms could be borrowed from Krähenbühl (2014). The Python codes for TrajDBSCAN are publicly available provided by Tran *et al.* (2011) on https://github.com/devil1993/TrajDBSCAN. Furthermore, Bermingham (2018) provided the Java scripts for POSMIT and CB-SMoT on https://github.com/lukehb/137-stopmove. However, the POSMIT algorithm had to be rewritten completely by referring to its pseudo code from Bermingham (2018). Fortunately, most of the authors provide the pseudo code in the respective paper.

4.2 Algorithm Implementation

The main goal of the thesis is to perform a basic step of the trajectory mining tasks (i.e., segmentation) proposed by Zheng (2015). Before the trajectory segmentation algorithms (i.e., the stop and move detection algorithms) were implemented or applied, the data needed to be pre-processed.

4.2.1 Data Pre-Processing

After importing the .csv-files into R, the *NA* values and special characters (e.g., tab stops) were dropped. During the time the older adults charged their devices, the word "charging" was stored in the longitude column of the dataset. As it can be assumed that the older adults charged their devices at home and no address information was available, the longitude cells containing "charging" were replaced with a place holder (e.g., -9999). Afterwards the values before and after the charging process were collected and values outside the 95 % confidence interval were removed. With the remaining values, the median was calculated, and the place holder replaced by this value. Finally, a coordinate transformation from WGS84 into LV95 was performed in order to calculate Euclidean distances without having to consider the curvature of the earth.

As the interest lies primarily in detecting stops and not in knowing the exact moving path of the older adults, map-matching was not performed. Instead, noise filtering, more specific heuristic-based outlier detection, was performed (see *Figure 4.2*).

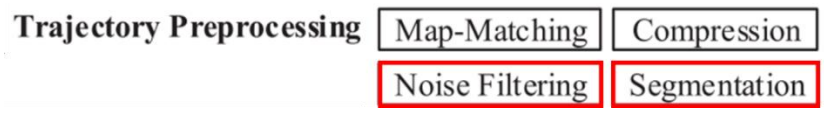


Figure 4.2 Trajectory pre-processing methods (modified from Zheng, 2015:2)

A speed-threshold of 200 km/h was set, which is the high-speed standard of Switzerland's trains, and all values above this threshold were removed as the heuristic-based outlier detection methods do not aim to replace the trajectory values (SBB, no date; Zheng, 2015). As smoothing algorithms, such as the Kalman filter, could bring new random errors when trajectory data is sparse, the other methods were not chosen for noise filtering (Parent et al., 2013). Furthermore, a time series segmentation was necessary in order to split the whole trajectory data into daily segments based on time (Damiani and Hachem, 2017). As a splitting criterion, the time of 3 a.m. was chosen for the functional end/beginning of a day because it can be assumed that people would have finished their daily business by three o'clock in the morning at the latest (Schneider et al., 2013). The trajectories were then segmented by pattern-driven segmentation using stop-move detection algorithms (ibid.).

4.2.2 Algorithm Programming

In general, all the processes were implemented and performed in R. For doing so, the packages dplyr, lubridate, plyr, rgeos, sp, stringr, and tidyr were used for applying the algorithms; ggplot2 and extrafont were used for generating graphs; leaflet was used for visualising the data on an interactive map[6]. On top of implementing the algorithms, the aim of the algorithm programming part was to run the algorithms with different thresholds in order to find the most suitable parameter set for the MOASIS data.

4.2.2.1 CandidateStops

The CandidateStops algorithm was only available in pseudo code and therefore had to be implemented in R. The pseudo code is illustrated in *Algorithm 4.1*. It only uses two input parameters: a list of spatiotemporal data points containing speed and time difference values and a speed threshold (ε).

The CandidateStops algorithm checks for every entry at index *i* if z is greater than the mode of the time difference between index *i* and index *i-1* and if the speed is smaller than the given speed threshold (Nogueira, Braga and Martin, 2014). If these two conditions are met, the point is considered as stop, while all other data points not fulfilling the conditions are classified as moves. The algorithm is a statistical method that works by considering the sampling rate standard z-score as calculated in line 15 of *Algorithm 4.1* and visible in *Equation 4.1*.

$$z = \frac{x - \mu}{\sigma} \tag{4.1}$$

[6] Consult the links below for further information about the R-packages:
https://www.rdocumentation.org/packages/dplyr, https://www.rdocumentation.org/packages/sp,
https://www.rdocumentation.org/packages/rgeos, https://www.rdocumentation.org/packages/plyr,
https://www.rdocumentation.org/packages/tidyr, https://www.rdocumentation.org/packages/ggplot2,
https://www.rdocumentation.org/packages/leaflet, https://www.rdocumentation.org/packages/lubridate,
https://www.rdocumentation.org/packages/stringr, https://www.rdocumentation.org/packages/extrafont,

The metric indicates how far the sample is from the mean of the whole dataset. The mean (μ) and standard deviation (σ) of the time difference between two entries are calculated in lines 10 and 11. The variable *x* is the time difference between index *i* and index *i-1*.

Algorithm 4.1 Pseudo Code of CandidateStops Algorithm (modified from Nogueira, Braga and Martin, 2014:105)

```
 1: Input:
 2: T = (p1, … , pN): list of spatiotemporal data points containing speed and time difference values
 3: ε: speed threshold
 4:
 5: Output:
 6: Tsm: List of resulting stay points
 7:
 8: function CANDIDATESTOPS(T, ε)
 9: Tsm ← ∅;
10:         μ ← mean of diffTime;
11:         σ ← standard deviation of diffTime;
12:         m ← mode of diffTime;
13:
14:         for each speed and duration in v and diffTime do
15:             z ← (duration - μ) / σ;
16:             if z > m and speed < ε then
17:                 Consider as stop;
18:                 Add stop to Tsm;
19:             else
20:                 continue
21:             end if
22:         end for
23: end function
```

4.2.2.2 MBGP Algorithm

The MBGP algorithm had already been implemented by Michelle Fillekes, one of the authors of Fillekes, Kim, *et al.* (2019), and made available for this thesis. *Algorithm 4.2* shows the pseudo code. The MBGP algorithm uses four input parameters: a list of spatiotemporal data points, a distance (D_{max}) and two time thresholds (T_{min} and T_{max}).

Algorithm 4.2 Pseudo Code of MBGP Algorithm (modified from Montoliu, Blom and Gatica-Perez, 2013:190)

Input:
$l_p = (p_1, \dots, p_N)$: list of spatiotemporal data points
T_{min}, T_{max}: time thresholds
D_{max}: distance threshold

Output:
l_{sp}: List of resulting stay points

$i \leftarrow 1$;
$l_{sp} \leftarrow \emptyset$;
while $i < N$ **do**
    $j \leftarrow i + 1$;
    while $j < N$ **do**
        $t \leftarrow$ *TimeDifference*(p_j, p_{j-1});
        if $(t > T_{max})$ **then**
            $i \leftarrow j$;
        end if
        d $\leftarrow$ *SpaceDistance*(p_i, p_j);
        if $d > D_{max}$ **then**
            $t \leftarrow$ *TimeDifference*(p_i, p_{j-1});
            if $t > T_{min}$ **then**
                $[lat, long] \leftarrow$ *EstimateCentroid*$(p_k \mid k \in [i, j-1])$;
                $T^{start} \leftarrow p_i.T$;
                $T^{end} \leftarrow p_{j-1}.T$;
                $sp \leftarrow [lat, long, T^{start}, T^{end}]$;
                $l_{sp} \leftarrow l_{sp} \cup sp$;
            end if
            i $\leftarrow j$;
        end if
        $j \leftarrow j+1$;
    end while
end while

The distance threshold D_{max} can be seen as the diameter of a stop as it represents the maximum allowed distance between the first GPS point of a stop cluster and the last (Montoliu, Blom and Gatica-Perez, 2013). The parameter T_{min} represents the minimum duration between the points in order to count as a stop (ibid.). T_{max} represents the maximum allowed time gap between two recorded location points to be considered as a part of the same stop cluster (ibid.). If the time span between two entries is too long, it could be due to a GPS signal loss and therefore a new stop cluster is assumed. *Figure 4.3* shows how

the three parameters work in order to find stops. The green lines show where the parameters' conditions are met and the red ones where the parameters' conditions are not met.

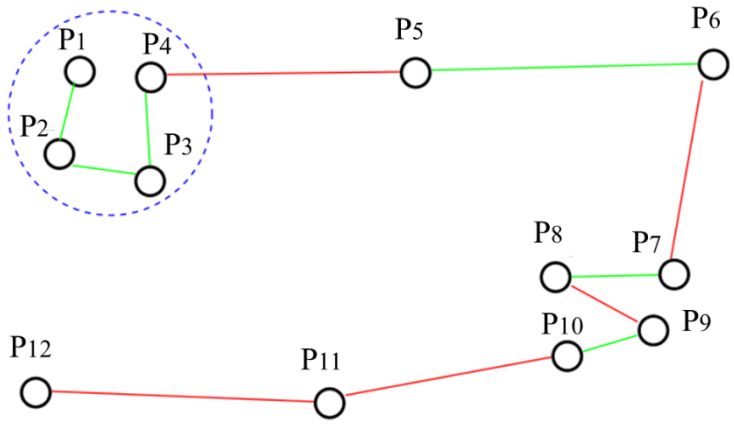


Figure 4.3 Schematic overview of MBGP algorithm (from Montoliu, Blom and Gatica-Perez, 2013:189)

The algorithm checks if the distance from the first point of the stop cluster to the current point is in the range of D_{max} and if the minimum time between a current point and its neighboring point is bigger than T_{min} (Montoliu, Blom and Gatica-Perez, 2013).

If the time span between the points is also less than T_{max}, the point belongs to the stop cluster, or starts a new stop cluster otherwise. In *Figure 4.3* the time span between P8 and P9 is bigger than T_{max} and therefore P9 does not belong to the stop cluster.

If all the stop clusters are detected, their coordinates can be replaced by the coordinates of the centroid of each cluster. This means that two points with the same semantic information share the same location information (ibid.).

4.2.2.3 POSMIT

Bermingham (2018) made the Java source code of the POSMIT algorithm publicly available on https://github.com/lukehb/137-stopmove. However, the code had to be newly implemented in R. The pseudo code is illustrated in *Algorithm 4.3*. It uses four input parameters: a list of spatiotemporal data points, an index search bandwidth (h_i), a stop variance (h_d), and a minimum stop probability (ε).

Algorithm 4.3 Pseudo Code of POSMIT Algorithm (modified from Bermingham, 2018:111)

```
1:  Input:
2:  T = (p1, … , pN): list of spatiotemporal data points
3:  h_i: index search bandwidth
4:  h_d: stop variance
5:  ε: minimum stop probability
6:
7:  Output:
8:  T_sm: List of spatiotemporal data points with added semantics
9:
10: function POSMIT(T, h_i, h_d, ε)
11: T_sm ← ∅;
12:         // Find stop probability of each entry.
13:         for (i ← 0; j < T.length; i++) do
14:             a_i = MOVE;
15:             if CALCSTOPPR(T, i, h_i, h_d) ≥ ε then
16:                 a_i ← STOP;
17:             end if
18:             Add entry ⟨x_i, y_i, t_i, a_i⟩ to T_sm;
19:         end for
20:         return T_sm
21: end function
```

The algorithm checks for each entry at index *i*, if its stop probability (see function CalcStopPr on line 15) is greater than a given minimum stop probability threshold (ε) (Bermingham, 2018). If so, the entry is marked as stop and as move otherwise. The function CalcStopPr is based on the Gaussian kernel smoothing function (*Equation 4.2*):

$$\Pr(Stop|x_i, y_i) = \frac{\sum_{j=l}^{u}\{K(\omega_{i,j})K(\Delta_{i,j})\}}{\sum_{j=l}^{u} K(\Delta_{i,j})} \tag{4.2}$$

The definitions of the helper functions $K(z),\ \Delta_{i,j},\ \omega_{i,j},\ l,\ and\ u$ are explained in detail below.

Here, $K(z)$ that is shown in *Equation 4.3*, represents a gaussian function with a standard deviation and height of one and a mean of zero (i.e., the result is in the range [0,1]) (ibid.). Due to the zero-mean Gaussian function, a small displacement between entries will result in a higher stop probability than entries with larger displacements (ibid.).

$$K(z) = e^{-0.5z^2} \tag{4.3}$$

The function $\Delta_{i,j}$ (*Equation 4.4*) represents the normalised index displacement where entries closer to *i* contribute more to the stop probability calculation (ibid.). In this function, h_i controls the width of the Gaussian function.

$$\Delta_{i,j} = \frac{|\, i - j \,|}{h_i} \tag{4.4}$$

Parameters l and u are lower and upper index search-bounds (*Equations 4.5 and 4.6*). They define the sampling window of indices that are considered for calculating the stop probability of each entry at index *i* (ibid.).

$$l = \max(0, i - 3 * h_i) \tag{4.5}$$

$$u = \min(n, i + 3 * h_i) \tag{4.6}$$

The last function needed for calculating the stop probability is the normalised spatial displacement, $\omega_{i,j}$ that is shown in *Equation 4.7* (ibid.). It is used for calculating the Euclidean distance between the coordinates of each entry *i* and its neighbouring entry *j* that is divided by the stop variance (h_d). Similar to h_i, h_d controls the width of the Gaussian function (ibid.). This function is most fundamental for calculating the stop probability as it spatially defines the notion of a stop. The spatially closer the two entries *i* and *j*, the more likely *i* is to be a stop (ibid.).

$$\omega_{i,j} = \frac{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{h_d} \tag{4.7}$$

The helper functions are inserted into the $\Pr(Stop|x_i, y_i)$ function in order to calculate the stop probability for each entry at index *i*.

4.2.2.4 SOC

The SOC algorithm was only available in pseudo code and therefore had to be implemented in R. The pseudo code is shown in *Algorithm 4.4*. It uses five input parameters: a list of spatiotemporal data points, a geographical distance to define a core sequence (*Eps*), a time duration that defines a core sequence (*Tau*), a minimum duration for a stop (*MinStp*), and a minimum duration for a move (*MinMov*). However, the *MinStp* parameter is not used in the algorithm as it is described in the pseudo-code in Xiang, Gao and Wu (2016:8). Since it is still mentioned by the authors, it is included in the algorithm's parameters but set to *NA* by default. It is assumed that *MinStp* is used to remove too short stops which is done eventually in post-processing.

Algorithm 4.4 Pseudo Code of SOC Algorithm (modified from Xiang, Gao and Wu, 2016:8)

```
 1: Input:
 2: T = (p1, … , pN): list of spatiotemporal data points
 3: Eps: geographical distance to define a core sequence
 4: Tau: time duration that defines a core sequence
 5: MinStp: minimum duration for a stop
 6: MinMov: minimum duration for a move
 7:
 8: Output:
 9: Stops = (s1, … , sM): the set of trajectory stops, each of which is a sequence in T
10:
11: // initialize the reachability distance of each point in T to UNDEFINED;
12: for i ← 1 to |T|
13:         ComputeDistance(T, pi, Eps, Tau);
14: end for
15: Seqs ← Extract_Sequence(T, Eps);
16: Stops ← Merge_Sequence(Seqs, Eps, MinMov);
17: PruneStops(Stops, Eps,Tau);
18: FalsePositiveRemove(Stops);
19:
20: function ComputeDistance(T: in out, pi: in, Eps: in, Tau: in)
21: Seq ← GetEpsSequence(T, pi, Eps);
22:         if Seq is a core sequence with respect to Tau
23:                 r = GetCoreDistance(Seq, Tau)
24:                 for each point q in Seq
25:                         if q precedes pi
26:                                 q.reachability distance = max{r, distance(pi, q)};
27:                         else
28:                                 break;
29:                         end if
30:                 end for
31:         else
32:                 break;
33:         end if
34: end function
```

First of all, helper functions were defined in order to compute the distance of row 13. The function is explained at row 20. Row 21 defines the function GetEpsSequence that is looking for all data points within distance of *Eps*. Row 22 checks for core sequences. If point *i* is in a sequence and the time difference between the first and last data point in the sequence is ≥ *Tau*, a core sequence is detected

(Xiang, Gao and Wu, 2016). Otherwise a value that is significantly higher than *Eps* is returned (in this case 2**Eps*). After detecting a core sequence, the minimal radius (*r*) is defined so that the sequence of any data point (*p*) to *r* is still a core sequence and the time difference still ≥ *Tau* (row 23). At rows 24-26 the reachability distance for each data point with its preceding data point is calculated (ibid.). The reachability distance that is stored for every data point is the higher value of either radius (*r*), or the distance between the data point and its preceding one (ibid.). After computing the distance at row 13, sequences are determined every time there is a change from reachability distance < *Eps* or > Eps (row 15). Row 16 calculates the centres of two consecutive sequences. If the distance is ≤ 2**Eps* and the time difference between the last data point of the first sequence and the first data point of the second sequence is < *MinMov*, the two sequences are merged (ibid.). Out of this merge, a new centre is calculated and compared to the following sequence. Afterwards, out of these sequences all the core points (i.e., data points that fulfil the requirements of a core sequence) are labelled as stops (row 17). Finally, at row 18, all false positives as for example very slow movements potentially occurring in case of traffic jams are removed (ibid.). For that, Xiang, Gao and Wu (2016) use two approaches called straightness and centred-distance (see *Equations 4.8 and 4.9*).

$$Sequence.centred-distance = \frac{\sum_{i=start}^{end-1} distance(p_i, Sequence.center) * (tdiff_{p_{i+1}-p_i})}{\sum_{i=start}^{end-1} tdiff_{p_{i+1}-p_i}} \quad (4.8)$$

If the centred-distance is significantly smaller than *Eps* (i.e., < 0.5**Eps*) and the straightness is > 0.8, then the stop is removed and reclassified as a move.

$$Sequence.straightness = \frac{distance(p_{start}, p_{end})}{\sum_{i=start}^{end-1} distance(p_i, p_i + 1)} \quad (4.9)$$

4.2.3 Data Post-Processing

Once the stops were detected, all moves shorter than 180 seconds were removed and classified as noise. A minimum move duration of 180 seconds is a commonly applied threshold as stated by Fillekes, Kim, *et al.* (2019). Subsequently, the stops that were spatially and temporally close to each other were merged. Therefore, a maximum duration and the maximum distance between the last point of the first stop cluster and the first point of the second stop cluster had to be defined in order to check if the clusters could be merged. The selected thresholds were 1 hour for the maximum duration, and 150 meters for the maximum distance, same as those adopted in Fillekes, Kim, *et al.* (2019). The detected moves were converted into polylines for better visualisation.

4.3 Algorithm Comparison and Evaluation

For evaluating and comparing the algorithms' performance, as well as having a basis for discussion, three scenarios were developed in order to find the algorithms that work best for the corresponding scenario (see *Table 4.2*).

Table 4.2 Evaluation Scenarios

Scenario 1	The dataset is very noisy. Which algorithm is most robust to noise?
Scenario 2	The dataset has a low sampling rate. Which algorithm performs best with varying sampling rates?
Scenario 3	The shapes of the stops should be very compact. Which algorithm's stops have the most compact shapes?

4.3.1 Threshold Analysis

To find the thresholds that deliver the best results, a vector of a range supported by literature was defined for each parameter and is explained for each algorithm in the following sections. Afterwards, the algorithm run several times, and recorded for each time (i.e., every new parameter combination) the number of stops as *Figure 4.4* shows for an imaginary example.

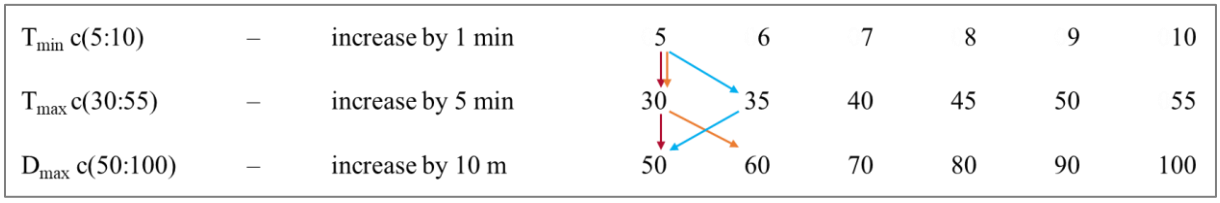


Figure 4.4 Example for the parameters of the MBGP algorithm

The arrows in *Figure 4.4* indicate such combinations: The first combination is the red one, meaning the first column of this matrix. Afterwards, D_{max} is increased while T_{max} and T_{min} stay the same (orange arrows). When all combinations with a changing D_{max} are found, T_{min} stays the same, T_{max} gets increased by one, and D_{max} is iterated through again (indicated by blue arrows) and so forth. In the example that *Figure 4.4* visualises, the algorithm would run $6^3 = 216$ times. With this approach, the influence of each single parameter on the result can be shown.

The threshold combination of the stop detection result that best represented the ground truth before post-processing the results was considered as most suitable for MOASIS data. Additionally, the best threshold combination to represent the post-processed data was also selected. With this approach, it can be shown if post-processing improved the result or whether another threshold combination should be

chosen when it is sure that the data always will be post-processed. To find the optimal combination, the centre of each stop cluster was determined and investigated if it is spatiotemporally close[7] to the centre of the stop cluster of the manually labelled ground truth.

4.3.1.1 CandidateStops

As Nogueira, Braga and Martin (2014) stated, a stop can be characterized by a velocity less than 1 m/s. A velocity of 1 m/s is the most common walking speed for adults up to the age of 75 (Ewert, 2012). Afterwards the velocity decreases to 0.8 m/s up to the age of 85 and stagnates at 0.7 m/s after the age of 85 (ibid.). Ewert (2012) shows that older adults with balance problems even walk with velocities of 0.5 m/s. Therefore, the algorithm run six times with the following speed thresholds:

ε = c(0.5:1) increasing each time by 0.1 m/s

4.3.1.2 MBGP Algorithm

Selecting the parameters D_{max}, T_{min}, and T_{max} of the MBGP algorithm was more complex. As D_{max} is the maximum distance the user can cover in order to count as a stop event, too small a value could divide a place or values too big could merge several places into only one (Montoliu, Blom and Gatica-Perez, 2013). As the interest lies rather in finding stop regions instead of single stop points, values from 150 m to 300 m could be suitable. In that case, the older adult's homes would be included and even some places with size of shopping malls could be detected.

The minimum time a person has to stay in the same place in order to be considered at stop is represented by the parameter T_{min}. Research shows that the average shopping time in a supermarket is about 13 minutes and in larger shopping facilities around 21 minutes (Schneider and Hennig, 2008). Therefore, the parameter was tested starting with 10 minutes (rounded down average supermarket shopping time) although Montoliu, Blom and Gatica-Perez (2013) stated that a T_{min} between 20 and 40 minutes delivers the best results. Because the interests of this thesis do not lie in detecting bus stops and tram stations, small values were not considered.

T_{max} is the maximum timespan between two stop clusters. By setting T_{max} high, the most significant places will be detected and a lot of places of interests could remain undetected. However, the risk of a T_{max} that is set too low is that more false positive places could be discovered (Montoliu, Blom and Gatica-Perez, 2013). It is assumed that a meeting for lunch or for coffee could take about three to four hours and due to the weak GPS signals in buildings, T_{max} should not fall below four hours. Furthermore, older adults tend to spend up to eight hours in bed (Huang *et al.*, 2002). Therefore, the parameter values

[7] Meaning the centres of the stop clusters are in a range of 150 meters and the start times of the respective clusters lie within a time interval of 20 minutes.

of T_{max} chosen for comparison were between four and eight hours. This resulted in the following vectors and 245 runs of the algorithm:

D_{max} = c(150:300) increasing each time by 25 meters

T_{min} = c(10:40) increasing each time by 5 minutes

T_{max} = c(4:8) increasing each time by 1 hour

4.3.1.3 POSMIT

For POSMIT, Bermingham (2018) describes three approaches, that can be performed in order to find the most appropriate thresholds for the search bandwidth (h_i), the stop variance (h_d), and the minimum stop probability (ε).

To determine the stop variance (h_d), Satopää et al. (2011) propose a Kneedle function in order to find elbow points in non-continuous datasets. The main goal of this function is to sort the spatial displacement between each entry and its neighbouring entry in the trajectory in ascending order to find elbow points (i.e., point of maximum curvature in a function) that should indicate a point of change (i.e., change from move to stop, h_d value) (Bermingham, 2018).

A smoothing spline is adapted to preserve the shape of the original dataset (Satopää *et al.*, 2011). Afterwards, the data is normalised into the unit-square $[0, 1]^2$ using a min-max normalisation (ibid.). Finally, (x, y-x) is performed on each pair of values in order to get the set of differences between the x and y values (ibid.). This difference transformation should highlight when "the difference curve changes from horizontal to sharply decreasing" (Satopää *et al.*, 2011:168). That would indicate an elbow in the dataset. The elbow point can be extracted from the transformed data as it is the local maxima of the curve (Bermingham, 2018). In case there were multiple local maxima, the global maximum was taken.

Bermingham (2018) proposes a heuristic in order to estimate the search bandwidth (h_i). At the beginning, entries are determined that are $\leq h_d$ meters away from their neighbouring entry. As long as the entries meet the condition, the number of contiguous entries is counted and stored if the number of entries is at least two. If one entry is $\geq h_d$, the counting stops and starts again when another row of contiguous entries that meet the criteria occurs (Bermingham, 2018). When the process reaches the end of the entries, the median is calculated out of the stored values and divided by two in order to get h_i. The division by two is necessary as the parameter h_i considers both directions (ibid.).

To estimate the minimum stop probability (ε), Bermingham (2018) calculates the stop probability for each data point and clusters the data into two clusters (i.e., stops and moves) using the k-means clustering method (k = 2). After building the clusters, Bermingham (2018)'s approach was adapted: the maximum stop probability of the move cluster and the minimum stop probability of the stop cluster were averaged.

The average out of these two values counted as the minimum stop probability (ε) (ibid.). The three parameters were estimated for each dataset as compared to fixed parameter values as well.

Taking out of Bermingham (2018)'s experiments, the minimum stop probability was tested with the fixed values of 0.25, 0.5, and 0.75. As for events like shopping or hiking, Bermingham (2018) got best results, when $h_i \leq 5$. Therefore, the values 1:5 were iterated through. As the estimated h_d got the best classification result in Bermingham (2018), it was not going to be further compared to different values. This resulted in the following vectors and the algorithm in running 24 times:

h_d = estimated h_d

h_i = c(estimated h_i, 1, 2, 3, 4, 5)

ε = c(estimated ε, 0.25, 0.5, 0.75)

4.3.1.4 SOC

For the SOC *MinMov* parameter, 180 seconds can be seen as an appropriate time duration that a normal moving behaviour should at least last, as this is a commonly applied threshold (Xiang, Gao and Wu, 2016). For *MinStp*, no parameter value is set (see *Section 4.2.2.4* for further explanation).

Tau and *Eps* define the boundary between stop and move in terms of how far a person can get before a stop becomes a move. The two parameters can be seen as equivalent to T_{min} and $D_{max}/2$ from the MBGP algorithm. $D_{max}/2$ was chosen because *Eps* is the radius of the core sequence instead of the diameter as D_{max} is in the MBGP algorithm. This resulted in the following vectors and in running the algorithm 28 times:

Eps = c(75:150) increasing each time by 25 meters

Tau = c(10:40) increasing each time by 5 minutes

MinMov = 180 seconds

4.3.2 Ground Truth Collection

To find the most suitable threshold, the algorithms' outcomes were compared to the manually labelled ground truth introduced in *Section 3.2*. Furthermore, the ground truth was replaced with the probabilistic formula introduced in Fedorchuk and Lamiroy (2017) in order to compute the probabilistic metrics and compare these results to the F-measure values of the algorithms validated with the manually labelled ground truth. The approach of the replaced ground truth will further be elaborated in *Section 4.3.3*.

4.3.3 Probabilistic Metrics

For calculating the probabilistic metrics, all results whose parameter combination fitted the manually labelled ground truth best were taken. The following approach is based on the assumption that each point has the same probability of being stop or move.

For each data point, the information was available if the algorithms detected them as stop (1) or move (0). Furthermore, two columns were added to the data (similar as in *Table 4.3)* that indicated that every data point is a stop or move, respectively (Lamiroy and Sun, 2013). Out of this scheme, the probability for each point being a stop was calculated (i.e., this is the replaced ground truth).

Table 4.3 Example of calculating probabilistic metric F-measure (modified from Lamiroy and Sun, 2013:7)

	probability that point is a stop	every point is a stop	CandidateStops	MBGP algorithm	POSMIT	SOC	every point is a move
data point 1	0.67	1.00	0.00	1.00	1.00	1.00	0.00
data point 2	0.83	1.00	1.00	1.00	1.00	1.00	0.00
data point 2	0.50	1.00	0.00	1.00	0.00	1.00	0.00
sum	2.00	3.00	1.00	3.00	2.00	3.00	0.00
$\sum probabilities$		2.00	0.83	2.00	1.50	2.00	0.00
precision		0.67	0.83	0.67	0.75	0.67	∞
recall		1.00	0.42	1.00	0.75	1.00	0.00
F-measure		0.80	0.56	0.80	0.75	0.80	

The precision was calculated out of the sum of the probabilities (∑*probabilities*) divided by the sum of the detected stops (coloured in blue and orange in *Table 4.3*). The recall was calculated out of the ∑*probabilities* divided by the sum of the probability that a point is a stop (coloured in blue and green in *Table 4.3*). The F-measure was calculated as the harmonic-mean between precision and recall (Fedorchuk and Lamiroy, 2017).

In order to compare the F-measure results explained in *Table 4.3* to the results of the manually labelled ground truth and for finding the optimal threshold combination in the first place, the F-measure was also computed with regard to true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) as introduced in *Section 2.4.1.1* (Lamiroy and Sun, 2013). The F-measure was also calculated for the results of the following sensitivity analysis methods.

4.3.4 Sensitivity Analysis

The sensitivity was analysed first by determining the average number and variance of stops per day and second with a varying regular and random sampling rate. To find the most appropriate thresholds, the algorithms run with the complete dataset where each point was approximately recorded every second. The varying sampling rate approach was applied only to the threshold combination, that seemed to fit the algorithms' results best. The regular sampling rates were a data point every 30 seconds, every minute, every five minutes and every ten minutes:

Regular sampling rate [s] = c(1, 30, 60, 300, 600)

With the function *sample_frac* of the dplyr R package, that was already mentioned in *Section 4.2.2*, a random amount of data points can be selected. In order to test the sampling rate robustness with a random sampling rate, a fraction of 30 % of the data points was randomly selected and also applied to the chosen threshold combination.

Similar to the approach of Domingues *et al.* (2018), the algorithms' behaviour when increasing the proportion of noise in the data was analysed. These results were compared to the ground truth. According to Bösche *et al.* (2013), adding Gaussian noise to the data would lead to jumps from one side of a road to the other, which is unusual in open areas away from buildings or tunnels.

Hence, there are R packages available that use the house noise model for ordinal variables to increase the proportion of noise in a dataset (Zhang, Liu and Song, 2015). Using the R package FunChisq[8], noise was added to the longitude column of the dataset with the *add.noise* function. The advantage of this function is that the true values were replaced by noise and the sum of each column stayed the same. According to the package description, the noise level of 0.5 creates the most random pattern. Therefore, a noise level of 0.25 was applied as no noise is added to the data at a level of 0.

[8] Consult the link for further information about the R-package: https://www.rdocumentation.org/packages/FunChisq

4.3.5 Shape Measures

In order to compare the shapes of the algorithms' results, the compactness measure of Ebdon (1985) cited in Kitchin and Tate (2000) that compares the shape of a polygon with the shape of a circle was selected (see *Equation 4.10*).

$$f_{comp} = \frac{4A}{\pi D^2} \tag{4.10}$$

The smallest enclosing circle was calculated with the R package shotGroups[9] that contains a function called *getMinCircle*. The diameter was determined by taking the square root of the division of the circle's area by π and then multiplying the result by two. The area of the stop was determined by calculating the convex-hull's area with the R package geometry[10] and the function *convhulln*. For each algorithm, the median of the shape index values was taken.

4.3.6 Running Time

After the optimal threshold combination was chosen, the algorithms' running times were determined by running the algorithms several times with different data frames and taking the mean of the running times.

[9] Consult the link for further information about the R-package: https://www.rdocumentation.org/packages/shotGroups
[10] Consult the link for further information about the R-package: https://www.rdocumentation.org/packages/geometry

5 Results

This chapter gives an overview of the evaluation on the selected algorithms. In order to make the results comparable, they all were visualised and compared to the same 90 study days, where ground truth was labelled manually. For choosing the most suitable threshold combination, a bar chart was included in the boxplot figures in order to show for what percentage of the test sets an F-measure was calculated. The results of the four algorithms are displayed in the following order for each algorithm: 1) threshold selection, 2) handling noise and sampling rate, and 3) shape measures. At the end of this chapter there is an overview of the results to draw the bigger picture (see *Section 5.6*).

5.1 CandidateStops

The CandidateStops algorithm had to be run six times for each testing day in order to build all threshold combinations as it only needs one input parameter, speed [m/s].

5.1.1 Threshold Selection

As *Figure 5.1* shows, the CandidateStops algorithm only delivered an F-measure value for about 3 % of the test sets for a speed threshold of 0.5 m/s, which is more than for the other thresholds. Furthermore, the F-measure value range and median was highest at a speed threshold of 0.5 m/s.

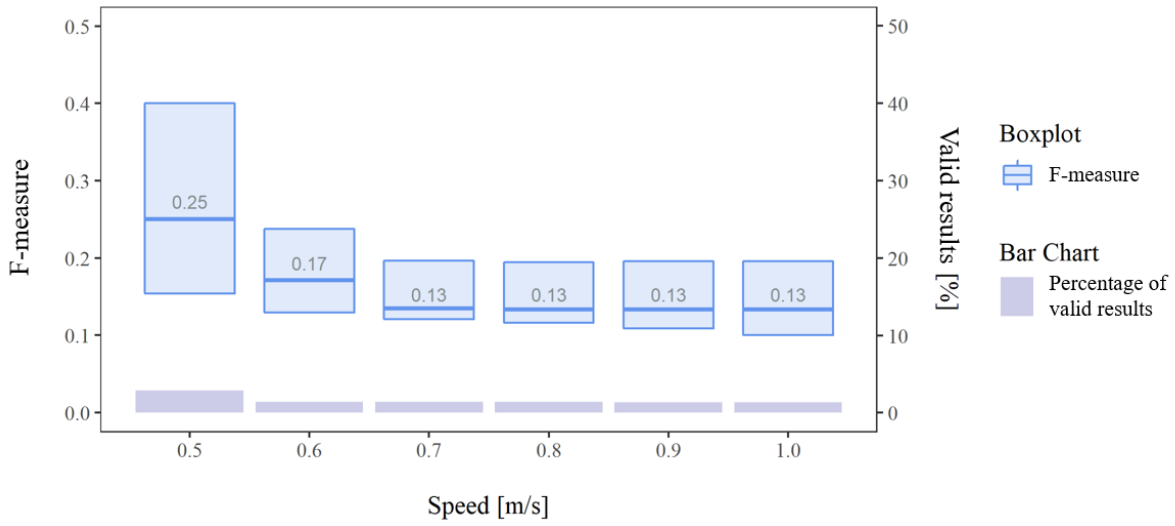


Figure 5.1 F-measure comparison of different speed thresholds of CandidateStops algorithm

The results of the boxplot statistics are visualised in *Table 5.1*. The selected threshold of 0.5 m/s is highest for all of the key distribution parameters except for the minimum value. The lowest performance was reached by a speed threshold of 1.0 m/s.

Table 5.1 Boxplot statistics of CandidateStops algorithm

Speed Threshold [m/s]	Min.	Q1	Median	Mean	Q3	Max.
0.5	0.112	**0.154**	**0.250**	**0.288**	**0.400**	**0.500**
0.6	**0.117**	0.129	0.171	0.206	0.238	*0.400*
0.7	0.091	0.121	0.135	0.180	0.196	*0.400*
0.8	0.083	0.116	*0.133*	0.177	*0.194*	*0.400*
0.9	0.069	0.109	*0.133*	0.173	0.196	*0.400*
1.0	*0.065*	*0.100*	*0.133*	*0.170*	0.195	*0.400*

By looking at the results of the post-processed CandidateStops algorithm in *Figure 5.2*, one can see that all the results improved, for the percentage of valid results as well as for the F-measure values. Still, a speed threshold of 0.5 m/s seems to deliver the best outcomes for the post-processed results as well.

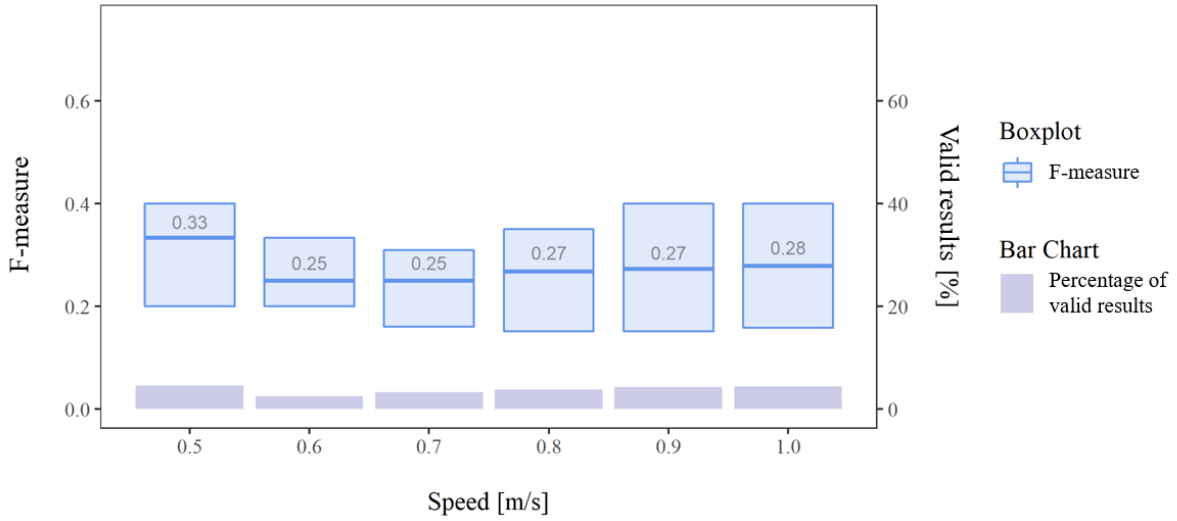


Figure 5.2 F-measure comparison of different speed thresholds of post-processed CandidateStops algorithm

This is confirmed by the boxplot statistics in *Table 5.2*. All of the key distribution parameters increased for a speed threshold of 0.5 m/s except for the maximum value. It stayed the same whereas it increased for the other speed thresholds 0.7 m/s, 0.8 m/s, 0.9 m/s, and 1.0 m/s.

Table 5.2 Boxplot statistics of post-processed CandidateStops algorithm

Speed Threshold [m/s]	Min.	Q1	Median	Mean	Q3	Max.
0.5	**0.125**	**0.200**	**0.333**	**0.320**	**0.400**	0.500
0.6	*0.118*	**0.200**	*0.250*	*0.253*	0.333	*0.400*
0.7	**0.125**	0.160	*0.250*	0.273	*0.310*	0.670
0.8	**0.125**	*0.151*	0.268	0.285	0.350	0.670
0.9	**0.125**	*0.151*	0.273	0.324	**0.400**	**0.750**
1.0	**0.125**	0.158	0.279	0.331	**0.400**	**0.750**

5.1.2 Handling Noise and Sampling Rate

After the speed threshold of 0.5 m/s was chosen, the algorithm was applied to the test sets with added noise and varying sampling rates. As *Figure 5.3* shows, the best results in terms of most valid results and highest median value are the randomly selected 30 % and a regular sampling rate of 600 s. The test sets with added noise had the same percentage of valid results compared to the pre-processed data, however the median F-measure value is higher for the test sets with added noise.

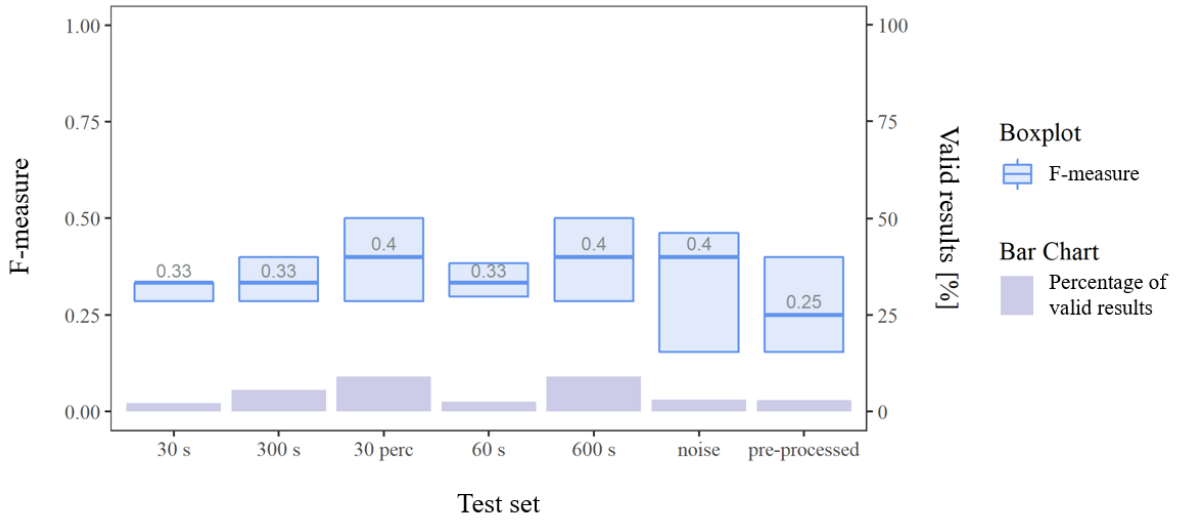


Figure 5.3 F-measure comparison of different test sets of CandidateStops algorithm

In *Table 5.3*, one can see that the randomly selected 30 % sampling rate and the regular sampling rate of 600 s have values of perfect precision and recall as the maximum F-measure value is 1. The pre-processed original data performed worst for most of the key distribution parameters.

Table 5.3 Boxplot statistics of different test sets of CandidateStops algorithm

Test set	Min.	Q1	Median	Mean	Q3	Max.
regular sampling rate 30 s	**0.250**	0.286	0.333	0.374	*0.333*	0.667
regular sampling rate 300 s	0.200	0.286	0.333	0.386	0.400	0.667
randomly selected 30 %	0.200	0.286	**0.400**	**0.451**	**0.500**	**1.000**
regular sampling rate 60 s	0.222	**0.298**	0.333	0.374	0.383	0.667
regular sampling rate 600 s	0.200	0.286	**0.400**	**0.451**	**0.500**	**1.000**
added noise	*0.089*	*0.154*	**0.400**	0.310	0.462	*0.500*
pre-processed data	0.112	*0.154*	*0.250*	*0.288*	0.400	*0.500*

In *Figure 5.4*, it can be seen that the fraction of randomly selected 30 %, the regular sampling rate of 600 s, and with added noise, are lower for the post-processed results compared to the pre-processed ones. The post-processed CandidateStops algorithm performs best with a regular sampling rate of 60 s.

Although the range of the resulting values of the regular sampling rate of 300 s is higher, the percentage of valid results and the median are slightly higher for a regular sampling rate of 60 s.

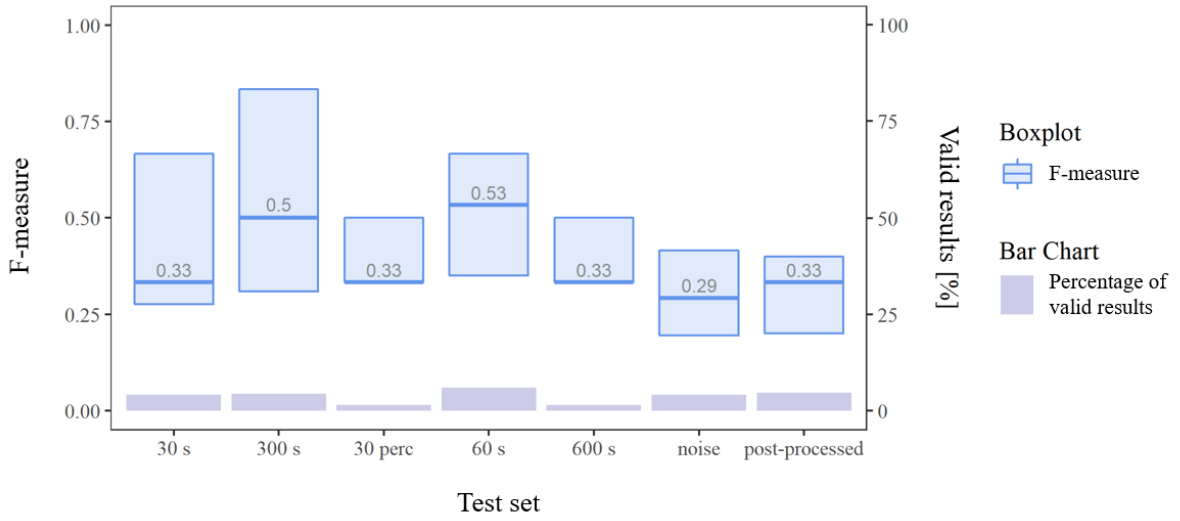


Figure 5.4 F-measure comparison of different test sets of post-processed CandidateStops algorithm

Table 5.4 shows that the regular sampling rate of 30 s, 300 s and 600 s have values of perfect precision and recall as the maximum F-measure value is 1. Compared to the results of the only pre-processed test sets, the randomly selected 30 % and the regular sampling rate of 600 s have lower median F-measure values and were only able to calculate F-measure values for a small number of test sets. The performance of the test sets with added noise decreased as well. Hence, the results of the test sets with a sampling rate of 30 s, 300 s, 60 s, and the post-processed original test sets increased compared to the only pre-processed versions.

Table 5.4 Boxplot statistics of different test sets of post-processed CandidateStops algorithm

Test set	Min.	Q1	Median	Mean	Q3	Max.
regular sampling rate 30 s	0.182	0.277	0.333	0.465	0.667	**1.000**
regular sampling rate 300 s	0.200	0.310	0.500	**0.569**	**0.833**	**1.000**
randomly selected 30 %	**0.333**	0.333	0.333	0.444	0.500	0.667
regular sampling rate 60 s	0.286	**0.350**	**0.533**	0.542	0.667	**1.000**
regular sampling rate 600 s	**0.333**	0.333	0.333	0.444	0.500	0.667
added noise	*0.125*	*0.195*	*0.292*	*0.307*	0.415	*0.500*
post-processed data	*0.125*	0.200	0.333	0.320	*0.400*	*0.500*

5.1.3 Shape Measures

For a regular sampling rate of 300 s, 60 s, and 600 s, as well as for the randomly selected 30 % sampling rate, no shape index values could be determined as there needed to be three different point locations in order to determine the area of the convex hull. This means that when the sampling rate exceeds 30 s, the

determined stops contain only single points instead of a stop area. The least compact shapes result when a regular sampling rate of 30 s is used, as *Figure 5.5* shows (a shape index value of 1 would indicate most compact shapes).

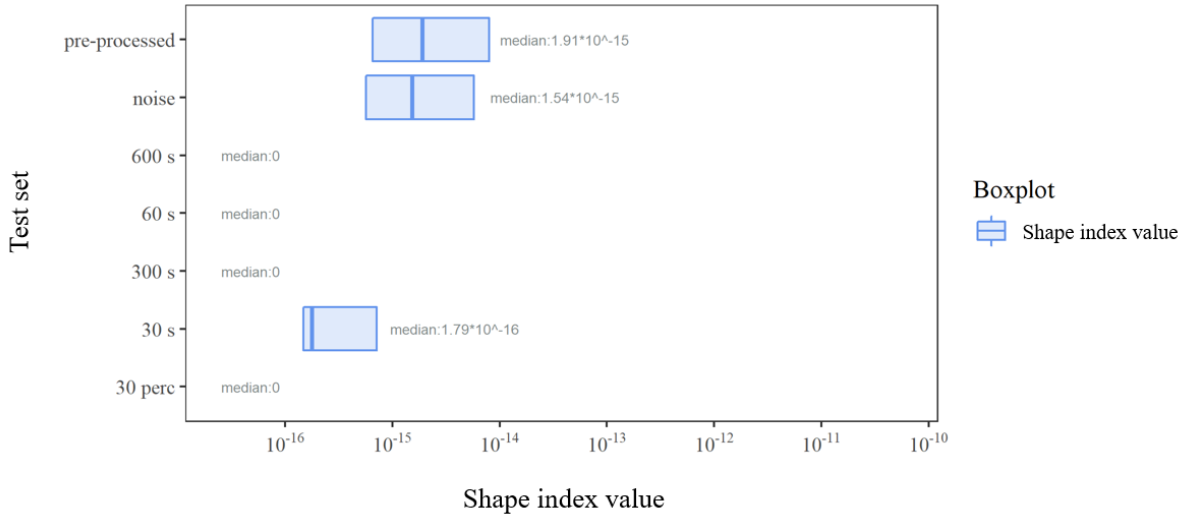


Figure 5.5 Comparison of shape index value of different test sets of CandidateStops algorithm

As *Table 5.5* shows the most compact shapes were detected by the pre-processed original data with the sampling rate of 1 s as this has the values closest to one. However, the regular sampling rate of 30 s has a higher minimal value than the pre-processed data and the test sets with added noise. The manually labelled ground truth has, in general, more compact shapes than those generated by the only pre-processed CandidateStops algorithm.

Table 5.5 Boxplot statistics of shape index values of different test sets of CandidateStops algorithm

Test set	Min.	Q1	Median	Mean	Q3	Max.
pre-processed data	$2.46*10^{-17}$	**$6.57*10^{-15}$**	**$1.91*10^{-15}$**	**$8.04*10^{-14}$**	**$8.01*10^{-15}$**	**$5.88*10^{-11}$**
added noise	$2.46*10^{-17}$	$5.67*10^{-16}$	$1.54*10^{-15}$	$1.77*10^{-14}$	$5.79*10^{-15}$	$3.84*10^{-11}$
regular sampling rate 600 s	*0.00*	*0.00*	*0.00*	*0.00*	*0.00*	*0.00*
regular sampling rate 60 s	*0.00*	*0.00*	*0.00*	*0.00*	*0.00*	*0.00*
regular sampling rate 300 s	*0.00*	*0.00*	*0.00*	*0.00*	*0.00*	*0.00*
regular sampling rate 30 s	**$1.23*10^{-16}$**	$1.51*10^{-16}$	$1.79*10^{-16}$	$1.04*10^{-15}$	$1.50*10^{-15}$	$2.83*10^{-15}$
randomly selected 30 %	*0.00*	*0.00*	*0.00*	*0.00*	*0.00*	*0.00*
manually labelled ground truth	$1.99*10^{-11}$	$6.72*10^{-10}$	$3.08*10^{-09}$	$6.97*10^{-08}$	$7.79*10^{-09}$	$8.71*10^{-07}$

The post-processed data have, in general, more compact shapes than only the pre-processed ones. Furthermore, different from the pre-processed results, the post-processed results were able to calculate shape index values for the regular sampling rate of 60 s. However, the data are rather wide-spread for the regular sampling rate of 60 s as the interquartile range of the boxplot in *Figure 5.6* is the biggest.

Compared to the only pre-processed values, the median of the test sets with added noise is higher than the median of the original data with a sampling rate of 1 s.

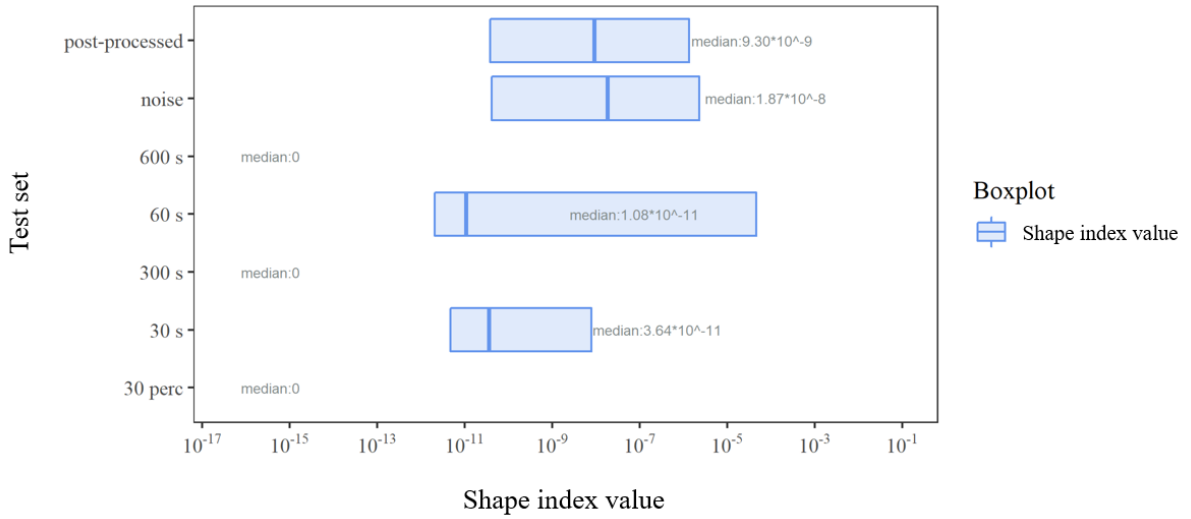


Figure 5.6 Comparison of shape index value of different test sets of post-processed CandidateStops algorithm

Although a regular sampling rate of 60 s has the biggest interquartile range, its minimum shape index value is higher than the minimum shape index value of the other test sets (see *Table 5.6*). Therefore, the range of data including the outliers, is smaller for the data with a regular sampling rate of 60 s compared to the other test sets. This indicates that this sampling rate produces more homogeneous shapes. The manually labelled ground truth has in general more compact shapes than the post-processed CandidateStops algorithm.

Table 5.6 Boxplot statistics of shape index values of different test sets of post-processed CandidateStops algorithm

Test set	Min.	Q1	Median	Mean	Q3	Max.
post-processed data	$3.82*10^{-17}$	$3.87*10^{-11}$	$9.30*10^{-09}$	$3.75*10^{-05}$	$1.34*10^{-06}$	$3.38*10^{-03}$
added noise	$3.82*10^{-17}$	**$4.21*10^{-11}$**	**$1.87*10^{-08}$**	$3.48*10^{-05}$	$2.32*10^{-06}$	$4.34*10^{-03}$
regular sampling rate 600 s	*0.00*	*0.00*	*0.00*	*0.00*	*0.00*	*0.00*
regular sampling rate 60 s	**$4.44*10^{-13}$**	$2.06*10^{-12}$	$1.08*10^{-11}$	**$9.78*10^{-03}$**	**$4.63*10^{-05}$**	**$1.12*10^{-01}$**
regular sampling rate 300 s	*0.00*	*0.00*	*0.00*	*0.00*	*0.00*	*0.00*
regular sampling rate 30 s	$1.20*10^{-13}$	$4.69*10^{-12}$	$3.64*10^{-11}$	$9.44*10^{-05}$	$7.94*10^{-09}$	$2.81*10^{-03}$
randomly selected 30 %	*0.00*	*0.00*	*0.00*	*0.00*	*0.00*	*0.00*
manually labelled ground truth	$1.99*10^{-11}$	$6.72*10^{-10}$	$3.08*10^{-09}$	$6.97*10^{-08}$	$7.79*10^{-09}$	$8.71*10^{-07}$

5.2 MBGP Algorithm

As the MBGP algorithm had to be run 245 times for each test set (i.e., 22'050 times for the original test sets without the varied sampling rate and added noise) in order to build all threshold combinations. The five combinations with the highest key distribution parameters and percentage of valid results were pre-selected before creating the boxplots for visualising the results. By looking at the parameter T_{max}, while parameters D_{max} and T_{min} vary, the F-measure results for each T_{max} value were exactly the same as *Table 5.7* shows. For the post-processed results, the F-measure results were the same as well. Therefore, the T_{max} value was set to 14400 s before selecting the best threshold combination.

Table 5.7 Boxplot statistics of pre-processed T_{max} values

T_{max} [s]	Min.	Q1	Median	Mean	Q3	Max.
14400	0.091	0.444	0.800	0.690	1.000	1.000
18000	0.091	0.444	0.800	0.690	1.000	1.000
21600	0.091	0.444	0.800	0.690	1.000	1.000
25200	0.091	0.444	0.800	0.690	1.000	1.000
28800	0.091	0.444	0.800	0.690	1.000	1.000

5.2.1 Threshold Selection

As the threshold for T_{max} was set to 14400 s, the threshold combinations for D_{max} and T_{min} had to be determined out of the remaining 49 combinations. Threshold combinations containing parameter D_{max} = 150 m, 175 m, and 300 m were not among the top five combinations as the percentage of valid results was the smallest. Threshold combinations containing parameter T_{min} = 600 s, 1500 s, and 1800 s were also not among the top five combinations as the percentage of valid results was smaller compared to the selected ones visualised in *Figure 5.7*. The threshold combination 275/14400/900 had the highest percentage of valid results when looking at the bar charts. Its median F-measure value is only slightly lower than the F-measure value of the combination 250/14400/900. Therefore, the threshold combination 275/14400/900 was chosen.

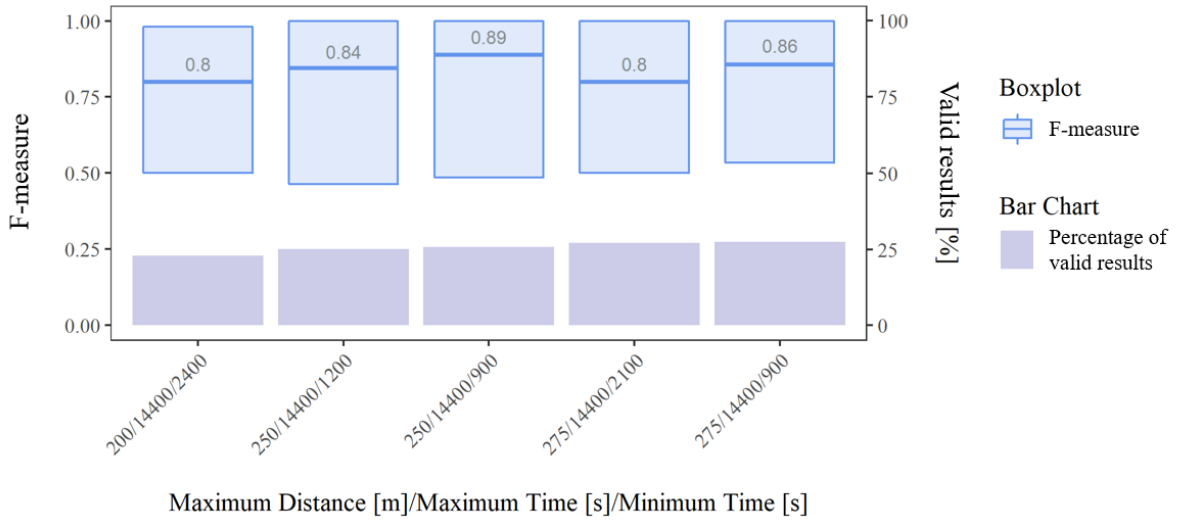


Figure 5.7 F-measure comparison of top five threshold combinations of MBGP algorithm

The selected threshold combination can be further explained with the help of *Table 5.8*. The first quartile of the selected threshold combination 275/14400/900 is higher than the first quartile of the combination with the highest median value and their mean values are almost the same. The third quartile and the maximum value are the same for all of the combinations except for the combination 200/14400/2400.

Table 5.8 Boxplot statistics of top five threshold combinations of MBGP algorithm

D_{max} [m]/T_{max} [s]/T_{min} [s]	Min.	Q1	Median	Mean	Q3	Max.
200/14400/2400	**0.133**	0.500	*0.800*	*0.684*	*0.981*	**1.000**
250/14400/1200	0.125	*0.464*	0.844	0.701	**1.000**	**1.000**
250/14400/900	0.125	0.485	**0.889**	**0.745**	**1.000**	**1.000**
275/14400/2100	0.105	0.500	*0.800*	0.738	**1.000**	**1.000**
275/14400/900	*0.100*	**0.533**	0.857	0.744	**1.000**	**1.000**

By looking at the results in *Figure 5.8*, the F-measure values of the top five combinations are lower compared to the pre-processed ones. However, the percentage of valid results increased slightly. The D_{max} value of 250 m dropped out of the top five combinations. Hence, the T_{min} value of 1500 s is new among the top five combinations. The chosen threshold combination for the pre-processed data (275/14400/900) has the second lowest median of F-measure values. The combination 200/14400/600 has the smallest percentage of valid results but the highest median of F-measure values. The same median of F-measure values is yielded by the threshold combination 275/14400/2100 and a higher percentage of valid results. Therefore, the threshold combination 275/14400/2100 is considered most suitable for the post-processed data run with the MBGP algorithm.

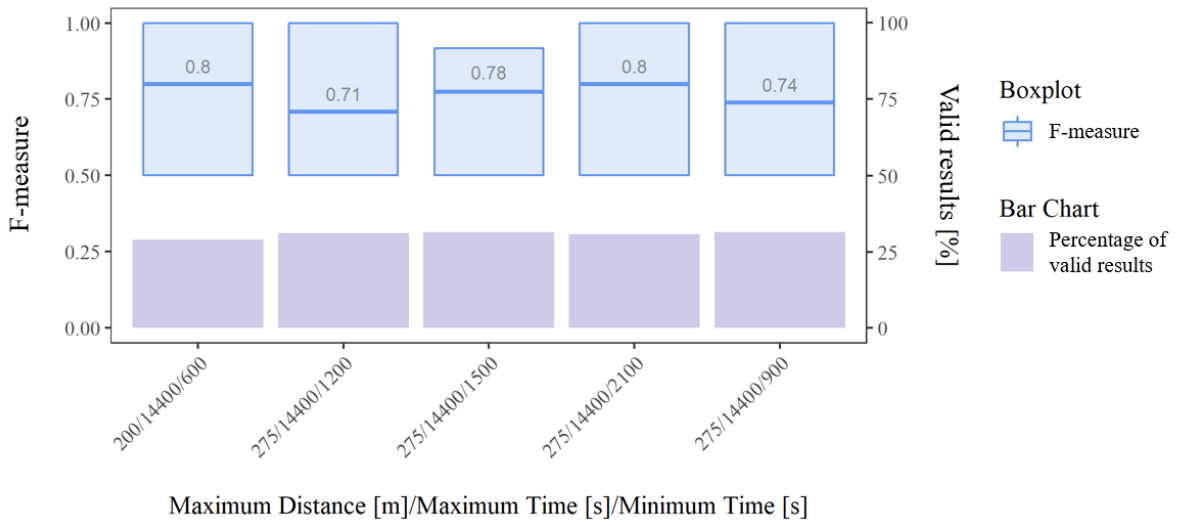


Figure 5.8 F-measure comparison of top five threshold combinations of post-processed MBGP algorithm

By looking at the boxplot statistics in *Table 5.9*, one can see that the key distribution parameters are highest for the selected threshold combination 275/14400/2100 except for the minimal value. Compared to the only pre-processed results, the minimal values increased.

Table 5.9 Boxplot statistics of top five threshold combinations of post-processed MBGP algorithm

D_{max} [m]/T_{max} [s]/T_{min} [s]	Min.	Q1	Median	Mean	Q3	Max.
200/14400/600	*0.154*	**0.500**	**0.800**	0.702	**1.000**	**1.000**
275/14400/1200	0.167	**0.500**	*0.708*	*0.697*	**1.000**	**1.000**
275/14400/2100	0.167	**0.500**	**0.800**	**0.708**	**1.000**	**1.000**
275/14400/900	0.167	**0.500**	0.739	0.706	**1.000**	**1.000**
275/14400/1500	**0.182**	**0.500**	0.775	0.707	*0.917*	**1.000**

As the threshold combination 275/14400/900 is more suitable for the pre-processed data than the combination 275/14400/2100, that seemed to be slightly more suitable for the post-processed data, further analysis was therefore carried out with both combinations.

5.2.2 Handling Noise and Sampling Rate

After the two combinations were chosen, the algorithm was applied to the test sets with added noise and varying sampling rates. First, the combination 275/14400/900 is visualised and compared to the post-processed results and afterwards the results for the combination 275/14400/2100 are shown.

5.2.2.1 Handling Noise and Sampling Rate with T_{min} of 900 s

As *Figure 5.9* shows, the best result for T_{min} = 900 s in terms of most valid results and highest median value is the pre-processed original data, followed by the test sets with added noise. In terms of the

percentage of valid results and the F-measure median, the regular sampling rate of 60 s outperformed the regular sampling rate of 30 s.

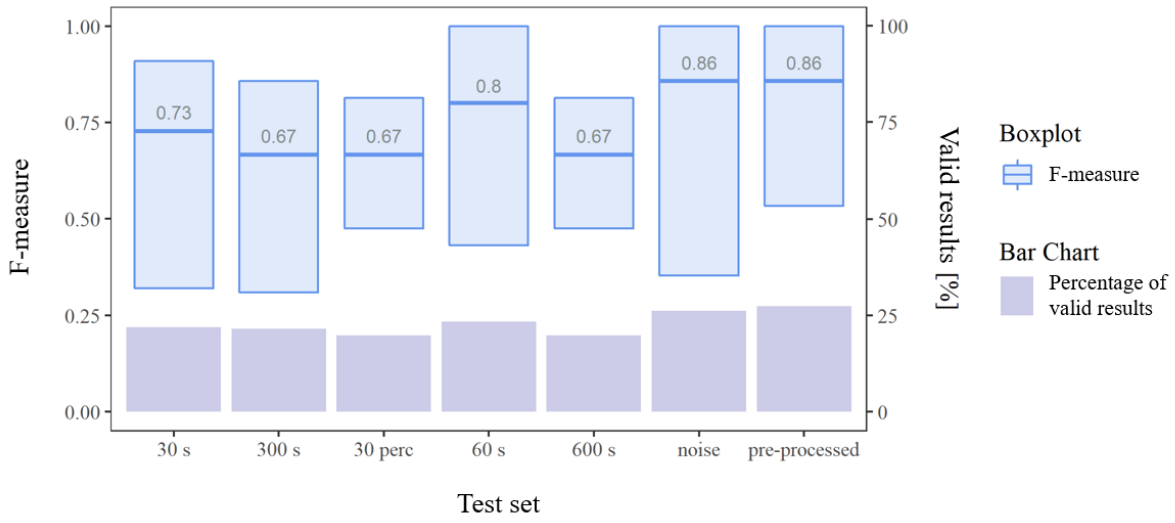


Figure 5.9 F-measure comparison of different test sets of MBGP algorithm [T_{min} = 900 s]

Table 5.10 shows that the pre-processed original data have more low value outliers compared to the other test sets, as the minimal value of the pre-processed data is the lowest. However, the pre-processed original data performed best for most of the key distribution parameters.

Table 5.10 Boxplot statistics of different test sets of MBGP algorithm [T_{min} = 900 s]

Test set	Min.	Q1	Median	Mean	Q3	Max.
regular sampling rate 30 s	0.111	0.319	0.727	0.637	0.909	**1.000**
regular sampling rate 300 s	0.118	*0.310*	*0.667*	*0.623*	0.857	**1.000**
randomly selected 30 %	**0.125**	**0.475**	*0.667*	0.634	*0.814*	**1.000**
regular sampling rate 60 s	0.111	0.431	0.800	0.680	**1.000**	**1.000**
regular sampling rate 600 s	**0.125**	**0.475**	*0.667*	0.634	*0.814*	**1.000**
added noise	**0.125**	0.353	**0.857**	0.713	**1.000**	**1.000**
pre-processed data	*0.100*	0.533	**0.857**	**0.744**	**1.000**	**1.000**

In *Figure 5.10*, it can be seen that the median F-measure values are lower for the post-processed data compared to the pre-processed data. This behaviour was also detected for the CandidateStops algorithm. The percentage of valid results increased for the post-processed original data and noise compared to their pre-processed equivalents, whereas the percentage of valid results decreased for the test sets with varying sampling rates. In opposition to the pre-processed results, the regular sampling rate of 30 s outperforms the regular sampling rate of 60 s in terms of the median F-measure value and the percentage of valid results.

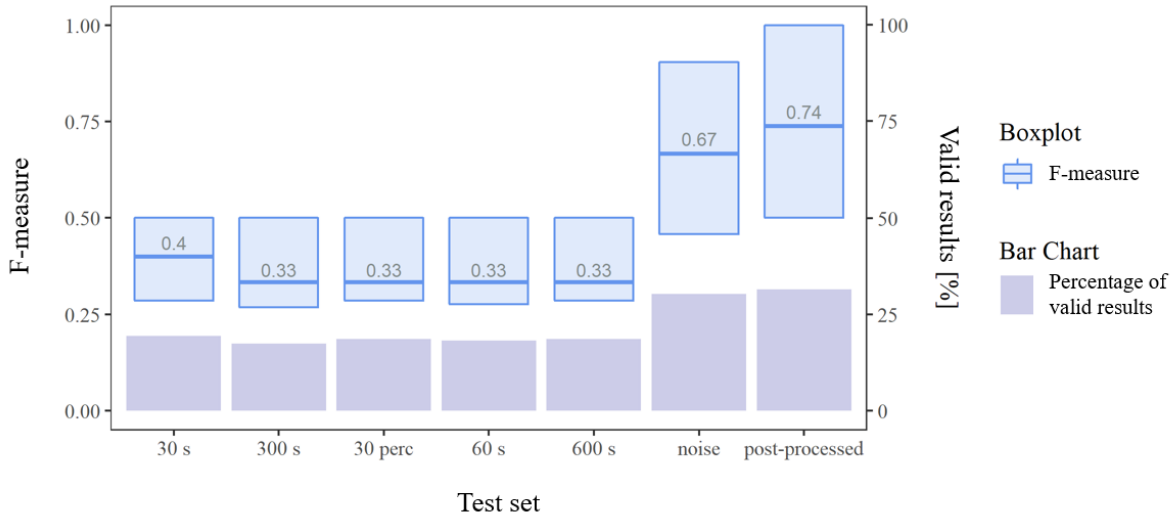


Figure 5.10 F-measure comparison of different test sets of post-processed MBGP algorithm [T_{min} = 900 s]

Furthermore, the F-measure values of the test sets with varying sampling rates are more similar for the post-processed results compared to the pre-processed ones (see *Table 5.11*). The third quartile in particular is the same for all the different sampling rates.

Table 5.11 Boxplot statistics of different test sets of post-processed MBGP algorithm [T_{min} = 900 s]

Test set	Min.	Q1	Median	Mean	Q3	Max.
regular sampling rate 30 s	**0.182**	0.286	0.400	0.425	*0.500*	**1.000**
regular sampling rate 300 s	**0.182**	*0.268*	*0.333*	0.402	*0.500*	**1.000**
randomly selected 30 %	**0.182**	0.286	*0.333*	*0.399*	*0.500*	**1.000**
regular sampling rate 60 s	**0.182**	0.277	*0.333*	0.410	*0.500*	**1.000**
regular sampling rate 600 s	**0.182**	0.286	*0.333*	*0.399*	*0.500*	**1.000**
added noise	*0.143*	0.458	0.667	0.649	0.904	**1.000**
post-processed data	0.167	**0.500**	**0.739**	**0.706**	**1.000**	**1.000**

5.2.2.2 Handling Noise and Sampling Rate with T_{min} of 2100 s

As *Figure 5.11* shows, the best result for T_{min} = 2100 s in terms of most valid results and highest median value is the post-processed original data, followed by the test sets with added noise. In terms of the percentage of valid results, the regular sampling rate of 60 s outperformed the other types of sampling rates. Compared to the pre-processed data with a T_{min} = 900 s, the median F-measure values are lower whereas the percentage of valid results stayed the same.

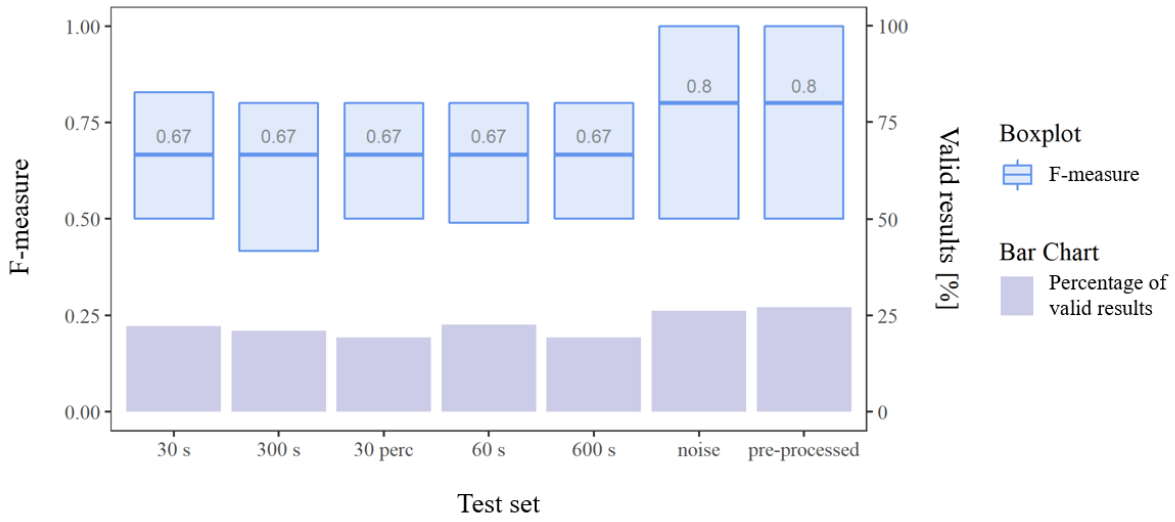


Figure 5.11 F-measure comparison of different test sets of MBGP algorithm [T_{min} = 2100 s]

Table 5.12 shows that the pre-processed original data have more low value outliers compared to the other test sets, as the minimal value of the pre-processed data is the lowest similar to the results with T_{min} = 900 s. However, the pre-processed original data performed best for most of the key distribution parameters.

Table 5.12 Boxplot statistics of different test sets of MBGP algorithm [T_{min} = 2100 s]

Test set	Min.	Q1	Median	Mean	Q3	Max.
regular sampling rate 30 s	0.118	**0.500**	*0.667*	0.645	0.829	**1.000**
regular sampling rate 300 s	0.125	*0.417*	*0.667*	*0.609*	*0.800*	**1.000**
randomly selected 30 %	0.133	**0.500**	*0.667*	0.621	*0.800*	**1.000**
regular sampling rate 60 s	0.118	0.490	*0.667*	0.634	*0.800*	**1.000**
regular sampling rate 600 s	0.133	**0.500**	*0.667*	0.621	*0.800*	**1.000**
added noise	**0.154**	**0.500**	**0.800**	0.714	**1.000**	**1.000**
pre-processed data	*0.105*	**0.500**	**0.800**	**0.738**	**1.000**	**1.000**

In *Figure 5.12*, one can see that the median F-measure values are lower for the post-processed data compared to the pre-processed data. The percentage of valid results increased for the post-processed original data and noise compared to their pre-processed equivalents, whereas the percentage of valid results decreased for the test sets with varying sampling rates. In opposition to the pre-processed results, the regular sampling rate of 30 s outperforms the regular sampling rate of 60 s in terms of the median F-measure value and the percentage of valid results. Compared to the results of T_{min} = 900 s the results for the varying sampling rates and added noise did not improve.

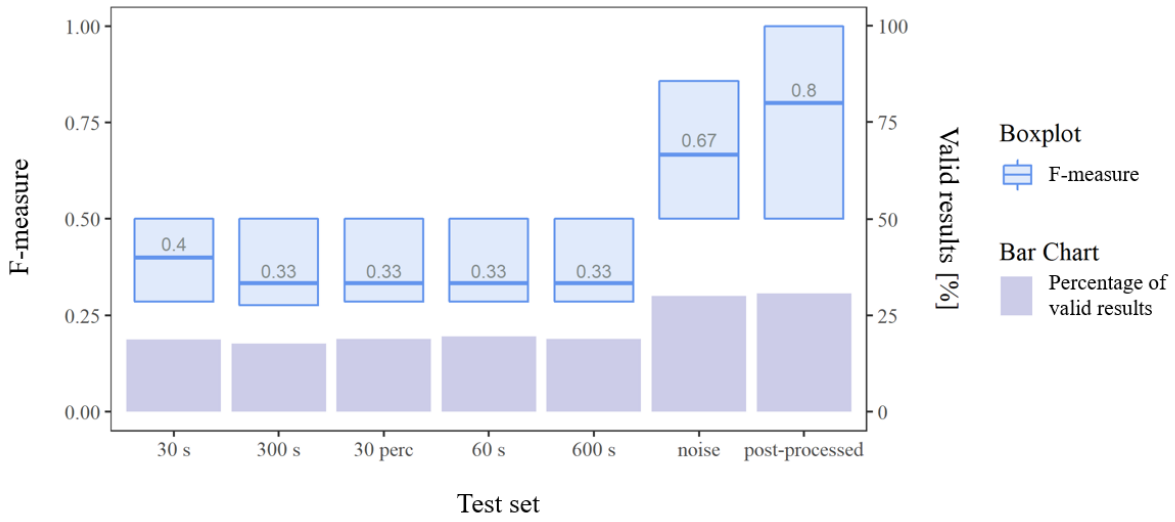


Figure 5.12 F-measure comparison of different test sets of post-processed MBGP algorithm [T_{min} = 2100 s]

Unlike the pre-processed data, the F-measure values of the test sets with the regular sampling rate of 30 s stayed not the same as the other test sets with varying sampling rates did (see *Table 5.13*). The third quartile and the minimum F-measure value are the same for all the different sampling rates.

Table 5.13 Boxplot statistics of different test sets of post-processed MBGP algorithm [T_{min} = 2100 s]

Test set	Min.	Q1	Median	Mean	Q3	Max.
regular sampling rate 30 s	**0.182**	0.286	0.400	0.444	*0.500*	**1.000**
regular sampling rate 300 s	**0.182**	*0.277*	*0.333*	0.398	*0.500*	**1.000**
randomly selected 30 %	**0.182**	0.286	*0.333*	*0.393*	*0.500*	**1.000**
regular sampling rate 60 s	**0.182**	0.286	*0.333*	0.419	*0.500*	**1.000**
regular sampling rate 600 s	**0.182**	0.286	*0.333*	*0.393*	*0.500*	**1.000**
added noise	*0.143*	**0.500**	0.667	0.658	0.857	**1.000**
post-processed data	0.167	**0.500**	**0.800**	**0.708**	**1.000**	**1.000**

5.2.3 Shape Measures

The shape measures were also applied to both threshold combinations. First, the combination 275/14400/900 is visualised and compared to the post-processed results and afterwards the results for the combination 275/14400/2100 are shown.

5.2.3.1 Shape Measures with T_{min} of 900 s

Unlike the CandidateStops algorithm, for all of the different test sets a shape index value could be determined (see *Figure 5.13*). The least compact shapes result out of the randomly selected 30 % sampling rate and the regular sampling rate of 600 s. The pre-processed original data has the most compact shapes. Compared to the CandidateStops algorithm, the shapes are more compact.

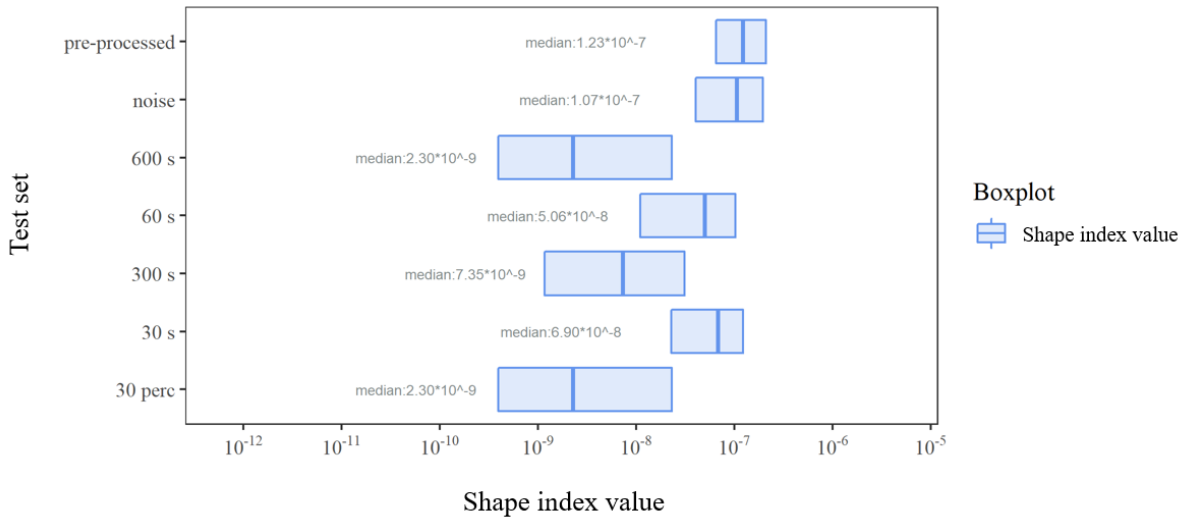


Figure 5.13 Comparison of shape index value of different test sets of MBGP algorithm [T_{min} = 900 s]

Except for the minimal value, the pre-processed original data has the highest key distribution parameters (see *Table 5.14*). The key distribution parameters of the regular sampling rate of 600 s and the randomly selected 30 % are equal and the lowest except for the minimal value. The manually labelled ground truth has fewer compact shapes than the only pre-processed MBGP algorithm.

Table 5.14 Boxplot statistics of shape index values of different test sets of MBGP algorithm [T_{min} = 900 s]

Test set	Min.	Q1	Median	Mean	Q3	Max.
pre-processed data	$2.76*10^{-11}$	**$6.55*10^{-08}$**	**$1.23*10^{-07}$**	**$1.77*10^{-07}$**	**$2.11*10^{-07}$**	**$5.35*10^{-06}$**
added noise	$2.74*10^{-11}$	$4.07*10^{-08}$	$1.07*10^{-07}$	$1.54*10^{-07}$	$1.97*10^{-07}$	$2.80*10^{-06}$
regular sampling rate 600 s	$1.56*10^{-12}$	*$3.97*10^{-10}$*	*$2.30*10^{-09}$*	*$1.75*10^{-08}$*	*$2.32*10^{-08}$*	*$1.99*10^{-07}$*
regular sampling rate 60 s	**$9.38*10^{-11}$**	$1.11*10^{-08}$	$5.06*10^{-08}$	$8.02*10^{-08}$	$1.02*10^{-07}$	$8.26*10^{-07}$
regular sampling rate 300 s	*$5.77*10^{-13}$*	$1.18*10^{-09}$	$7.35*10^{-09}$	$2.86*10^{-08}$	$3.14*10^{-08}$	$4.62*10^{-07}$
regular sampling rate 30 s	*$5.77*10^{-13}$*	$2.30*10^{-08}$	$6.90*10^{-08}$	$1.03*10^{-07}$	$1.23*10^{-07}$	$7.46*10^{-07}$
randomly selected 30 %	$1.56*10^{-12}$	*$3.97*10^{-10}$*	*$2.30*10^{-09}$*	*$1.75*10^{-08}$*	*$2.32*10^{-08}$*	*$1.99*10^{-07}$*
manually labelled ground truth	$1.99*10^{-11}$	$6.72*10^{-10}$	$3.08*10^{-09}$	$6.97*10^{-08}$	$7.79*10^{-09}$	$8.71*10^{-07}$

The post-processed data have, in general, more compact shapes than the pre-processed test sets. In particular, the varying sampling rates got higher median shape index values whereas the median shape index values of the post-processed original data and the post-processed test sets with added noise only slightly increased. As *Figure 5.14* shows, the results are rather wide spread for the different varying sampling rates, especially for the sampling rate of 600 s.

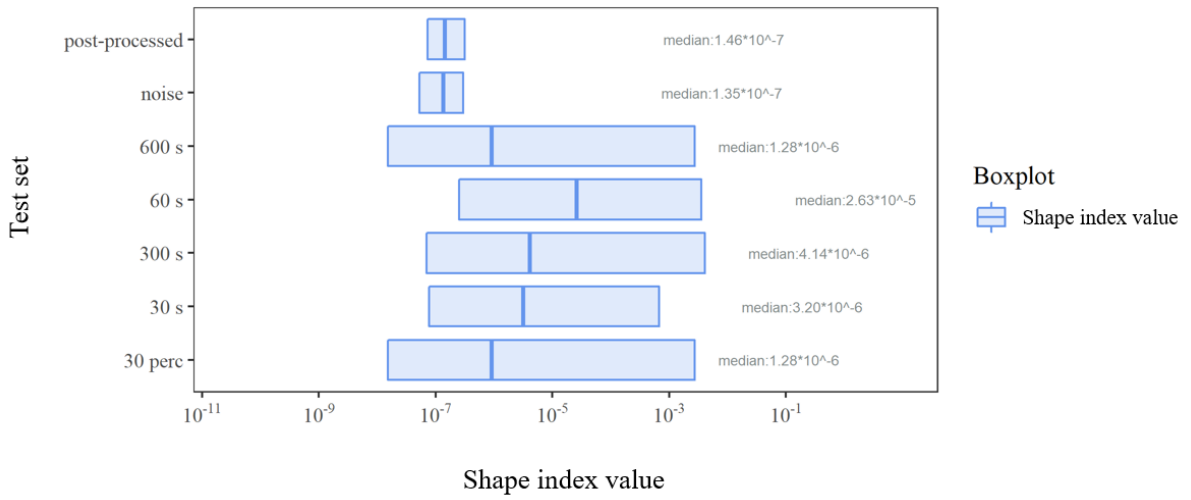


Figure 5.14 Comparison of shape index value of different test sets of post-processed MBGP algorithm [T_{min} = 900 s]

Unlike the results of the pre-processed data the original data and test sets with added noise have the lowest key distribution parameters as *Table 5.15* shows. The manually labelled ground truth has fewer compact shapes compared to the post-processed MBGP algorithm.

Furthermore, the different sampling rates have the same maximum values but have very different interquartile ranges that are wider compared to their pre-processed equivalents. The shapes of the post-processed data with varying sampling rates are less homogeneous than the shapes of the pre-processed data.

Table 5.15 Boxplot statistics of shape index values of different test sets of post-processed MBGP algorithm [T_{min} = 900 s]

Test set	Min.	Q1	Median	Mean	Q3	Max.
post-processed data	*2.76*10^{-11}*	7.29*10^{-08}	1.46*10^{-07}	9.30*10^{-02}	3.18*10^{-07}	*8.62*10^{00}*
added noise	2.74*10^{-11}	5.24*10^{-08}	*1.35*10^{-07}*	*8.11*10^{-02}*	*2.94*10^{-07}*	*8.62*10^{00}*
regular sampling rate 600 s	8.74*10^{-11}	*1.52*10^{-08}*	1.28*10^{-06}	4.79*10^{-01}	2.72*10^{-03}	**1.06*10^{01}**
regular sampling rate 60 s	9.01*10^{-10}	**2.53*10^{-07}**	**2.63*10^{-05}**	4.23*10^{-01}	3.56*10^{-03}	**1.06*10^{01}**
regular sampling rate 300 s	**1.12*10^{-09}**	6.96*10^{-08}	4.14*10^{-06}	**4.99*10^{-01}**	**4.12*10^{-03}**	**1.06*10^{01}**
regular sampling rate 30 s	4.90*10^{-10}	7.87*10^{-08}	3.20*10^{-06}	2.65*10^{-01}	6.84*10^{-04}	**1.06*10^{01}**
randomly selected 30 %	8.74*10^{-11}	*1.52*10^{-08}*	1.28*10^{-06}	4.79*10^{-01}	2.72*10^{-03}	**1.06*10^{01}**
manually labelled ground truth	1.99*10^{-11}	6.72*10^{-10}	3.08*10^{-09}	6.97*10^{-08}	7.79*10^{-09}	8.71*10^{-07}

5.2.3.2 Shape Measures with T_{min} of 2100 s

As *Figure 5.15* shows, the least compact shapes result from the randomly selected 30 % sampling rate and the regular sampling rate of 600 s. The pre-processed original data has the most compact and most homogeneous shapes. Compared to the results of T_{min} = 900 s, the boxplot patterns and the median shape index values look similar.

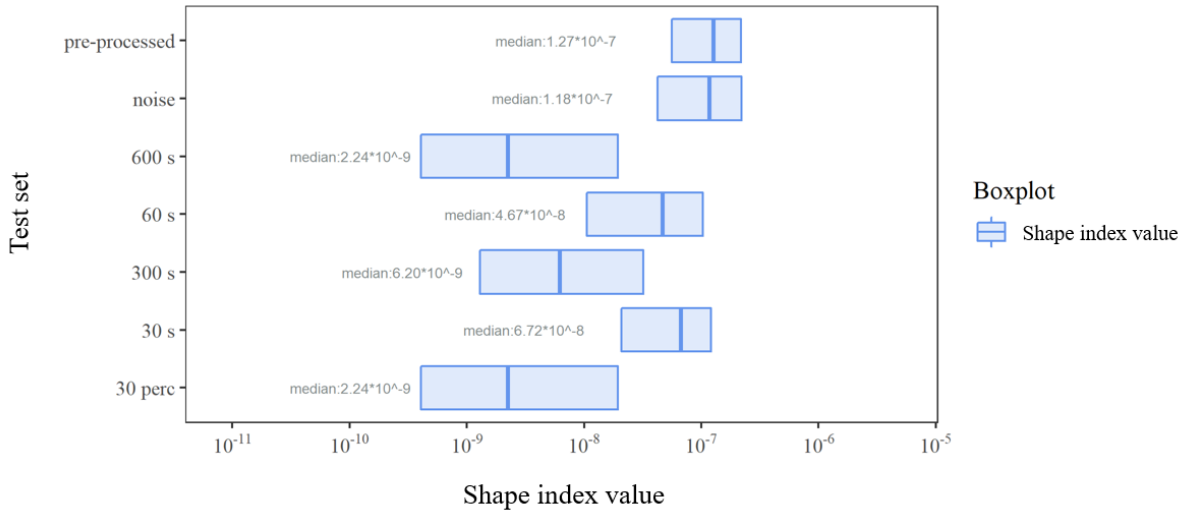


Figure 5.15 Comparison of shape index value of different test sets of MBGP algorithm [T_{min} = 2100 s]

Except for the minimal value and third quartile, the pre-processed original data have the highest key distribution parameters (see *Table 5.16*). The key distribution parameters of the regular sampling rate of 600 s and the randomly selected 30 % are equal and the lowest. The manually labelled ground truth has fewer compact shapes than the only pre-processed MBGP algorithm.

Table 5.16 Boxplot statistics of shape index values of different test sets of MBGP algorithm [T_{min} = 2100 s]

Test set	Min.	Q1	Median	Mean	Q3	Max.
pre-processed data	$2.76*10^{-11}$	**$5.58*10^{-08}$**	**$1.27*10^{-07}$**	**$1.79*10^{-07}$**	$2.18*10^{-07}$	**$5.35*10^{-06}$**
added noise	$2.74*10^{-11}$	$4.25*10^{-08}$	$1.18*10^{-07}$	$1.58*10^{-07}$	$2.20*10^{-07}$	$9.19*10^{-07}$
regular sampling rate 600 s	*$7.80*10^{-12}$*	*$4.08*10^{-10}$*	*$2.24*10^{-09}$*	*$1.68*10^{-08}$*	*$1.94*10^{-08}$*	*$1.98*10^{-07}$*
regular sampling rate 60 s	**$9.38*10^{-11}$**	$1.05*10^{-08}$	$4.67*10^{-08}$	$7.58*10^{-08}$	$1.03*10^{-07}$	$6.04*10^{-07}$
regular sampling rate 300 s	$9.04*10^{-12}$	$1.30*10^{-09}$	$6.20*10^{-09}$	$3.00*10^{-08}$	$3.22*10^{-08}$	$4.62*10^{-07}$
regular sampling rate 30 s	$8.71*10^{-11}$	$2.08*10^{-08}$	$6.72*10^{-08}$	$1.02*10^{-07}$	**$1.21*10^{-07}$**	$7.46*10^{-07}$
randomly selected 30 %	*$7.80*10^{-12}$*	*$4.08*10^{-10}$*	*$2.24*10^{-09}$*	*$1.68*10^{-08}$*	*$1.94*10^{-08}$*	*$1.98*10^{-07}$*
manually labelled ground truth	$1.99*10^{-11}$	$6.72*10^{-10}$	$3.08*10^{-09}$	$6.97*10^{-08}$	$7.79*10^{-09}$	$8.71*10^{-07}$

The post-processed data have in general more compact shapes than the pre-processed ones. Specifically, the varying sampling rates got higher median shape index values whereas the median shape index values of the post-processed original data and the post-processed test sets with added noise only slightly

increased. As *Figure 5.16* shows, the data are rather wide-spread for the different varying sampling rates, especially for the sampling rate of 600 s and the randomly selected 30 %.

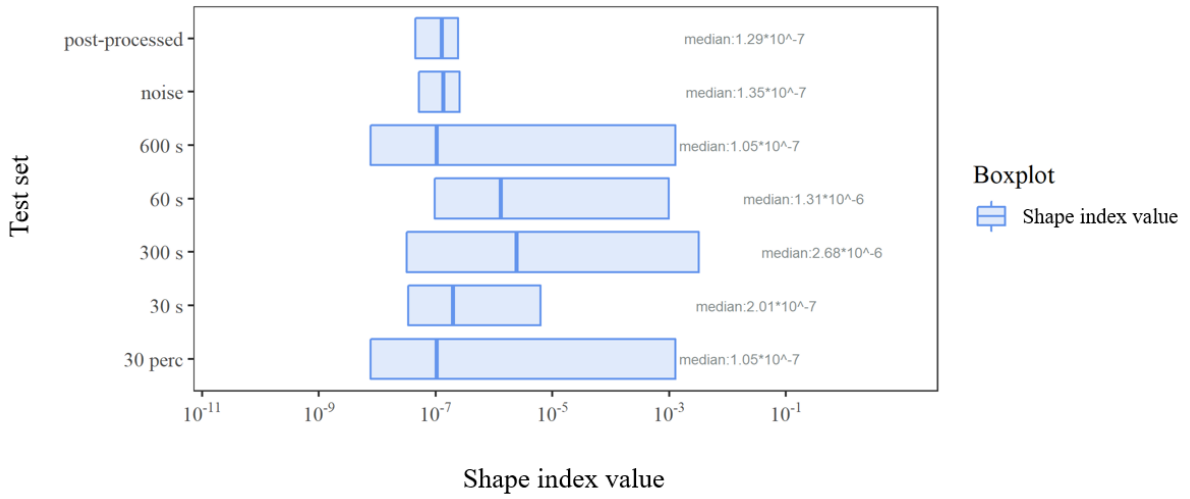


Figure 5.16 Comparison of shape index value of different test sets of post-processed MBGP algorithm [T_{min} = 2100 s]

Unlike the results of the post-processed data with T_{min} = 900 s, test sets with a regular sampling rate of 600 s and randomly selected 30 % have the lowest key distribution parameters for the first quartile and median value as can be seen in *Table 5.17*. Furthermore, the different sampling rates have the same maximum values but have very different interquartile ranges that are wider compared to their pre-processed equivalents. The manually labelled ground truth has fewer compact shapes compared to the post-processed MBGP algorithm.

Table 5.17 Boxplot statistics of shape index values of different test sets of post-processed MBGP algorithm [T_{min} = 2100 s]

Test set	Min.	Q1	Median	Mean	Q3	Max.
post-processed data	$3.39*10^{-10}$	$4.48*10^{-08}$	$1.29*10^{-07}$	*$2.21*10^{-07}$*	*$2.45*10^{-07}$*	*$4.66*10^{-06}$*
added noise	*$2.74*10^{-11}$*	$5.20*10^{-08}$	$1.35*10^{-07}$	$8.45*10^{-02}$	$2.59*10^{-07}$	$8.62*10^{00}$
regular sampling rate 600 s	$8.74*10^{-11}$	*$7.87*10^{-09}$*	*$1.05*10^{-07}$*	$4.67*10^{-01}$	$1.28*10^{-03}$	**$1.06*10^{01}$**
regular sampling rate 60 s	$4.78*10^{-10}$	**$9.72*10^{-08}$**	$1.31*10^{-06}$	$3.49*10^{-01}$	$9.85*10^{-04}$	**$1.06*10^{01}$**
regular sampling rate 300 s	**$5.51*10^{-10}$**	$3.21*10^{-08}$	**$2.68*10^{-06}$**	**$4.90*10^{-01}$**	**$3.24*10^{-03}$**	**$1.06*10^{01}$**
regular sampling rate 30 s	$4.90*10^{-10}$	$3.40*10^{-08}$	$2.01*10^{-07}$	$2.01*10^{-01}$	$6.27*10^{-06}$	**$1.06*10^{01}$**
randomly selected 30 %	$8.74*10^{-11}$	*$7.87*10^{-09}$*	*$1.05*10^{-07}$*	$4.67*10^{-01}$	$1.28*10^{-03}$	**$1.06*10^{01}$**
manually labelled ground truth	$1.99*10^{-11}$	$6.72*10^{-10}$	$3.08*10^{-09}$	$6.97*10^{-08}$	$7.79*10^{-09}$	$8.71*10^{-07}$

5.3 POSMIT

The POSMIT algorithm had to be run 24 times for each test set (i.e., 2160 times for the original test sets without the varied sampling rate and added noise) in order to build all threshold combinations as it needs three input parameters, of which only two, search bandwidth (h_i) and stop probability (ε), needed to be iteratively tested.

5.3.1 Threshold Selection

As *Figure 5.17* shows, the POSMIT algorithm has a high percentage of valid results for the search bandwidth of 1 and a stop probability of 0.25 as well as for the calculated search bandwidth and stop probability of 0.25. Furthermore, the median F-measure values were highest for those two combinations. The other threshold combinations had very low percentages of valid results, but the respective median F-measure values varied from 0.04 to 0.37.

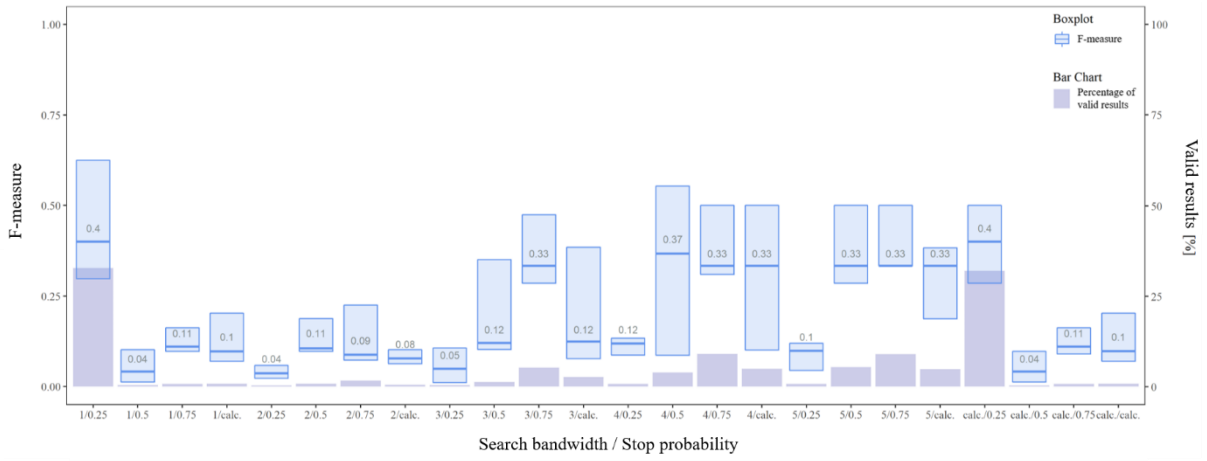


Figure 5.17 F-measure comparison of different threshold combinations of POSMIT algorithm

The results displayed in *Table 5.18* give a more detailed overview of the boxplot statistics. The boxplot statistics show that the F-measure values are highest for the threshold combination 1/0.25. As it also has the highest percentage of valid results, this combination was chosen for further analysis.

Table 5.18 Boxplot statistics of POSMIT algorithm

Thresholds h_i/ε	Min.	Q1	Median	Mean	Q3	Max.
1/0.25	**0.182**	0.298	**0.400**	**0.475**	**0.625**	**1.000**
1/0.5	0.004	0.012	0.041	0.055	0.102	0.120
1/0.75	0.068	0.097	0.110	0.134	0.162	0.235
1/calc.	0.010	0.070	0.097	0.123	0.203	0.237
2/0.25	*0.002*	0.023	*0.037*	*0.048*	*0.058*	0.129
2/0.5	0.063	0.097	0.105	0.138	0.188	0.235
2/0.75	0.027	0.073	0.088	0.216	0.225	0.800
2/calc.	0.009	0.063	0.078	0.075	0.102	*0.117*
3/0.25	0.003	*0.011*	0.049	0.058	0.107	0.124
3/0.5	0.063	0.102	0.120	0.227	0.351	0.500
3/0.75	0.040	0.286	0.333	0.336	0.475	0.667
3/calc.	0.008	0.077	0.124	0.268	0.385	0.800
4/0.25	0.033	0.087	0.119	0.127	0.133	0.264
4/0.5	0.021	0.086	0.367	0.348	0.554	0.750
4/0.75	0.040	0.310	0.333	0.428	0.500	**1.000**
4/calc.	0.008	0.101	0.333	0.338	0.500	0.750
5/0.25	0.039	0.044	0.098	0.087	0.119	0.144
5/0.5	0.057	0.286	0.333	0.372	0.500	0.667
5/0.75	0.049	**0.333**	0.333	0.447	0.500	**1.000**
5/calc.	0.003	0.187	0.333	0.311	0.383	0.667
calc./0.25	**0.182**	0.286	**0.400**	0.472	0.500	**1.000**
calc./0.5	0.004	0.012	0.041	0.054	0.097	0.120
calc./0.75	0.068	0.090	0.110	0.133	0.162	0.235
calc./calc.	0.010	0.070	0.098	0.124	0.203	0.237

The post-processed median F-measure values for the combination 1/0.25 and the calculated search bandwidth/stop probability of 0.25 stayed the same compared to the only pre-processed results. However, the other median F-measure values, as well as the percentage of valid results increased as *Figure 5.18* shows. The percentage of valid results stayed the same for the chosen threshold 1/0.25.

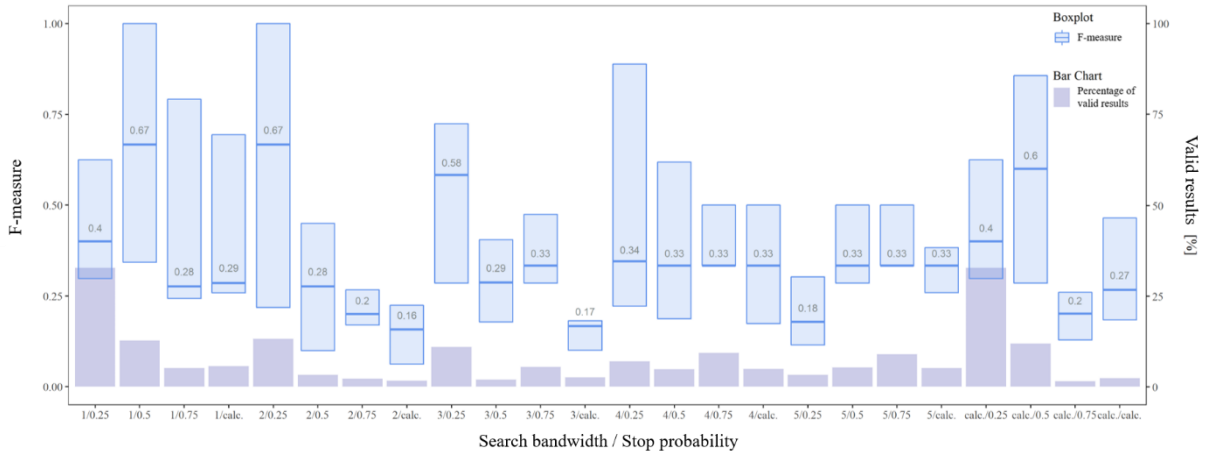


Figure 5.18 F-measure comparison of different threshold combinations of post-processed POSMIT algorithm

By looking only at the boxplot statistics in *Table 5.19*, the threshold combination 1/0.5 seemed to deliver the best results. However, as the percentage of valid results was higher for the threshold combination 1/0.25 this result was chosen. This result was chosen over the calculated/0.25 threshold combination (both combinations had the exact same results) because knowing the parameter value is more transparent and makes interpretations easier compared to not knowing the calculated search bandwidth value.

Table 5.19 Boxplot statistics of post-processed POSMIT algorithm

Thresholds h_i/ε	Min.	Q1	Median	Mean	Q3	Max.
1/0.25	**0.182**	0.298	0.400	0.475	0.625	**1.000**
1/0.5	0.036	**0.343**	**0.667**	**0.602**	**1.000**	**1.000**
1/0.75	0.053	0.243	0.276	0.460	0.792	**1.000**
1/calc.	0.063	0.258	0.286	0.462	0.694	**1.000**
2/0.25	*0.035*	0.218	**0.667**	0.564	**1.000**	**1.000**
2/0.5	0.043	0.100	0.276	0.371	0.449	**1.000**
2/0.75	0.091	0.170	0.200	0.281	0.267	0.800
2/calc.	0.038	*0.062*	*0.158*	*0.189*	0.224	0.522
3/0.25	0.036	0.286	0.583	0.547	0.724	**1.000**
3/0.5	0.098	0.178	0.287	0.293	0.405	*0.500*
3/0.75	0.121	0.286	0.333	0.352	0.475	0.667
3/calc.	0.056	0.100	0.167	0.259	*0.182*	0.800
4/0.25	0.074	0.222	0.345	0.487	0.889	**1.000**
4/0.5	0.091	0.187	0.333	0.392	0.619	0.750
4/0.75	0.138	0.333	0.333	0.441	0.500	**1.000**
4/calc.	0.069	0.174	0.333	0.340	0.500	0.667
5/0.25	0.050	0.115	0.179	0.292	0.302	**1.000**
5/0.5	0.105	0.286	0.333	0.367	0.500	0.667
5/0.75	0.143	0.333	0.333	0.448	0.500	**1.000**
5/calc.	0.043	0.259	0.333	0.331	0.383	0.667
calc./0.25	**0.182**	0.298	0.400	0.475	0.625	**1.000**
calc./0.5	0.036	0.286	0.600	0.561	0.857	**1.000**
calc./0.75	0.053	0.129	0.201	0.223	0.260	*0.500*
calc./calc.	0.063	0.184	0.267	0.304	0.464	*0.500*

5.3.2 Handling Noise and Sampling Rate

After the threshold combination 1/0.25 was chosen, the algorithm was applied to the test sets with added noise and varying sampling rates. As *Figure 5.19* shows, the best results in terms of most valid results and highest median value is the test sets with added noise followed by the pre-processed original data. The worst results in terms of percentage of valid results delivered the regular sampling rate of 600 s and the sampling rate of randomly selected 30 %.

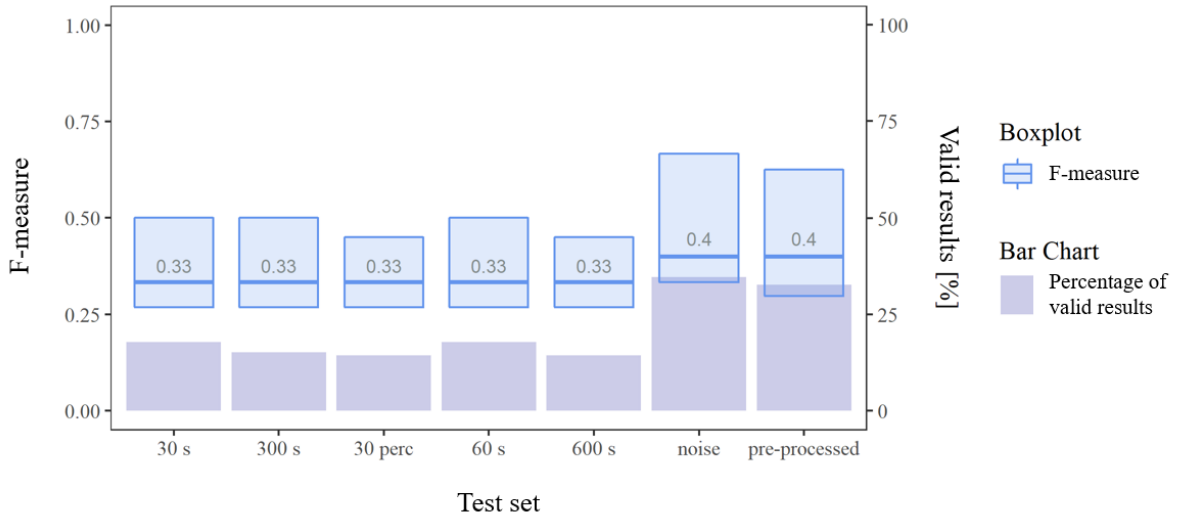


Figure 5.19 F-measure comparison of different test sets of POSMIT algorithm

Similar to the CandidateStops algorithm, the test sets with added noise have the highest key distribution parameters as can be seen in *Table 5.20*. Furthermore, the regular sampling rate of 600 s and the sampling rate of randomly selected 30 % show the same results. All of the different test sets have the same minimal values and the different sampling rates have the same first quartile and median values.

Table 5.20 Boxplot statistics of different test sets of POSMIT algorithm

Test set	Min.	Q1	Median	Mean	Q3	Max.
regular sampling rate 30 s	**0.182**	*0.268*	*0.333*	0.410	0.500	**1.000**
regular sampling rate 300 s	**0.182**	*0.268*	*0.333*	0.391	0.500	**1.000**
randomly selected 30 %	**0.182**	*0.268*	*0.333*	*0.369*	*0.450*	*0.667*
regular sampling rate 60 s	**0.182**	*0.268*	*0.333*	0.410	0.500	**1.000**
regular sampling rate 600 s	**0.182**	*0.268*	*0.333*	*0.369*	*0.450*	*0.667*
added noise	**0.182**	**0.333**	**0.400**	**0.511**	**0.667**	**1.000**
pre-processed data	**0.182**	0.298	**0.400**	0.475	0.625	**1.000**

Figure 5.20 shows that the percentage of valid results decreased for the test sets with added noise compared to the pre-processed test sets with added noise (i.e., now same percentage as post-processed original data). Furthermore, the median F-measure values stayed the same for all the different test sets.

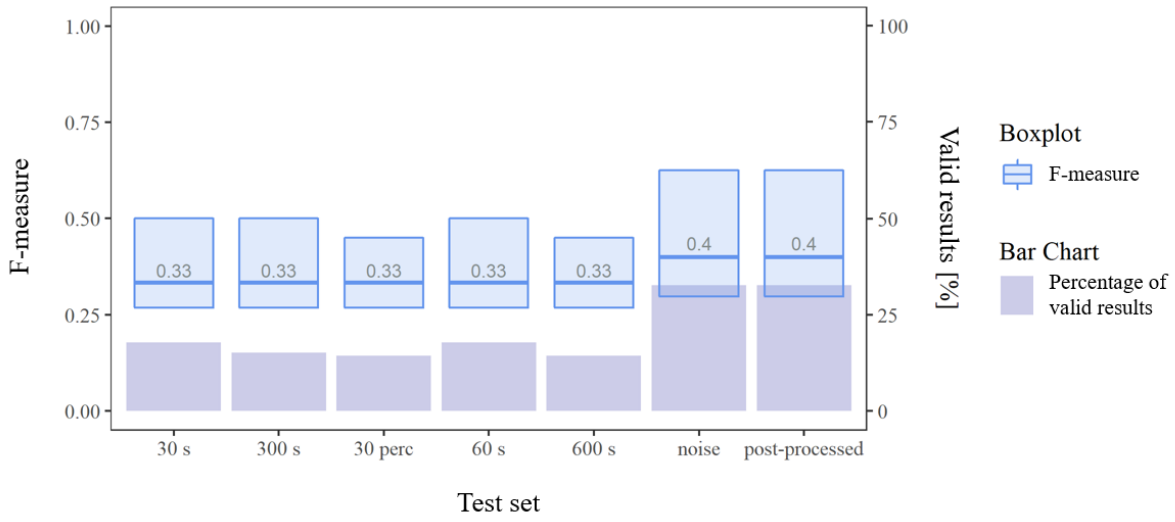


Figure 5.20 F-measure comparison of different test sets of post-processed POSMIT algorithm

Table 5.21 shows that the values only changed for the added noise and post-processed original test sets compared to the pre-processed test sets. The values of added noise decreased and are identical to the post-processed original data values.

Table 5.21 Boxplot statistics of different test sets of post-processed POSMIT algorithm

Test set	Min.	Q1	Median	Mean	Q3	Max.
regular sampling rate 30 s	**0.182**	*0.268*	*0.333*	0.410	0.500	**1.000**
regular sampling rate 300 s	**0.182**	*0.268*	*0.333*	0.391	0.500	**1.000**
randomly selected 30 %	**0.182**	*0.268*	*0.333*	*0.369*	*0.450*	*0.667*
regular sampling rate 60 s	**0.182**	*0.268*	*0.333*	0.410	0.500	**1.000**
regular sampling rate 600 s	**0.182**	*0.268*	*0.333*	*0.369*	*0.450*	*0.667*
added noise	**0.182**	**0.298**	**0.400**	**0.475**	**0.625**	**1.000**
post-processed data	**0.182**	**0.298**	**0.400**	**0.475**	**0.625**	**1.000**

5.3.3 Shape Measures

Opposite to the CandidateStops algorithm, shape index values could be determined for all test sets of the POSMIT algorithm (see *Figure 5.21*). The median shape index values and the interquartile ranges of the different test sets are close together. The median shape index values are highest compared to the other two algorithms, CandidateStops and MBGP.

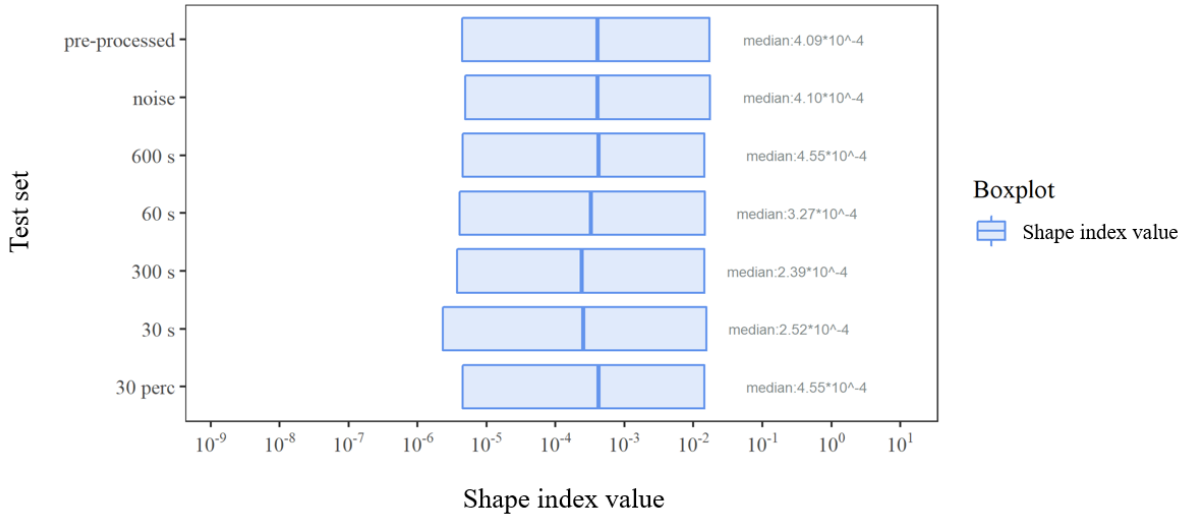


Figure 5.21 Comparison of shape index value of different test sets of POSMIT algorithm

As *Table 5.22* shows, the least compact shape was detected by the regular sampling rate of 300 s followed by the regular sampling rate of 30 s. The test sets with added noise have the highest key distribution parameters except for the median value. The manually labelled ground truth has fewer compact shapes compared to the POSMIT algorithm.

Table 5.22 Boxplot statistics of shape index values of different test sets of POSMIT algorithm

Test set	Min.	Q1	Median	Mean	Q3	Max.
pre-processed data	$1.39*10^{-08}$	$4.43*10^{-06}$	$4.09*10^{-04}$	**$7.48*10^{-01}$**	$1.72*10^{-02}$	**$1.13*10^{01}$**
added noise	**$1.42*10^{-08}$**	**$4.93*10^{-06}$**	$4.10*10^{-04}$	**$7.48*10^{-01}$**	**$1.73*10^{-02}$**	**$1.13*10^{01}$**
regular sampling rate 600 s	$1.61*10^{-09}$	$4.55*10^{-06}$	**$4.55*10^{-04}$**	$6.36*10^{-01}$	*$1.44*10^{-02}$*	*$1.07*10^{01}$*
regular sampling rate 60 s	$6.26*10^{-09}$	$4.10*10^{-06}$	$3.27*10^{-04}$	$6.05*10^{-01}$	$1.47*10^{-02}$	$1.12*10^{01}$
regular sampling rate 300 s	*$1.36*10^{-09}$*	$3.75*10^{-06}$	*$2.39*10^{-04}$*	$6.00*10^{-01}$	$1.45*10^{-02}$	$1.11*10^{01}$
regular sampling rate 30 s	$2.99*10^{-09}$	*$2.32*10^{-06}$*	$2.52*10^{-04}$	*$5.34*10^{-01}$*	$1.55*10^{-02}$	**$1.13*10^{01}$**
randomly selected 30 %	$1.61*10^{-09}$	$4.55*10^{-06}$	**$4.55*10^{-04}$**	$6.36*10^{-01}$	*$1.44*10^{-02}$*	*$1.07*10^{01}$*
manually labelled ground truth	$1.99*10^{-11}$	$6.72*10^{-10}$	$3.08*10^{-09}$	$6.97*10^{-08}$	$7.79*10^{-09}$	$8.71*10^{-07}$

The post-processed data show the same results in *Figure 5.22* as the pre-processed data in *Figure 5.21*. Furthermore, boxplot statistics were also identical and were therefore not included in the thesis. Reasons for the same results are discussed in *Chapter 6*.

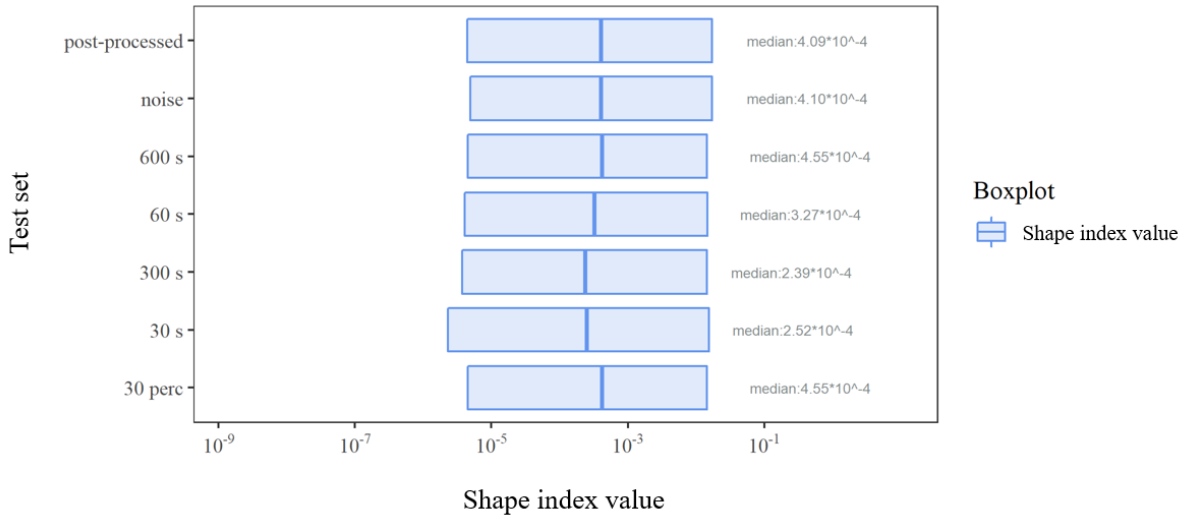


Figure 5.22 Comparison of shape index value of different test sets of post-processed POSMIT algorithm

5.4 SOC

The SOC algorithm was planned to be run 28 times for each testing day in order to build all threshold combinations. However, it was not able to find stops for *Eps* values other than 75 m. Therefore, the algorithm only had to run seven times in order to find the optimal threshold *Tau*.

5.4.1 Threshold Selection

As *Figure 5.23* shows, the median F-measure values were the same for all *Tau* values. By looking at the percentage of valid results, the *Tau* value of 1200 s and 1800 s had the same percentage of valid results.

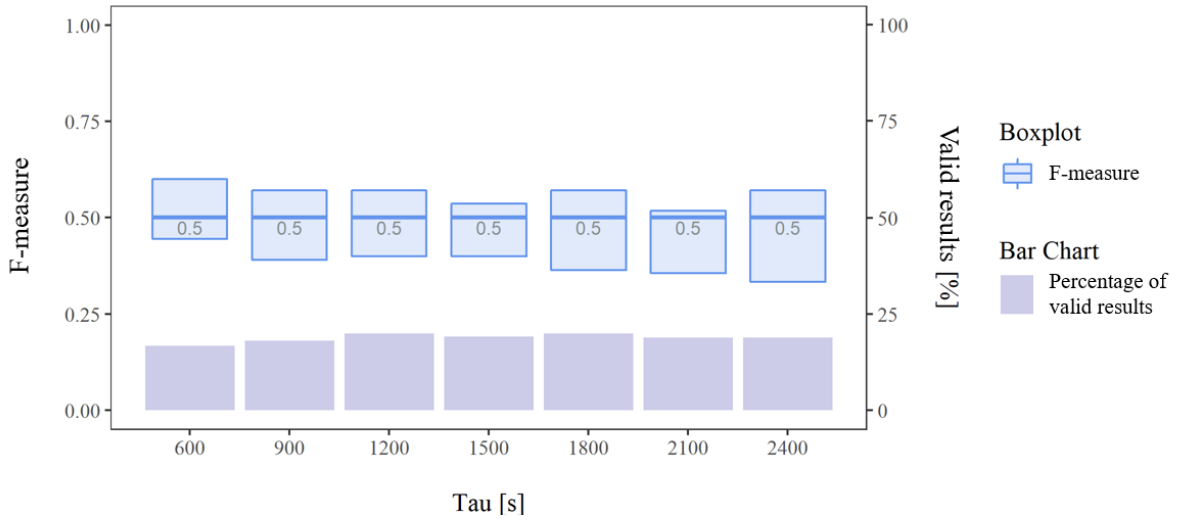


Figure 5.23 F-measure comparison of different Tau thresholds of SOC algorithm

To choose the most suitable threshold, *Table 5.23* gave an overview of the boxplot statistics. Based on the key distribution parameters a *Tau* value of 600 s seemed most suitable. However, for this value, the percentage of valid results was lowest. By looking at the two results where the percentage of valid results was highest, a *Tau* value of 1200 s had a higher first quartile value and mean value than the *Tau* value of 1800 s. In general, the percentage of valid results are second highest after the MBGP algorithm, compared to the other three algorithms.

Table 5.23 Boxplot statistics of SOC algorithm

Tau [s]	Min.	Q1	Median	Mean	Q3	Max.
600	**0.182**	**0.444**	**0.500**	**0.520**	**0.600**	**1.000**
900	**0.182**	0.391	**0.500**	0.508	0.571	**1.000**
1200	**0.182**	0.400	**0.500**	0.498	0.571	**1.000**
1500	**0.182**	0.400	**0.500**	0.492	0.536	**1.000**
1800	**0.182**	0.364	**0.500**	0.485	0.571	**1.000**
2100	*0.167*	0.356	**0.500**	0.470	*0.518*	**1.000**
2400	*0.167*	*0.333*	**0.500**	*0.460*	0.571	**1.000**

By looking at the results of the post-processed SOC algorithm in *Figure 5.2*, one can see that the median F-measure values stayed the same or decreased in case of a *Tau* value of 2100 s. The third quartiles of the boxplots seemed to have decreased compared to the pre-processed results. The percentage of valid results stayed the same or increased slightly in case of Tau = 1500 s.

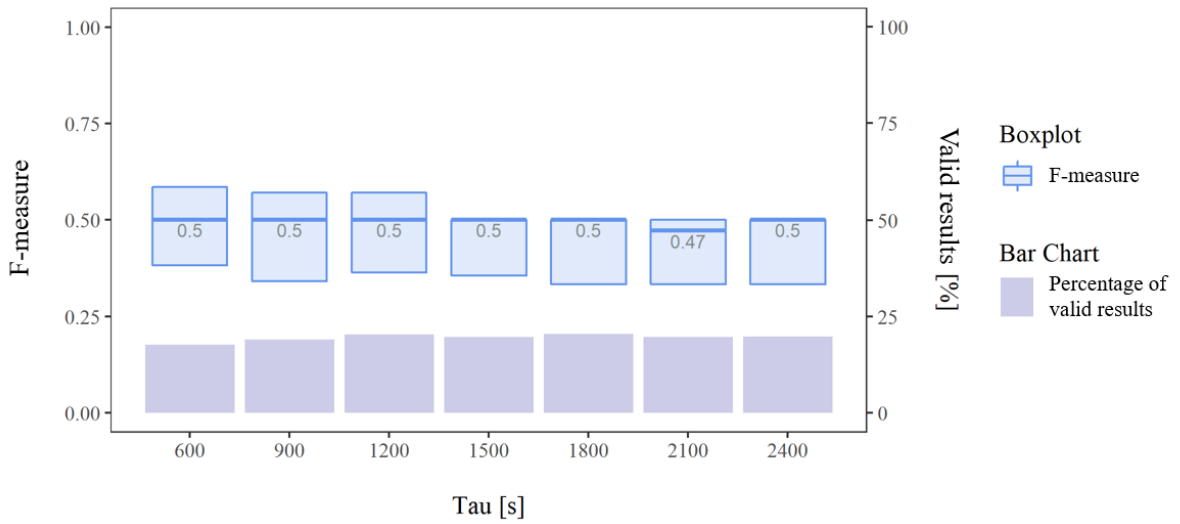


Figure 5.24 F-measure comparison of different Tau thresholds of post-processed SOC algorithm

According to *Table 5.24* a *Tau* value of 600 s delivered the best key distribution parameters. However, as the percentage of valid results was lowest for this *Tau* value, a *Tau* value of 1200 s was chosen for the post-processed results (same as for the pre-processed results) as it had the second highest key distribution parameters of the boxplot statistics.

Table 5.24 Boxplot statistics of post-processed SOC algorithm

Tau [s]	Min.	Q1	Median	Mean	Q3	Max.
600	**0.182**	**0.382**	**0.500**	**0.512**	**0.586**	**1.000**
900	**0.182**	0.341	**0.500**	0.501	0.571	**1.000**
1200	**0.182**	0.364	**0.500**	0.496	0.571	**1.000**
1500	**0.182**	0.356	**0.500**	0.490	*0.500*	**1.000**
1800	**0.182**	*0.333*	**0.500**	0.483	*0.500*	**1.000**
2100	*0.167*	*0.333*	*0.472*	0.466	*0.500*	**1.000**
2400	*0.167*	*0.333*	**0.500**	*0.457*	*0.500*	**1.000**

5.4.2 Handling Noise and Sampling Rate

After the *Tau* value of 1200 s was chosen, the algorithm was applied to the test sets with added noise and varying sampling rates. As *Figure 5.25* shows, the best results in terms of most valid results and highest median value are the pre-processed original data followed by the test sets with added noise. These results are similar to the results of POSMIT and MBGP algorithm. The different varying sampling rates have the similar percentage of valid results.

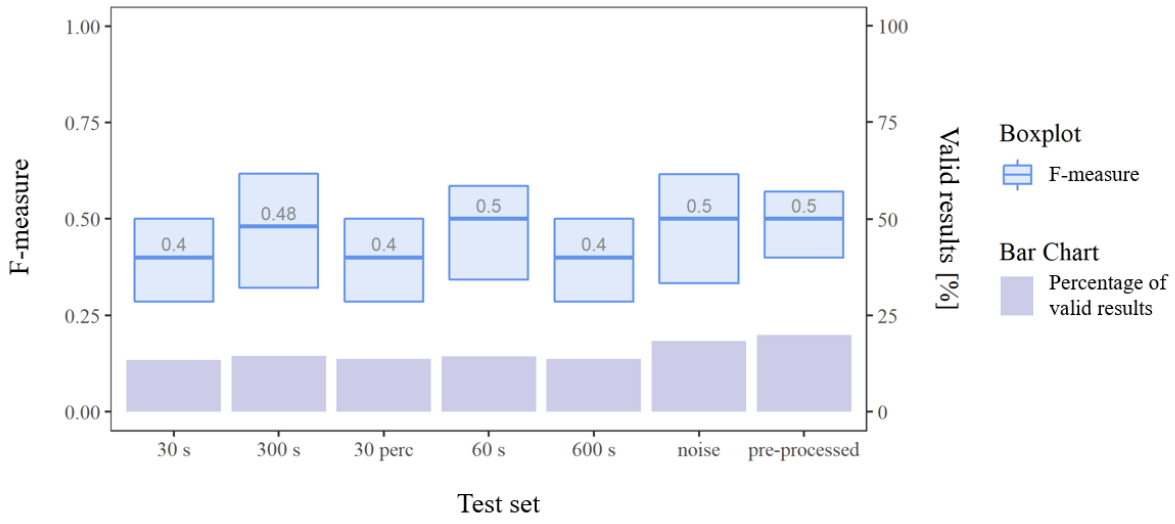


Figure 5.25 F-measure comparison of different test sets of SOC algorithm

Table 5.25 confirms that the test sets with added noise and the pre-processed original data have the highest or second highest key distribution parameters except for the third quartile value that was highest at the regular sampling rate of 300 s. The test sets with added noise and the pre-processed original data had maximum values of 1 which means values of perfect precision and recall.

Table 5.25 Boxplot statistics of different test sets of SOC algorithm

Test set	Min.	Q1	Median	Mean	Q3	Max.
regular sampling rate 30 s	**0.222**	*0.286*	*0.400*	0.417	*0.500*	0.857
regular sampling rate 300 s	0.154	0.321	0.481	0.465	**0.617**	*0.800*
randomly selected 30 %	*0.143*	*0.286*	*0.400*	*0.410*	*0.500*	*0.800*
regular sampling rate 60 s	0.182	0.343	**0.500**	0.477	0.586	0.857
regular sampling rate 600 s	*0.143*	*0.286*	*0.400*	*0.410*	*0.500*	*0.800*
added noise	0.182	0.333	**0.500**	**0.500**	0.615	**1.000**
pre-processed data	0.182	**0.400**	**0.500**	0.498	0.571	**1.000**

Compared to the pre-processed data, the F-measure median value decreased for all of the different varying sampling rates except for the regular sampling rate of 30 s. The percentage of valid results also decreased for the different varying sampling rates. They stayed the same for the test sets with added noise and the post-processed original data.

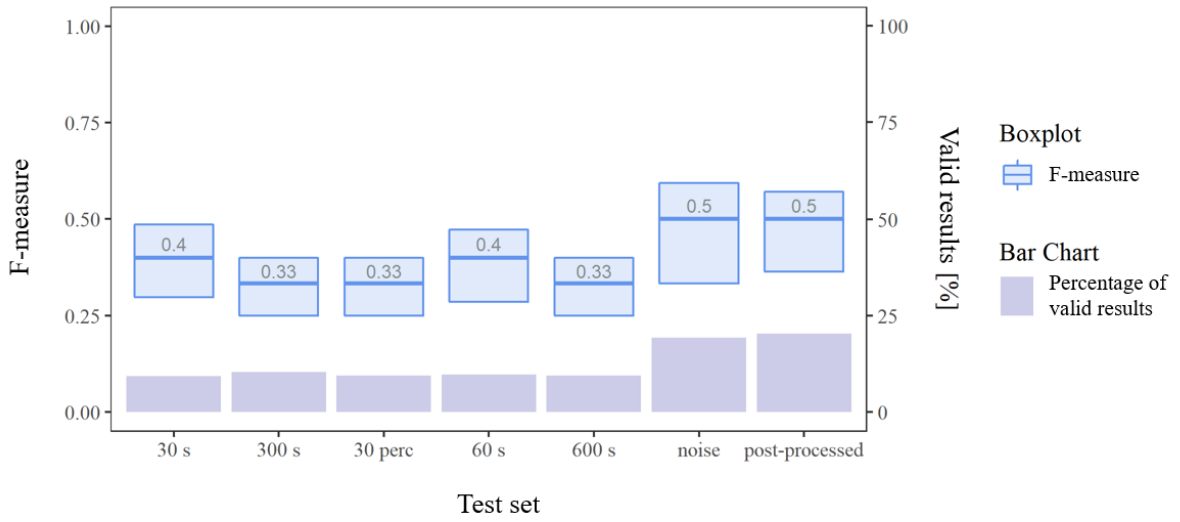


Figure 5.26 F-measure comparison of different test sets of post-processed SOC algorithm

Figure 5.26 shows, that the test sets with added noise had the highest key distribution parameters followed by the post-processed original data. The sampling rate of randomly selected 30 % and the regular sampling rate of 600 s had the worst key distribution parameters.

Table 5.26 Boxplot statistics of different test sets of post-processed SOC algorithm

Test set	Min.	Q1	Median	Mean	Q3	Max.
regular sampling rate 30 s	**0.182**	0.298	0.400	0.379	0.486	*0.500*
regular sampling rate 300 s	**0.182**	*0.250*	*0.333*	0.345	*0.400*	*0.500*
randomly selected 30 %	**0.182**	*0.250*	*0.333*	*0.341*	*0.400*	*0.500*
regular sampling rate 60 s	**0.182**	0.286	0.400	0.378	0.472	0.667
regular sampling rate 600 s	**0.182**	*0.250*	*0.333*	*0.341*	*0.400*	*0.500*
added noise	**0.182**	**0.333**	**0.500**	**0.495**	**0.593**	**1.000**
post-processed data	**0.182**	0.364	**0.500**	0.496	0.571	**1.000**

5.4.3 Shape Measures

Compared to the POSMIT algorithm, the shape index values were lower and more similar to the shape index values of the MBGP algorithm (see *Figure 5.27*). The regular sampling rate of 600 s and the sampling rate of randomly selected 30 % have fewer compact shapes than the test sets with added noise, which have the most compact shapes.

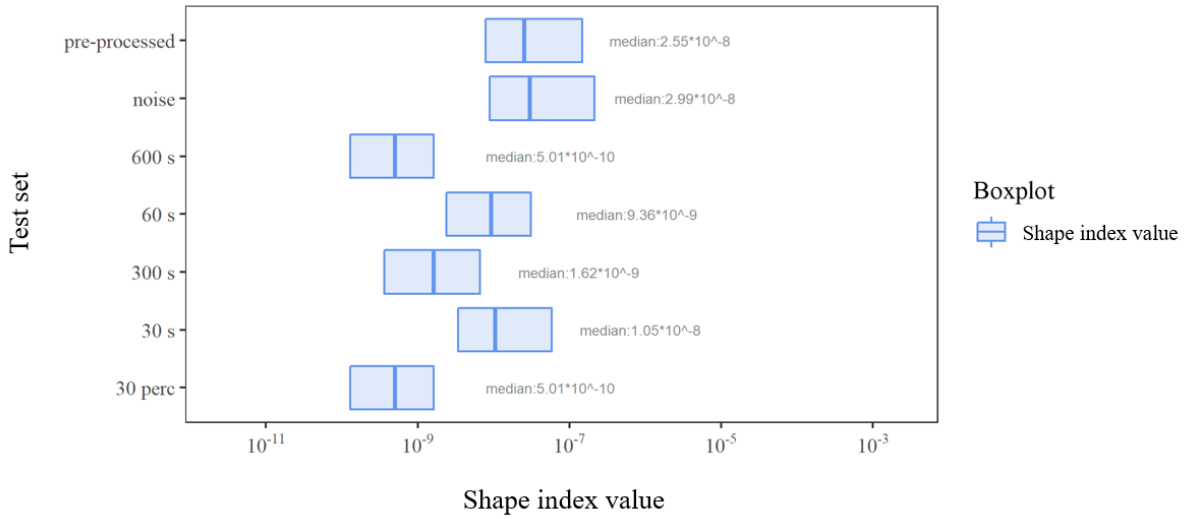


Figure 5.27 Comparison of shape index value of different test sets of SOC algorithm

As it can be seen in *Table 5.27*, the results that can be seen in the boxplots visualised in *Figure 5.27* above were confirmed and the test sets with added noise have the highest key distribution parameters except for the mean value. Furthermore, the key distribution parameters for the regular sampling rate of 600 s and the sampling rate of randomly selected 30 % are identical. The manually labelled ground truth has fewer compact shapes than the only pre-processed SOC algorithm.

Table 5.27 Boxplot statistics of shape index values of different test sets of SOC algorithm

Test set	Min.	Q1	Median	Mean	Q3	Max.
pre-processed data	$8.82*10^{-10}$	$7.84*10^{-09}$	$2.55*10^{-08}$	**$2.03*10^{-05}$**	$1.47*10^{-07}$	**$2.57*10^{-03}$**
added noise	**$8.90*10^{-10}$**	**$8.86*10^{-09}$**	**$2.99*10^{-08}$**	$2.57*10^{-05}$	**$2.16*10^{-07}$**	**$2.57*10^{-03}$**
regular sampling rate 600 s	*$2.46*10^{-12}$*	*$1.29*10^{-10}$*	*$5.01*10^{-10}$*	*$1.23*10^{-07}$*	*$1.63*10^{-09}$*	*$1.47*10^{-05}$*
regular sampling rate 60 s	$5.54*10^{-11}$	$2.38*10^{-09}$	$9.36*10^{-09}$	$1.22*10^{-06}$	$3.12*10^{-08}$	$1.39*10^{-04}$
regular sampling rate 300 s	$4.52*10^{-12}$	$3.61*10^{-10}$	$1.62*10^{-09}$	$1.95*10^{-07}$	$6.59*10^{-09}$	$1.63*10^{-05}$
regular sampling rate 30 s	$3.24*10^{-11}$	$3.43*10^{-09}$	$1.05*10^{-08}$	$1.26*10^{-06}$	$5.85*10^{-08}$	$1.40*10^{-04}$
randomly selected 30 %	*$2.46*10^{-12}$*	*$1.29*10^{-10}$*	*$5.01*10^{-10}$*	*$1.23*10^{-07}$*	*$1.63*10^{-09}$*	*$1.47*10^{-05}$*
manually labelled ground truth	$1.99*10^{-11}$	$6.72*10^{-10}$	$3.08*10^{-09}$	$6.97*10^{-08}$	$7.79*10^{-09}$	$8.71*10^{-07}$

Figure 5.28 shows that the key distribution parameters of the different varying sampling rates of the post-processed data have wider interquartile ranges of shape index values than their respective

pre-processed data. The post-processed original data's median shape index value slightly decreased whereas the median shape index value of the test sets with added noise increased.

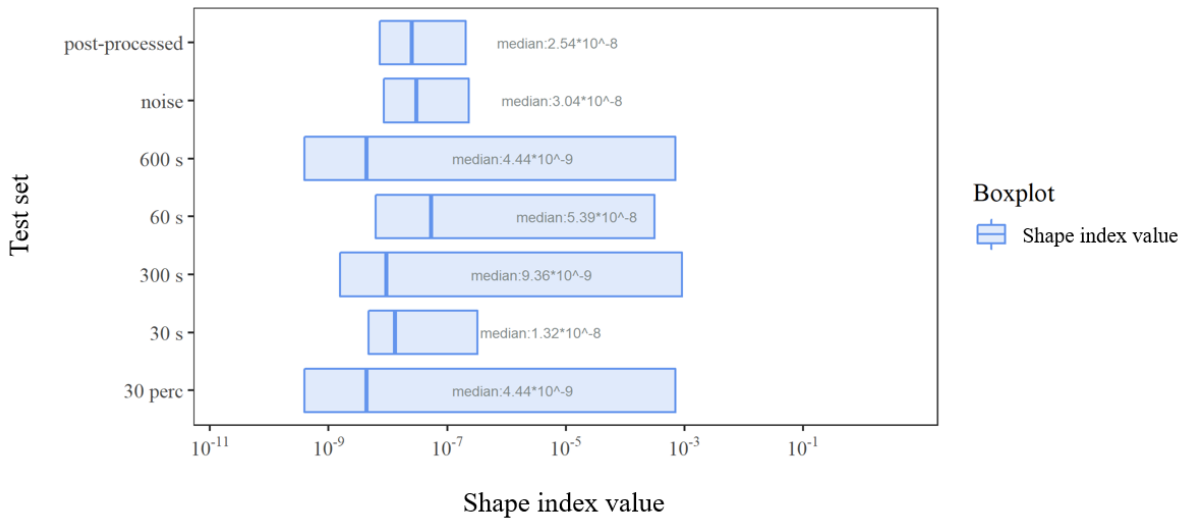


Figure 5.28 Comparison of shape index value of different test sets of post-processed POSMIT algorithm

The regular sampling rate of 600 s has the same key distribution parameters as the sampling rate of randomly selected 30 % (see *Table 5.28*). They also have the widest interquartile ranges which means that these two statuses have the least homogenous shapes. The test sets with added noise and post-processed original data have the most homogenous shapes as their interquartile ranges are closest. The manually labelled ground truth has fewer compact shapes than the post-processed SOC algorithm.

Table 5.28 Boxplot statistics of shape index values of different test sets of post-processed SOC algorithm

Test set	Min.	Q1	Median	Mean	Q3	Max.
post-processed data	$8.82*10^{-10}$	$7.33*10^{-09}$	$2.54*10^{-08}$	*$2.17*10^{-05}$*	*$2.06*10^{-07}$*	*$2.57*10^{-03}$*
added noise	**$8.90*10^{-10}$**	**$8.52*10^{-09}$**	$3.04*10^{-08}$	$2.70*10^{-05}$	$2.29*10^{-07}$	*$2.57*10^{-03}$*
regular sampling rate 600 s	*$1.97*10^{-11}$*	*$3.89*10^{-10}$*	*$4.44*10^{-09}$*	$1.24*10^{-01}$	$7.41*10^{-04}$	$4.01*10^{00}$
regular sampling rate 60 s	$3.07*10^{-10}$	$6.17*10^{-09}$	**$5.39*10^{-08}$**	$1.31*10^{-01}$	$3.55*10^{-04}$	**$5.10*10^{00}$**
regular sampling rate 300 s	$3.37*10^{-11}$	$1.56*10^{-09}$	$9.36*10^{-09}$	**$1.48*10^{-01}$**	**$9.22*10^{-04}$**	$4.01*10^{00}$
regular sampling rate 30 s	$1.03*10^{-10}$	$4.70*10^{-09}$	$1.32*10^{-08}$	$2.87*10^{-02}$	$3.22*10^{-07}$	$1.53*10^{00}$
randomly selected 30 %	*$1.97*10^{-11}$*	*$3.89*10^{-10}$*	*$4.44*10^{-09}$*	$1.24*10^{-01}$	$7.41*10^{-04}$	$4.01*10^{00}$
manually labelled ground truth	$1.99*10^{-11}$	$6.72*10^{-10}$	$3.08*10^{-09}$	$6.97*10^{-08}$	$7.79*10^{-09}$	$8.71*10^{-07}$

5.5 Ground Truth Comparison

After having chosen the threshold combinations per algorithm, the ground truth was calculated for the pre-processed original data with the probabilistic formula introduced in *Section 4.3.3*. In *Figure 5.29*, the boxplots of the F-measure values are visualised per algorithm ("prob." means calculated using the probabilistic formula; "man." indicates the manually labelled ground truth).

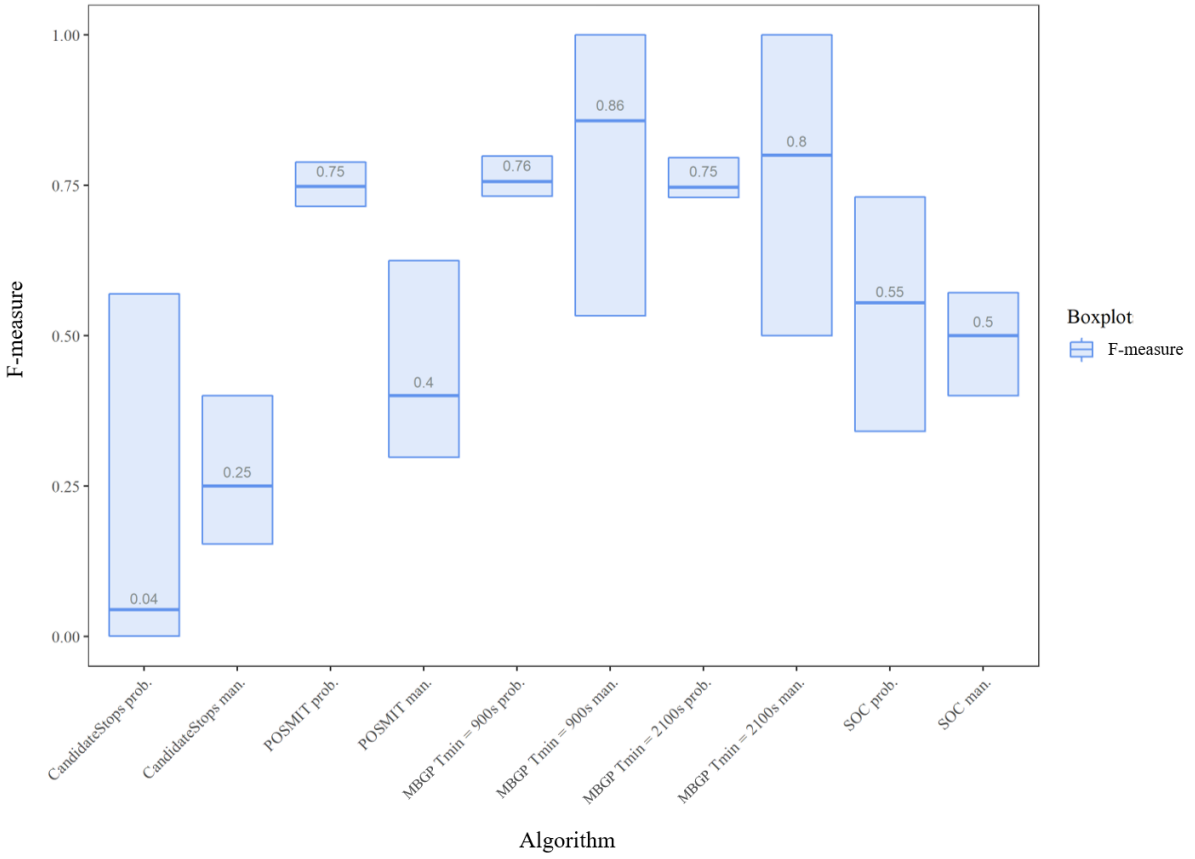


Figure 5.29 Ground truth comparison of replaced ground truth (prob.) and manually labelled ground truth (man.)

The manually labelled ground truth shows higher F-measure median values for the CandidateStops and MBGP algorithm, whereas the ground truth calculated with the probabilistic formula produced higher values for the POSMIT and SOC algorithm. By comparing the different manually labelled ground truths one can see that the MBGP algorithm with T_{min} = 900 s has the highest key distribution parameters (see *Table 5.29*).

Table 5.29 Boxplot statistics of F-measures calculated based on manually labelled ground truth (man.)

Algorithm	Min.	Q1	Median	Mean	Q3	Max.
CandidateStops	$1.12*10^{-01}$	*$1.54*10^{-01}$*	*$2.50*10^{-01}$*	*$2.88*10^{-01}$*	*$4.00*10^{-01}$*	*$5.00*10^{-01}$*
POSMIT	**$1.82*10^{-01}$**	$2.98*10^{-01}$	$4.00*10^{-01}$	$4.75*10^{-01}$	$6.25*10^{-01}$	**$1.00*10^{00}$**
MBGP T_{min} = 900 s	*$1.00*10^{-01}$*	**$5.33*10^{-01}$**	**$8.57*10^{-01}$**	**$7.44*10^{-01}$**	**$1.00*10^{00}$**	**$1.00*10^{00}$**
MBGP T_{min} = 2100 s	$1.05*10^{-01}$	$5.00*10^{-01}$	$8.00*10^{-01}$	$7.38*10^{-01}$	**$1.00*10^{00}$**	**$1.00*10^{00}$**
SOC	**$1.82*10^{-01}$**	$4.00*10^{-01}$	$5.00*10^{-01}$	$4.98*10^{-01}$	$5.71*10^{-01}$	**$1.00*10^{00}$**

The replaced ground truth shows similar results. The F-measure values are different than the manually labelled ones, but the replaced ground truth of the MBGP algorithm with T_{min} = 900 s also yielded the highest key distribution parameters. Similar to the manually labelled ground truth, the CandidateStops algorithm's performance is lowest, as *Table 5.30* shows. The key distribution parameters of the two

threshold combinations of the MBGP algorithm are more similar compared to the manually labelled ground truth values.

Table 5.30 Boxplot statistics of F-measures calculated based on replaced ground truth (prob.)

Algorithm	Min.	Q1	Median	Mean	Q3	Max.
CandidateStops	$7.19*10^{-05}$	$7.44*10^{-04}$	$4.42*10^{-01}$	$2.16*10^{-01}$	$5.69*10^{-01}$	$7.68*10^{-01}$
POSMIT	$5.90*10^{-01}$	$7.15*10^{-01}$	$7.48*10^{-01}$	$7.51*10^{-01}$	$7.89*10^{-01}$	$8.72*10^{-01}$
MBGP T_{min} = 900 s	**$6.00*10^{-01}$**	**$7.32*10^{-01}$**	**$7.56*10^{-01}$**	**$7.63*10^{-01}$**	**$7.99*10^{-01}$**	**$8.78*10^{-01}$**
MBGP T_{min} = 2100 s	$5.61*10^{-01}$	$7.30*10^{-01}$	$7.47*10^{-01}$	$7.56*10^{-01}$	$7.96*10^{-01}$	**$8.78*10^{-01}$**
SOC	$1.31*10^{-02}$	$3.41*10^{-01}$	$5.55*10^{-01}$	$5.15*10^{-01}$	$7.31*10^{-01}$	$8.72*10^{-01}$

5.6 Stop-Move Classification Comparison

5.6.1 General Overview

To give a brief overview of the overall stop-move classification results presented in the previous sections, the following confusion matrix (see *Table 5.31*) visualises the classification results for each algorithm. The results show the percentage of true positive, false negative, and false positive cases out of the classification in order to make the results comparable among the algorithms. The sum of true positives and false negatives is 100 % (where 100 % are the total number of manually labelled stops). The false positive value is the percentage of stops that the algorithm detected too many, compared to the manually labelled ground truth. Conversely, the false negative value indicates the percentage of stops detected too few.

An example of false negatives can be seen in *Figure 5.30*. The black circles visualise the manually labelled stop. Except for the MBGP algorithm with T_{min} = 900 s, the algorithms were not able to detect this stop (blue = stop, yellow = move). This stop duration was longer than 15 minutes (900 seconds) and shorter than 35 minutes (2100 seconds) because it was not detected by the MBGP algorithm with T_{min} = 2100 s. This example shows how important it is to have an optimal threshold value for the data.

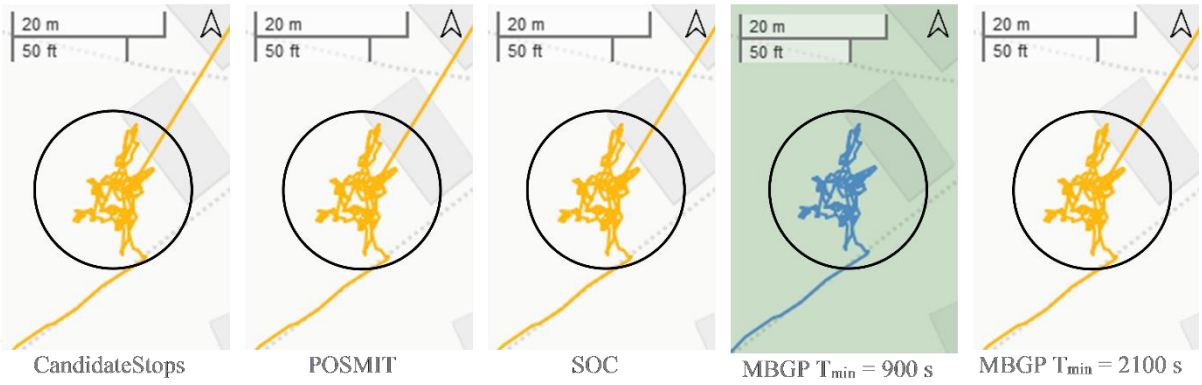


Figure 5.30 Example of false negative stops detected by the algorithms (pre- and post-processed results were identical)

Table 5.31 Confusion matrix (true positives (TP), false negatives (FN), and false positives (FP)) of pre-processed data

	Test set	TP [%]		FN [%]		FP [%]	
CandidateStops	**pre-processed original**	**0.44**		**99.56**		**0.30**	
	regular sampling rate 30 s	0.39		99.61		28.67	
	regular sampling rate 300 s	16.88		83.12		466.23	
	randomly selected 30 %	25.00		75.00		491.67	
	regular sampling rate 60 s	1.58		98.42		96.57	
	regular sampling rate 600 s	25.00		75.00		491.67	
	added noise	0.40		99.60		0.29	
MBGP T_{min} = 900 s (left columns) T_{min} = 2100 s (right columns)	**pre-processed original**	**24.87**	**28.13**	**75.13**	**71.88**	**69.11**	**95.49**
	regular sampling rate 30 s	21.56	25.26	78.44	74.74	75.32	105.26
	regular sampling rate 300 s	23.75	26.74	76.25	73.26	92.50	117.44
	randomly selected 30 %	23.25	24.89	76.75	75.11	114.02	140.44
	regular sampling rate 60 s	24.05	25.96	75.95	74.04	77.03	104.56
	regular sampling rate 600 s	23.25	24.89	76.75	75.11	114.02	140.44
	added noise	22.20	27.24	77.80	72.76	61.66	91.99
POSMIT	**pre-processed original**	**68.89**		**31.11**		**344.44**	
	regular sampling rate 30 s	43.82		56.18		374.16	
	regular sampling rate 300 s	39.77		60.23		382.95	
	randomly selected 30 %	42.68		57.32		410.98	
	regular sampling rate 60 s	44.32		55.68		378.41	
	regular sampling rate 600 s	42.68		57.32		410.98	
	added noise	67.78		32.22		332.22	
SOC	**pre-processed original**	**38.93**		**61.07**		**202.01**	
	regular sampling rate 30 s	25.44		74.56		185.80	
	regular sampling rate 300 s	27.65		72.35		186.47	
	randomly selected 30 %	26.88		73.13		200.00	
	regular sampling rate 60 s	24.86		75.14		167.57	
	regular sampling rate 600 s	26.88		73.13		200.00	
	added noise	33.12		66.88		187.26	

The pre-processed original data run with the POSMIT algorithm had the highest percentage of true positive values compared to the other algorithms, followed by the SOC algorithm. Hence, the POSMIT algorithm detected three times more false positive values and the SOC algorithm two times more false positive values. The pre-processed original data have the highest true positive values compared to the other test sets calculated with the respective algorithm except for the CandidateStops algorithm. There, both the sampling rate of randomly selected 30 % and the regular sampling rate of 600 s had a percentage

of 25 %. The MBGP algorithm with a T_{min} of 2100 s had a higher percentage of true positive values compared to a T_{min} of 900 s. Generally, the MBGP algorithm overclassified fewer points as stops for a T_{min} of 900 s compared to the T_{min} of 2100 s.

Figure 5.31 gives an example of how false positive stops could have occurred. At the presumed home locations where the participants stayed for longer periods of time, the GPS signal was wandering, which led to a star-shaped trajectory around the home location. Therefore, some algorithms detected multiple stops (indicated with green circles) for the same home location although they should only detect two stops (i.e., this participant left the house in the morning and returned in the evening). This behaviour led to many false positive stops in the evaluation. In the example of *Figure 5.31*, the CandidateStops algorithm detected the most of these false positive stops followed by the MBGP algorithm for a T_{min} of 900 s.

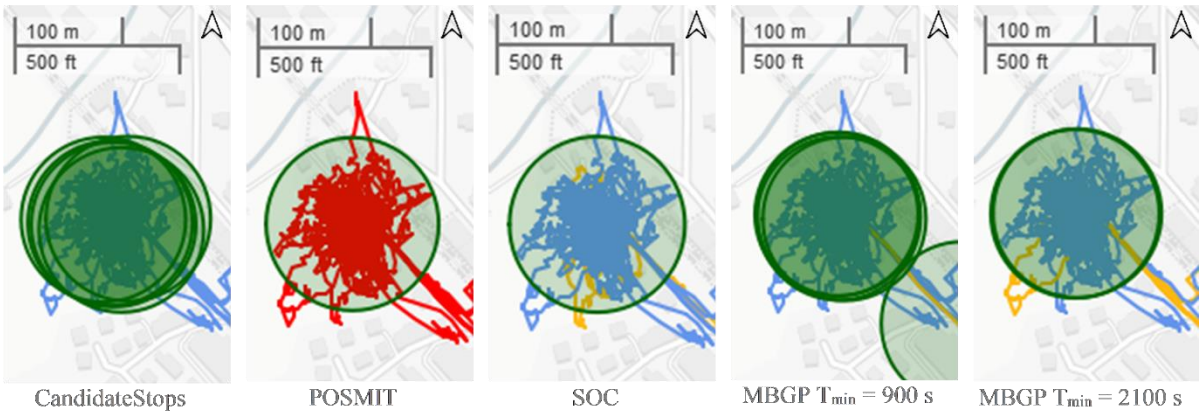


Figure 5.31 Example of false positive stops detected by algorithms (pre-processed)

The patterns described above can also be found when looking at the post-processed data in *Table 5.32* and at another example visualised in *Figure 5.32*. Compared to *Figure 5.31* post-processing only decreased the number of false positive stops for the MBGP algorithm with T_{min} = 900 s.

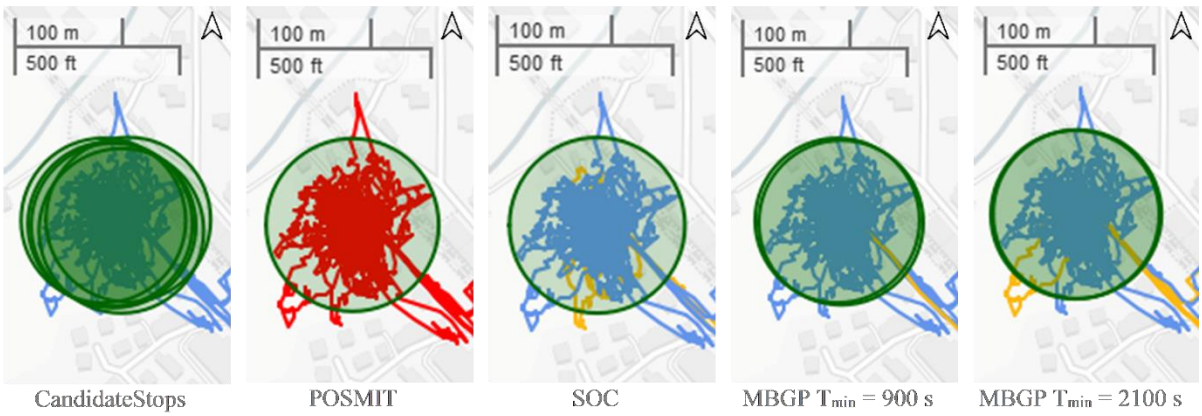


Figure 5.32 Example of false positive stops detected by algorithms (post-processed)

In *Table 5.32*, one can further see that, in contrast to the example in *Figure 5.32*, post-processing the data generally led to increased numbers of true positives and false positives except for the POSMIT algorithm. As the number of true positives increased through post-processing, the number of false negatives inevitably decreased. In *Table 5.32*, the sum of true positives and false negatives is 100 % (where 100 % are the total number of manually labelled stops).

Table 5.32 Confusion matrix (true positives (TP), false negatives (FN), and false positives (FP)) of post-processed data

	Test set	TP [%]		FN [%]		FP [%]	
CandidateStops	**post-processed original**	**4.36**		**95.64**		**73.03**	
	regular sampling rate 30 s	4.59		95.41		185.20	
	regular sampling rate 300 s	23.33		76.67		520.00	
	randomly selected 30 %	15.79		84.21		484.21	
	regular sampling rate 60 s	8.40		91.60		275.57	
	regular sampling rate 600 s	15.79		84.21		484.21	
	added noise	4.05		95.95		75.48	
MBGP T_{min} = 900 s (left columns) T_{min} = 2100 s (right columns)	**post-processed original**	**31.08**	**35.00**	**68.92**	**65.00**	**84.80**	**107.92**
	regular sampling rate 30 s	40.74	34.96	59.26	65.04	303.70	267.48
	regular sampling rate 300 s	43.33	44.44	56.67	55.56	370.00	368.89
	randomly selected 30 %	46.67	47.78	53.33	52.22	366.67	365.56
	regular sampling rate 60 s	43.48	45.45	56.52	54.55	360.87	330.30
	regular sampling rate 600 s	46.67	47.78	53.33	52.22	366.67	365.56
	added noise	26.55	31.46	73.45	68.54	83.19	107.87
POSMIT	**post-processed original**	**68.89**		**31.11**		**344.44**	
	regular sampling rate 30 s	43.82		56.18		374.16	
	regular sampling rate 300 s	39.77		60.23		377.27	
	randomly selected 30 %	42.68		57.32		391.46	
	regular sampling rate 60 s	44.32		55.68		378.41	
	regular sampling rate 600 s	42.68		57.32		391.46	
	added noise	68.89		31.11		344.44	
SOC	**post-processed original**	**42.54**		**57.46**		**202.24**	
	regular sampling rate 30 s	35.62		64.38		282.19	
	regular sampling rate 300 s	56.25		43.75		406.25	
	randomly selected 30 %	52.08		47.92		418.75	
	regular sampling rate 60 s	38.46		61.54		316.92	
	regular sampling rate 600 s	52.08		47.92		418.75	
	added noise	34.67		65.33		196.00	

5.6.2 Variance and Average Number of Stops

Apart from the accuracy of the different algorithms that was shown by displaying the confusion matrix, the algorithms can also be compared through the variance and the average number of stops they detected. *Table 5.33* gives an overview of the variance and average number of stops per day and algorithm. Except for the only pre-processed CandidateStops algorithm, the results are quite similar. Post-processing the CandidateStops algorithm substantially improved the variance and mean results. The variance as well as the mean of the two post-processed MBGP algorithms' results are the same, whereas they show different results after only pre-processing the data. The MBGP algorithms' results are closest to the results of the manually labelled ground truth.

Table 5.33 Overview of variance and average number of stops per day

Algorithm	Variance	Mean
CandidateStops pre-processed	1536.720	9.900
CandidateStops post-processed	5.159	4.144
POSMIT pre-processed	5.061	4.133
POSMIT post-processed	5.061	4.133
MBGP pre-processed T_{min} = 900 s	5.251	4.126
MBGP post-processed T_{min} = 900 s	5.011	4.035
MBGP pre-processed T_{min} = 2100 s	5.177	4.092
MBGP post-processed T_{min} = 2100 s	5.011	4.035
SOC pre-processed	4.648	4.432
SOC post-processed	4.865	4.493
Manually labelled ground truth	6.028	3.800

5.6.3 Running Time

Finally, in order to compare the general performance of the algorithms, the average running time per algorithm was determined by checking how long an algorithm did take on average to process a test set. As *Table 5.34* shows, the SOC algorithm is the slowest algorithm, while the MBGP algorithm runs fastest.

Table 5.34 Average algorithm running time per data frame

Algorithm	Average Running Time
CandidateStops	40.0 s
MBGP	9.1 s
POSMIT	42.5 s
SOC	6.7 min ≈ 402.8 s

5.6.4 Summary of Results

To give an overview of the key results that will be discussed in *Chapter 6*, *Table 5.35* was created based on the hypotheses and the three scenarios introduced in *Chapter 4*.

Table 5.35 Result overview of the four algorithms

	Optimal parameter thresholds	Median F-measure of test sets			Shape index values of test sets			Average running time
		test sets	value pre-processed	value post-processed	test sets	value pre-processed	value post-processed	
CandidateStops	ε = 0.5 m/s	original	0.25	0.33	original	$1.91*10^{-15}$	$9.30*10^{-09}$	40.0 s
		noise	0.40	0.29	noise	$1.54*10^{-15}$	$1.87*10^{-08}$	
		30 s	0.33	0.33	30 s	$1.79*10^{-16}$	$3.64*10^{-11}$	
		60 s	0.33	0.53	60 s	0.00	$1.08*10^{-11}$	
		300 s	0.33	0.50	300 s	0.00	0.00	
		600 s	0.40	0.33	600 s	0.00	0.00	
		keep 30 %	0.40	0.33	keep 30 %	0.00	0.00	
MBGP	D_{max} = 275 m T_{max} = 14400 s T_{min} = 900 s (left columns) or T_{min} = 2100 s (right columns)	original	0.86	0.80 / 0.74	0.80	original	$1.23*10^{-07}$	$1.27*10^{-07}$ / $1.46*10^{-07}$	$1.29*10^{-07}$	9.1 s
		noise	0.86	0.80 / 0.67	0.67	noise	$1.07*10^{-07}$	$1.18*10^{-07}$ / $1.35*10^{-07}$	$1.35*10^{-07}$	
		30 s	0.73	0.67 / 0.40	0.40	30 s	$6.90*10^{-08}$	$6.72*10^{-08}$ / $3.20*10^{-06}$	$2.01*10^{-07}$	
		60 s	0.80	0.67 / 0.33	0.33	60 s	$5.06*10^{-08}$	$4.67*10^{-08}$ / $2.63*10^{-05}$	$1.31*10^{-06}$	
		300 s	0.67	0.67 / 0.33	0.33	300 s	$7.35*10^{-09}$	$6.20*10^{-09}$ / $4.14*10^{-06}$	$2.68*10^{-06}$	
		600 s	0.67	0.67 / 0.33	0.33	600 s	$2.30*10^{-09}$	$2.24*10^{-09}$ / $1.28*10^{-06}$	$1.05*10^{-07}$	
		keep 30 %	0.67	0.67 / 0.33	0.33	keep 30 %	$2.30*10^{-09}$	$2.24*10^{-09}$ / $1.28*10^{-06}$	$1.05*10^{-07}$	
POSMIT	ε = 0.25 h_d = calculated h_i = 1	original	0.40	0.40	original	$4.09*10^{-04}$	$4.09*10^{-04}$	42.5 s
		noise	0.40	0.40	noise	$4.10*10^{-04}$	$4.10*10^{-04}$	
		30 s	0.33	0.33	30 s	$2.52*10^{-04}$	$2.52*10^{-04}$	
		60 s	0.33	0.33	60 s	$3.27*10^{-04}$	$3.27*10^{-04}$	
		300 s	0.33	0.33	300 s	$2.39*10^{-04}$	$2.39*10^{-04}$	
		600 s	0.33	0.33	600 s	$4.55*10^{-04}$	$4.55*10^{-04}$	
		keep 30%	0.33	0.33	keep 30 %	$4.55*10^{-04}$	$4.55*10^{-04}$	
SOC	*Eps* = 75 m *Tau* = 1200 s *MinMov* = 180 s	original	0.50	0.50	original	$2.55*10^{-08}$	$2.54*10^{-08}$	402.8 s
		noise	0.50	0.50	noise	$2.99*10^{-08}$	$3.04*10^{-08}$	
		30 s	0.40	0.40	30 s	$1.05*10^{-08}$	$1.32*10^{-08}$	
		60 s	0.50	0.40	60 s	$9.36*10^{-09}$	$5.39*10^{-08}$	
		300 s	0.48	0.33	300 s	$1.62*10^{-09}$	$9.36*10^{-09}$	
		600 s	0.40	0.33	600 s	$5.01*10^{-10}$	$4.44*10^{-09}$	
		keep 30 %	0.40	0.33	keep 30 %	$5.01*10^{-10}$	$4.44*10^{-09}$	

6 Discussion

6.1 Initial Findings

The first observation that one can make about *Table 5.35* is that the POSMIT algorithm has identical results for all of the different test sets and shape measures before and after post-processing the results. The decision for the most suitable threshold combination was made only based on the boxplot statistics and percentage of valid results. Furthermore, the most suitable threshold combination was selected for the only pre-processed and the post-processed data independently. By comparing the percentage of true positives and the average number of stops per day, one could see as well that the results were identical. Therefore, post-processing the results (at least in the way it was carried out in this thesis) neither changed the number of stops nor did it improve the results as other threshold combinations did when looking only at the boxplot statistics of the POSMIT algorithm.

As the decision was made to estimate/calculate the spatial parameter (h_d) for each testing day individually, a sensitivity to the spatial parameter could not be determined for the POSMIT algorithm. The method to estimate/calculate the spatial parameter prevents the algorithm from having to deal with changes in the spatial parameter. Therefore, the algorithm is not sensitive to changes in the spatial parameters at all. Based on these findings, one of the hypotheses stated at the beginning of the thesis concerning the POSMIT algorithm can be rejected.

H3-1: The detection results of the probability-based algorithms are less sensitive to changes in spatial parameters for the specific MOASIS dataset than for grid- or density-based algorithms.

This hypothesis was formulated before studying the literature in detail and before selecting the value ranges for the single parameters. The CandidateStops algorithm that is a density-based algorithm does not have a spatial parameter. The density-based MBGP algorithm was able to find stops for all proposed spatial parameter values, in contrast to the centre-based SOC algorithm. The SOC algorithm was not able to find stops for values larger than 75 m. Therefore, H3-1 could be re-phrased to:

> The detection results of the centre-based algorithms are more sensitive to changes in spatial parameters for the specific MOASIS dataset compared to density-based algorithms.

Furthermore, the question arises if prioritizing the percentage of valid results over the F-measure values visualised with boxplots was an appropriate approach, as the percentage of valid results was in general very low for all algorithms. This means that the manually labelled ground truth does not match well with the detected stops of the algorithms. One reason could be that the parameters used to check if the stops match with the manually labelled ground truth are poorly chosen and therefore a lot of stops remained undetected. Nevertheless, if the time range and radius from the first data point of a stop cluster were too big, the same true positive stop would be detected multiple times by an algorithm that detects rather stop

points instead of stop regions. Another reason could be that the manually labelled ground truth's quality is not that good as labelling stops manually is rather subjective.

6.1.1 Influence of Manually Labelled Ground Truth

As long as there is no actual ground truth, manual ground truth labelling is always subjective and therefore, there is no guarantee that the actual stops were discovered. Manual stop identification can be different depending on the zoom level. As *Figure 6.1* shows, because of the high zoom level, it seems like the person walked to a location and stayed there.

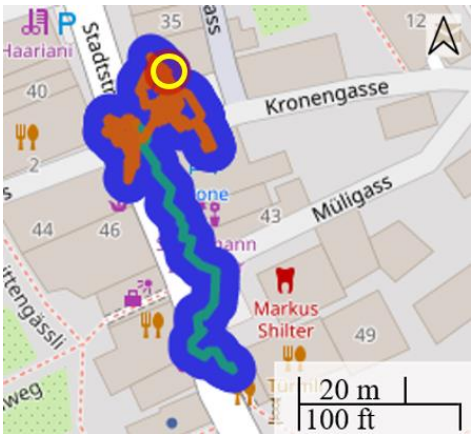


Figure 6.1 Manual ground truth labelling example on high zoom level (green = move, orange = stop)

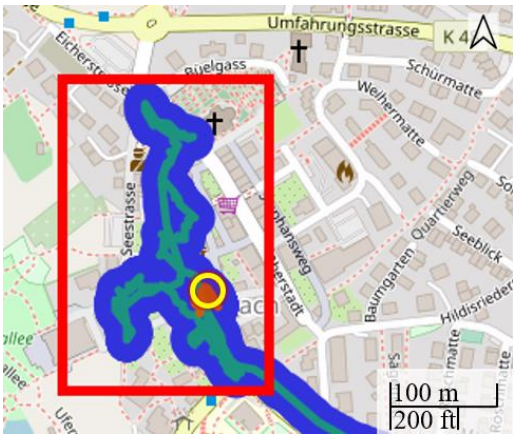


Figure 6.2 Manual ground truth labelling example on low zoom level (green = move, orange = stop)

However, by looking at the zoomed-out extent of the same area outlined in red in *Figure 6.2* (the yellow circle in *Figure 6.1* as well as in *Figure 6.2* is at the same location). For a period of only a few minutes, the movement patterns look quite unnatural (especially in the x-direction), which could also indicate signal wandering. Depending on how this situation was judged, it could influence whether the starting point of the stop detected by the algorithm lies in the range of 150 meters and 20 minutes to the starting point of the detected stop or not. In cases of signal wandering due to indoor situations, the acceleration was quite restless and therefore an evaluation by eye and "common sense" influences the labelling process, although three scientific criteria were introduced that could describe a stop. In order to assess the quality of the manually labelled ground truth, the interrater agreement could have been determined (Fleiss, Levin and Paik, 2003). This means that several people would have had to additionally label the same ground truth manually. If the agreement between the manually labelled ground truths had been good, there would have been a high probability, but no guarantee, that the detected stops were also the real stops (ibid.). This approach might have provided a better basis to detect problematic cases.

To find out if the probabilistic formula to replace the ground truth delivers better results than the manually labelled ground truth, actual ground truth would be needed. For the MBGP algorithm and the

SOC algorithm, the values of manually labelled ground truth and the replaced ground truth showed similar median F-measure results. This could indicate that the results of these two algorithms are closer to the actual ground truth than the results of the POSMIT and CandidateStops algorithm.

6.1.2 Relationship between functioning of Algorithms and Results

The CandidateStops algorithm checks each data point individually, which means that stops are detected even for very short-term changes in motion (which may be caused by signal wandering). Although moves shorter than 3 minutes are removed during post-processing, the result still contains so many stops that some of them are detected correctly. However, most of the detected stops are false positives. This argument is supported by the fact that the number of true and false positives detected by the CandidateStops algorithm seem to depend very much on the sampling rate.

The other three algorithms not only consider individual points but also their neighbours and therefore, the influence of individual points becomes smaller. This "smoothing mechanisms" reduce the sensitivity to varying sampling rates and it could also be the reason why post-processing seems to influence the CandidateStops algorithm's results most. It does not have its own mechanism to deal with outliers and short-term fluctuations in the raw data, whereas the other algorithms are able to reduce the influence of such fluctuations.

6.1.3 Meaningfulness of Running Time

The empirically measured average running time is a result that needs to be interpreted with some caution. Apart from the computational complexity of the algorithm, it is dependent on several practical factors, such as the efficiency of implementation of the algorithm in R, the CPU, as well as the available memory. Nevertheless, with all due caution it can be observed that the MBGP algorithm ran fastest out of the four algorithms and seemed to have the highest median F-measure values, especially for the only pre-processed data with the threshold combination T_{min} = 900 s.

6.2 Scenario 1: Handling Noise

When looking only at the original data, the MBGP algorithm had the second highest percentage of valid results, after the POSMIT algorithm. Hence, the POSMIT algorithm had the second lowest median F-measure value compared to the other algorithms. Therefore, one could argue that under perfect conditions the MBGP algorithm with pre-processed data and T_{min} of 900 s delivered the best results. However, it seems more realistic that real-life data have large sampling rates (> 1 s) and include noise.

Scenario 1 wants to find the algorithm that could best handle noise in case one had to deal with very noisy datasets. For noise simulation, Gaussian noise was not selected as it is only suitable to simulate GPS errors, that in general were filtered out during pre-processing. Instead of adding additional noise to

the data, and therefore having to deal with more data points, the chosen method replaced existing longitude values with noise values. As some median F-measure values were higher for the test sets with added noise compared to the original data, the question arises if the selected noise model really was an appropriate choice for the data or if other non-Gaussian/non-linear noise models would have been more suitable. The choice of the noise model depends on the type of noise that wants to be simulated. In case of the added noise, GPS inaccuracies were simulated. The spatial distortions caused by the added noise could have moved the starting point of the stop cluster closer to the stop cluster of the manually labelled ground truth. This could explain why the F-measure values were higher for some of the algorithms. However, it seems unlikely to happen systematically.

Based on *Table 4.1* that was used to select the four algorithms, the SOC and POSMIT algorithms were said to be resilient to noise. This characteristic was not explicitly mentioned for the CandidateStops and MBGP algorithms.

As the results show, the F-measure values either stayed the same after post-processing the data or decreased. There was no algorithm where post-processing increased the F-measure results of test sets with added noise. However, post-processing increased the percentage of valid results by about 5 % for the MBGP algorithm and the CandidateStops algorithm. Only based on the percentage of valid results, the POSMIT algorithm could best handle noise. Only based on the F-measure values, the only pre-processed MBGP algorithm with a T_{min} of 900 s performed best as the median F-measure value is 0.86. By considering the combination of F-measure values and percentage of valid results, the post-processed MBGP algorithm seems to best handle noise as its percentage of valid results is around 30 % and the median F-measure value is 0.67. Interestingly, while there are obvious differences in the F-measure results of the only pre-processed data of the MBGP algorithm with thresholds T_{min} = 900 s and T_{min} = 2100 s, their F-measure values were almost equal for the post-processed test sets with added noise. While the MBGP algorithm would be most suitable for handling noise in datasets, the CandidateStops algorithm would be least suitable as the percentage of valid results is at maximum 5 %. For the first scenario, the ranking of algorithms based on fulfilled criteria is true for the CandidateStops algorithm that fulfilled the least criteria, but is not true for the MBGP algorithm that was third in the ranking out of the four algorithms.

6.3 Scenario 2: Handling Sampling Rate

Scenario 2 wants to find the algorithm that performs best with varying or low sampling rates. According to the following hypothesis, the POSMIT algorithm was expected to cope best with varying sampling rates:

H3-2: The probability-based algorithms perform better than the other algorithms when the sampling rate is bigger than 1 s.

Based on the F-measure values, the POSMIT algorithm had similar results as the CandidateStops algorithm. For a regular sampling rate of 600 s, the POSMIT algorithm had worse results for the only pre-processed data as well as for the post-processed test sets with sampling rates of 60 s and 300 s. The SOC algorithm and the MBGP algorithm had higher results. However, based on the percentage of valid results, the POSMIT algorithm handles different sampling rates second best after the MBGP algorithm.

Similar to the resilience to noise, the CandidateStops and the MBGP algorithm were not explicitly said to handle sparse trajectory data well (see *Table 4.1*). Even though the CandidateStops algorithm has similar F-measure values compared to the other algorithms, one can say that the algorithm does not handle sparse trajectory data well as the percentages of valid results were below 10 %. Unlike the algorithm comparison of *Table 4.1*, the MBGP algorithm seems to handle sparse trajectory data best. In contrast to the other algorithms, the MBGP algorithm has a constraint that prevents two data points separated by large time periods from belonging to the same stop cluster (Montoliu, Blom and Gatica-Perez, 2013). As the values of T_{max} were considerably higher than the maximum chosen regular sampling rate, the chance of a data point being matched to the wrong stop cluster is minimal.

In terms of highest percentage of valid results and F-measure values, the regular sampling rate of 60 s for the only pre-processed test sets of the threshold combination with T_{min} = 900 s had the closest result to the original sampling rate. Apart from this, the only pre-processed MBGP algorithm's results of T_{min} = 900 s, as well as the results of T_{min} = 2100 s, had the same median F-measure values for a sampling rate of randomly selected 30 % and a regular sampling rate of 300 s and 600 s. The interquartile ranges of the F-measure values and the percentage of valid results are close for the two parameter combinations. Additionally, the median F-measure values are identical for the three sampling rates. The post-processed MBGP algorithm's results have the same median F-measure values for all of the different varying sampling rates except for the regular sampling rate of 30 s. Similar to the only pre-processed MBGP algorithm's results, the two different threshold combinations have the same median F-measure values and very similar key distribution parameters. These findings 1) lead to the rejection of hypothesis H3-2, and 2) showed that the MBGP algorithm is not sensitive to temporal input parameters either (i.e., seems to handle sparse trajectory data best).

As explained above, the results of the two MBGP threshold combinations that only have different T_{min} values are almost identical, especially for the post-processed data. The analysis of the parameter T_{max} further showed, that the F-measure results were all identical for the different T_{max} values. Therefore, one can argue that the post-processed data run with the density-based MBGP algorithm in particular are not sensitive to temporal input parameters. The SOC algorithm that is also in the category of density- and centre-based algorithms, showed similar results. The median F-measure values of the different selected *Tau* values were identical, and the percentage of valid results were quite similar. As the *Tau* value is similar to the T_{min} value of the MBGP algorithm, one can reject the hypothesis stated below:

H3-3: The detection results of density-based algorithms using a minimum stop duration threshold are most sensitive to temporal input parameters as setting the optimal duration threshold is not trivial (Gong *et al.*, 2015).

The CandidateStops algorithm does not have a temporal parameter, that is why it could not be considered for analysing the impact of minimum stop durations for density-based algorithms.

6.4 Scenario 3: Shape Compactness

In Scenario 3, the algorithm that has the most compact shapes should be found. A compact shape could indicate that the stop is more of a stop region instead of a single stop point such as at a bus stop for example. Most of the algorithms' shape index values were less homogeneous for the post-processed data. Based on the data range of the shape index values, the CandidateStops algorithm has the least compact and the POSMIT algorithm the most compact shapes. The SOC and MBGP algorithm had quite similar values, whereas the SOC algorithm had lower shape index values than the MBGP algorithm. The MBGP algorithm with T_{min} = 900 s further had more compact shapes than the results with T_{min} = 2100 s. Based on these results, the following hypothesis can be rejected as the probability-based algorithm delivers the most compact shapes.

H4-2: The density-based algorithms deliver most compact stops for the specific MOASIS dataset due to their ability to build clusters of contiguous data points that are close to each other in space and time (Bermingham and Lee, 2018).

However, the values are all closer to 0 than to 1 and therefore generally not that compact. Post-processing the data improved the shape compactness. This indicates that more smaller stop points were found (especially for the only pre-processed data). This could be a reason for the high number of false positives that the algorithms detected. They detected up to three times more stops than stops were labelled manually. One explanation could be that when participants stayed indoors for a long time, the GPS signal was wandering, and these smaller outliers could be detected as moves. Due to the GPS signal wandering, multiple stops could be detected for the same stop region. Therefore, the hypothesis stated below cannot be rejected.

H4-1: The spatial extent varies between the applied algorithms with respect to shape, size, and spatial extent. A stop can be classified as an area or as multiple smaller points. Since a high resolution of detected stops is anticipated to occur, especially without post-processing the results, the algorithms are expected to find smaller stop points instead of one bigger area. Therefore, all algorithms will deliver more false positives than false negatives when compared to the ground truth.

One hypothesis about the compactness of stops detected by the geography-based algorithms was left open in the discussion so far.

H4-3: The least compact stops result from the geography-based algorithms as they use predefined geographic geometries to detect stops (Bermingham and Lee, 2018). Hence, it can be assumed that movements in e.g., a building will not be detected.

As geography-based algorithms were only part of the literature review, this hypothesis can only be answered based on literature. Literature confirms that the geography-based algorithms can only detect the stops of the predefined geographic geometries (Alvares *et al.*, 2007). However, one could argue that the stops that will be detected could be compact, as the users can select the geographic geometries. Therefore, the users can decide if they want to use predefined geographic point or polygon geometries and control the compactness of the stops' shapes by themselves. That is why hypothesis H4-3 can be rejected.

6.5 Influence of Pre- and Post-Processing on Results

During the process of writing this thesis, serious outliers were discovered in the MOASIS data by the researchers; they also removed speed outliers for pre-processing the data. This would indicate that the chosen pre-processing method of only removing speed outliers would not be enough if the algorithm had to run with data containing outliers other than speed outliers. As the algorithms run with the test sets with which ground truth was labelled manually and no severe outliers were discovered during the labelling process, the proposed method for pre-processing was suitable (i.e., removing GPS points with the speed of higher than 200 km/h). The question arises how the algorithms' performance would change after the data were pre-processed with another method. A Kalman Filter, that applies probabilistic estimations to the model, could help to remove the severe outliers of the MOASIS data because it can effectively reject uncorrelated disturbances by preserving the shape of noise-free registrations in data with high sampling rates (Leśniak, Danek and Wojdyła, 2009; Park *et al.*, 2019). Furthermore, post-processing did not improve most of the results in terms of median F-measure values and percentage of valid results. As the approaches for how the algorithms categorize the data points into stops and moves are very different, using a unique post-processing method for each algorithm could be more suitable instead of using the same method for all of the algorithms. Another argument could be that the parameters were badly chosen. Therefore, setting the parameters of the post-processing method differently could improve the results. However, this would require a more sophisticated analysis of these parameters.

Additionally, post-processing the data led to a higher number of false positive stops. One reason causing this effect is visualised in *Figure 6.3*. The blue circles in *Figure 6.3* visualise the manually labelled stops, the white circles visualise stops detected by the algorithms. As too short moves (i.e., moves shorter

than 3 minutes) were removed and close stops merged in the post-processing step, stops that were labelled as true positive stops (see Case 1) could be labelled as false positive stops after post-processing because they do not meet the criteria of a true positive stop anymore (see Case 2).

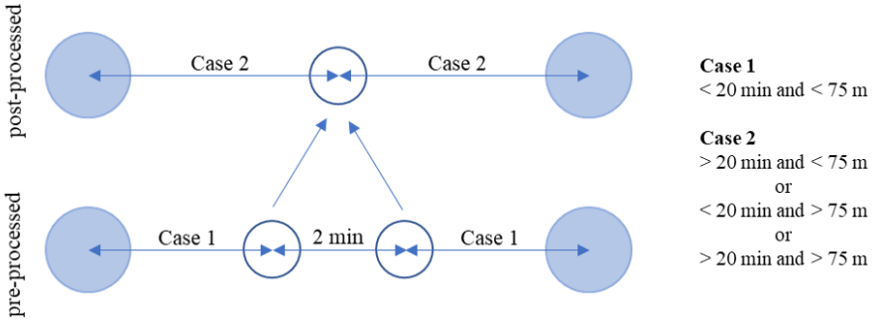


Figure 6.3 Schematic explanation of false positive stops remaining after post-processing

By changing the post-processing method to removing too short stops instead of too short moves, the number of false positive stops could be decreased because by eliminating too short stops instead of merging them, less stops would be detected in general. However, this could also negatively influence the number of detected true positive stops if actual stops were removed.

6.6 Evaluation Criteria and Measures

As the discussion showed, the MBGP algorithm with T_{min} = 900 s was best for two out of three scenarios: best in handling noisy data and best in handling different sampling rates. Furthermore, its results concerning average number of stops per day were closest to the manually labelled ground truth. This indicates that the algorithm selection criteria described in *Table 4.1* were not properly chosen as the MBGP algorithm was not among the most suitable algorithms when the algorithms were ranked according to the chosen criteria. The number of parameters (i.e., at least 3 input parameters) and that the algorithms should have at least one spatial and one temporal input parameter seemed to be strong criteria. The CandidateStops algorithm that only used speed as input parameter did not match one of these two criteria and performed worst for all of the chosen scenarios and evaluation measures.

The evaluation further showed that algorithms, whose parameters are easy to understand, performed better than algorithms with complex parameters and that the criterion “ease of understanding” is therefore particularly important. Considering this criterion, the MBGP algorithm is best suited because its parameters can be mentally visualised and are also understandable for non-expert researchers. In contrast, the POSMIT and the SOC algorithms are more complex: although Bermingham (2018) stated that POSMIT can be applied by non-expert researchers, as there is an opportunity to estimate the parameters, getting a profound understanding of them is not trivial. Regarding the SOC algorithm, especially its calculation of core sequences, it is very specific and not easy to understand.

Measuring sensitivity to accuracy with varying sampling rates and added noise compared to the original data combined with the F-measures seemed to be an appropriate approach to evaluate the algorithms' results. This is because even within the same dataset, the data size of each individual or each day may vary due to behavioural fluctuations, different sensor types, or amount of time spent in-/outdoors. Since finding the most suitable parameter threshold combinations was an extensive research method, the sensitivity of the single input parameters could also be looked at. This led to the finding that the results for parameter T_{max} of the MBGP algorithm did not change depending on the chosen values. The results of the SOC algorithm's *Tau* values were also close and therefore, the SOC algorithm is not that sensitive to changes of the temporal input parameter. The CandidateStops algorithm on the other hand showed considerable differences in results for the different tested speed variations, especially for testing days where the participant only took walks or stayed at home[11]. The differences were also considerable for the parameters of the POSMIT algorithm. The results for a stop probability of 0.25 and a search bandwidth of 1 were highest and results changed a lot when these parameters were changed, as the threshold analysis showed.

In case of the shape measures, the algorithms all have similar results and most likely the SOC algorithm matches with the shape index values of the manually labelled ground truth. Since no good differentiation among the algorithms is possible, other shape measurements might be better suited to evaluate the shape of the stops.

6.7 Comparison with Fillekes, Kim, *et al.* (2019)

Since the MBGP algorithm seems to suit the MOASIS data best in two out of the three proposed scenarios (i.e., Scenario 1 and Scenario 2) as well as regarding the percentage of valid results and the average number of stops per day, the parameter thresholds can be compared with Fillekes, Kim, *et al.* (2019). They used the MBGP algorithm with data from another healthy aging research project whereby the data were similar to the MOASIS data used in this thesis (i.e., data had a sampling rate of 1 s). The threshold combination determined through their sensitivity analysis was: D_{max} = 125 m, T_{max} = 18000 s, and T_{min} = 360 s. The values of D_{max} and T_{min} are over 50 % smaller than the values determined in this thesis (D_{max} = 275 m, T_{max} = 14400 s, and T_{min} = 900 s). As the thesis' focus was rather in finding stop regions instead of single stop points, too small threshold values were not considered. The T_{min} parameter for example, was tested starting with 10 minutes as this value was based on the average shopping time in a supermarket. Furthermore, Montoliu, Blom and Gatica-Perez (2013) stated that a T_{min} between 20 and 40 minutes delivers the best results. In both studies, the F-measure barely reacted to varying T_{max}.

[11] When paths were covered with higher speed than walking speed, the CandidateStops algorithm seemed to perform better than the POSMIT algorithm. See Appendix 9.1 for an example, where the participant covered a long distance by train and car. However, in this example the MBGP algorithm with T_{min} = 900 s performed best.

7 Conclusion

7.1 Summary and Major Findings

Analysing mobility patterns of older adults from a spatial and temporal perspective is an important factor in health planning processes. Stop-move detection is essential to extract semantic information on the human daily mobility (e.g., traveling behaviours) and potential activities performed at stop locations from trajectory data captured by location sensor devices. The aim of this thesis was therefore to review and compare stop-move detection algorithms for GPS data in light of healthy aging research projects, using GPS data collected in the MOASIS project. In order to find the algorithm that works best with the MOASIS data, four research questions (RQs) were stated: two at a conceptual and two at an analytical level.

RQ1 was concerned with the selection of conceptual criteria to evaluate the algorithms. Based on a literature review, the key criteria for a suitable algorithm were summarised at a higher conceptual level as 1) maximum parsimony of the algorithmic model, 2) ease of understanding, and 3) high performance. To evaluate the suitability of stop-move detection algorithms regarding these criteria, four algorithms were selected: two fully meeting the criteria (POSMIT and SOC), one partially meeting the criteria (MBGP algorithm), and one which does not directly correspond to them (CandidateStops). The first and the second criterion seemed to be the most relevant ones in order to find a suitable algorithm. Considering these criteria, the MBGP algorithm seems best suited because it has a low number of 4 parameters, while including both spatial and temporal parameters. Additionally, its parameters are intuitive and thus also understandable for non-expert researchers. Based on these results, it can be stated that performance should not be the main criterion when evaluating stop move-detection algorithms but rather their set of parameters and their conceptual complexity.

In order to find the most suitable algorithm for a given task, the candidate algorithms must be comparable and therefore, RQ2 was directed towards the definition of an appropriate set of evaluation measures. In this thesis the following performance indicators were selected from the literature: F-measure, average number and variance of stops per day, robustness to noise and varying sampling rates, as well as compactness of shape. Since the calculation of F-measures requires ground truth, 90 days of data were manually labelled and used to calculate the percentage of valid results using F-measure (i.e., for how many days an F-measure could be calculated). As it can be assumed that no single algorithm outperforms all others in all aspects, three scenarios were worked out, each responding to one of the following requirements: 1) best handles noisy data, 2) works well with varying sampling rates, and 3) has the most compact stop region shapes. Despite the fact that the results of the four algorithms were very different in terms of percentage of valid results, the average number and variance of stops per day were similar. The MBGP algorithm seemed to suit the MOASIS data best in two out of

the three proposed scenarios (i.e., Scenario 1 and Scenario 2) as well as regarding the percentage of valid results and the average number of stops per day. The POSMIT algorithm suited the MOASIS data best in terms of the third scenario. The least suitable algorithm for MOASIS data seemed to be the CandidateStops algorithm suggesting that it is important that stop-move detection algorithms have at least a spatial and a temporal parameter.

RQ3 was asking whether and how the spatial and temporal parameters affect the results of the detected stops and moves differently: while the temporal parameter of the SOC algorithm does not strongly influence the results (i.e., median F-measure values are the same but interquartile ranges differ slightly), the spatial parameter does. For threshold values greater than 75 m, results could not even be generated. The MBGP algorithm showed as well that at least one of the temporal parameters does not influence results at all as the different T_{max} values led to identical results. The spatial parameter D_{max} on the other hand affects the results more, as some tested parameter settings generated very low percentages of valid results. These findings suggest that the SOC algorithm and the MBGP algorithm are only sensitive to their spatial but not – or at least only to a lesser degree – to their temporal parameters. The results further revealed that the CandidateStops algorithm showed considerable differences in performance for the different speed variations used for testing. The POSMIT algorithm was also sensitive to its input parameters. Other than stated in the literature, the density-based MBGP algorithm was least sensitive to variations in the number of input data points and the temporal sampling interval.

Besides testing the sensitivity of the algorithms, the thesis, mainly RQ4, addressed the characteristics of the stops detected by the algorithms. The stops' shape compactness was determined using Ebdon (1985)'s shape index value (in Kitchin and Tate, 2000). All values were closer to 0 than to 1 and therefore all algorithms detected not that compact shapes. However, post-processing the data improved the shape compactness. This indicates that more smaller stop points were found (especially for the only pre-processed data) which could be a reason for the high number of false positives that the algorithms detected. Based on the data range of the shape index values, the CandidateStops algorithm has the least compact and the POSMIT algorithm the most compact shapes.

To sum up the thesis, one can observe that five out of six hypotheses could be rejected. As these hypotheses were stated on the basis of claims reported in the literature, this general finding underlines the importance of empirical, comparative evaluation of different algorithms to put the results reported in individual publications into perspective. In short, probability-based algorithms do not handle varying sampling rates better than the other three algorithms. Density-based algorithms are not more sensitive to the temporal input parameters and do not detect more compact stops than the other algorithms other than stated by Bermingham and Lee (2018) and Gong *et al.* (2015). The hypothesis that could not be rejected was that in general, more false positives than false negatives would be detected by all the algorithms. The algorithms often detected several stops instead of one very long stop. This might be due

to badly chosen parameter thresholds; the data might not have been processed well or the algorithms are not able to detect very long stops as one continuous stop.

Additionally, an algorithm evaluation without reliable ground truth is not recommended because it makes a significant difference to the result whether the F-measure value is 0.4 or 0.7. If the ground truth cannot be assumed to be accurate, the resulting F-measure is of limited value. The alternative, ground truth based on a probability formula, worked very well for two out of four algorithms. An evaluation criterion that was not found to be that important in this thesis is the running time, since there is room for improvement in the implementation of the algorithms (e.g., with parallelization or an implementation in another programming language).

In most of the individual tests conducted in this thesis, and therefore in the overall evaluation as well, the MBGP algorithm performed best. The parameter set for the MBGP algorithm showing the most promising results and therefore found most suitable for the MOASIS data is: D_{max} = 275 m, T_{max} = 14400 s, and T_{min} = 900 s (where D_{max} represents the maximum distance of the stop cluster's diameter, T_{min} represents the minimum duration between the points in order to count as stop, and T_{max} represents the maximum allowed time gap between the points of the same stop cluster). The POSMIT algorithm detected the most compact shapes. The CandidateStops algorithm performed worst, leading to the conclusion that a good algorithm should contain some kind of smoothing, since GPS data can be expected to always contain noise. Furthermore, pre- and post-processing should be adapted to the strengths and weaknesses of the individual algorithms, since post-processing was sometimes almost counterproductive to the result. However, this would make the algorithms' results no longer comparable in the sense of an evaluation across algorithms. The fact that the results of the MBGP algorithm are not noticeably better than those of the other algorithms is due to certain limitations of this study.

7.2 Limitations

One of the most serious limitations of this thesis was the rather small number of test sets used, as stops and moves had to be labelled manually in a post-hoc process due to missing actual ground truth. As long as there is no actual ground truth, one is not sure if the manually labelled ground truth really matches the actual ground truth or if the algorithms even detected the stops better than the person performing the post-hoc labelling. As this question remains unanswered, finding a reason for the small percentage of valid results is also limited. Furthermore, only four algorithms were selected, implemented, and evaluated in this thesis. As the literature study has shown, many more algorithms have been proposed, some of which might actually perform better in practice than in theory, as this study has shown, leading to the rejection of five out of six hypotheses based on reported findings and claims found in the literature. One example is the, according to the literature, moderately suitable MBGP algorithm that outperformed the purportedly more suitable SOC and POSMIT algorithms. Although the MBGP algorithm suits

MOASIS data best out of the four algorithms, it does not mean that exactly this threshold combination or even this algorithm are most suitable for analysing the mobility patterns of older adults collected in other studies. As the comparison with Fillekes, Kim, *et al.* (2019)'s sensitivity analysis, using the MBGP algorithm but a slightly different data set, revealed, the MBGP algorithm's T_{max} and D_{max} values determined in this thesis were more than twice as high. Indeed, the algorithm parameters will always have to be tuned to the characteristics of the particular data used. All the more important it then is that the algorithm is parsimonious regarding its parameters and intuitively understandable.

7.3 Future Work

In the future, the analysis of stop-move detection algorithms for GPS data of older adults should be further refined. Future work should concentrate on maximizing the percentage of valid results since improving the F-measure values is not the priority as they are high in general. To that end, collecting more reliable and accurate ground truth data should be pursued. Ideally, this would involve labelling of stops while the mobility activity is performed. If post-hoc labelling has to be used, several persons should carry out the labelling, which would allow determining the interrater agreement as introduced in *Section 6.1.1*.

The evaluation results could be optimized by pre- and post-processing the data differently per algorithm. The chosen post-processing approach of this thesis could be improved through grid search, by iterating through multiple, different parameter combinations. Furthermore, research should be done to find out how algorithms (e.g., the MBGP algorithm that overall performed best) perform differently depending on the pre- and post-processing methods applied. Since the MBGP algorithm is only the best out of four tested algorithms, other algorithms proposed in the literature might be evaluated using the methodology proposed in this thesis and be compared to the results of the MBGP algorithm. The evaluation R-script can easily be adapted to accommodate different algorithms. An alternative algorithm that could be suitable for the MOASIS data might be the 'Novel kernel-based algorithm' proposed by Thierry, Chaix and Kestens (2013), because it has both spatial and temporal parameters and, according to the literature, handles noise and sparse trajectory data well, which is particularly important for the noisy MOASIS data. The algorithm is also easy to understand, because it has only two parameters, kernel bandwidth and minimum stop duration. Other than the four selected algorithms, it builds a kernel density surface instead of analysing the points sequentially. If the focus should lie on only detecting specific stops of large spatial (and temporal) extent, the SMoT algorithm by Alvares *et al.* (2007) would be recommended, as it takes candidate geometries as input.

Many detected stops that the four algorithms classified wrongly, might be due to noisy data or unsuccessfully chosen parameter combinations. Therefore, it could be interesting to see how the algorithms handle different kinds of noise by applying different noise models to the data.

Since IMU data were also collected during the MOASIS research project, combining GPS and IMU data for finding stops and moves in trajectories correctly could be another approach to improve the stop detection results. The limitations of the GPS sensor (e.g., low signal accuracy in urban areas or heavy drain of battery) could be overcome with the IMU data as even small scale activities such as housework can be derived from accelerometer signals (Ellis *et al.*, 2014). Usually, transport mode detection results (also including stop detection) using GPS and IMU data in combination are better than if GPS or IMU data are used alone, respectively (Cetin, Ustun and Sahin, 2016; Ustun, 2016). One algorithm that combines GPS and IMU data and uses machine learning for transport mode detection was proposed by Ellis *et al.* (2014).

Algorithms based on machine learning could be also useful for only using GPS data, as the algorithms often detected several stops instead of one very long stop. Assuming that it is not due to badly chosen parameters as mentioned above, more complex algorithms could be tested, which would adjust the parameters according to the stop situation. However, it would be important, especially when developing new algorithms, to keep simplicity as a principle, because as shown in this study algorithms that were easy to understand performed better.

By finding the algorithm that suits MOASIS data best, a base has been established for further analysis of individual mobility and activities, so that conclusions about their state of health can be drawn and planning processes made more efficient.

8 References

Alvares, L. O., Bogorny, V., Kuijpers, B., de Macedo, J. A. F., Moelans, B. and Vaisman, A. (2007) 'A model for enriching trajectories with semantic geographical information', *Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems*, (i), p. 1. doi: 10.1145/1341012.1341041.

Ankerst, M., Breunig, M. M., Kriegel, H. P. and Sander, J. (1999) 'OPTICS: Ordering Points to Identify the Clustering Structure', *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 28(2), pp. 49–60. doi: 10.1145/304181.304187.

Aronov, B., Driemel, A., Van Kreveld, M., Löffler, M. and Staals, F. (2015) 'Segmentation of Trajcetories on Nonmonotone Criteria', *ACM Transactions on Algorithms*, 12(2), pp. 1–28.

Bereuter, P., Fillekes, M. and Weibel, R. (2016) 'Analyzing GPS and Accelerometer Data in the Study of the Mobility, Activity and Social Interaction of Older Adults', *AGILE Workshop on Visually-supported Computational Movement Analysis, 14 June 2016, Helsinki, Finland*. doi: 10.5167/uzh-130642.

Bermingham, L. L. (2018) 'From spatio-temporal trajectories to succinct and semantically meaningful patterns', *PhD thesis, James Cook University*. doi: https://doi.org/10.4225/28/5afa1141b90e5.

Bermingham, L. and Lee, I. (2018) 'A probabilistic stop and move classifier for noisy GPS trajectories', *Data Mining and Knowledge Discovery*. Springer US, 32(6), pp. 1634–1662. doi: 10.1007/s10618-018-0568-8.

Bösche, K., Sellam, T., Pirk, H., Beier, R., Mieth, P. and Manegold, S. (2013) 'Scalable Generation of Synthetic GPS Traces with Real-Life Data Characteristics', in Nambiar, R. and Poess, M. (eds) *Selected Topis in Performance Evaluation and Benchmarking - 4th TPC Technology Conference, TPCTC 2012, Istanbul, Turkey, August 27, 2012, Revised Selected Papers*. Springer Verlag Berlin Heidelberg, pp. 140–155.

Braune, C., Besecke, S. and Kruse, R. (2015) 'Density Based Clustering: Alternatives to DBSCAN', in Emre Celebi, M. (ed.) *Partitional Clustering Algorithms*. Springer International Publishing Switzerland, pp. 193–214.

Cetin, M., Ustun, I. and Sahin, O. (2016) 'Classification Algorithms for Detecting Vehicle Stops from Smartphone Accelerometer Data', *95th Annual Meeting of the Transportation Research Board and consideration for Publication in Transportation Research Record*, pp. 1–14.

Chaix, B., Benmarhnia, T., Kestens, Y., Brondeel, R., Perchoux, C., Gerber, P. and Duncan, D. T. (2019) 'Combining sensor tracking with a GPS-based mobility survey to better measure physical activity in trips: Public transport generates walking', *International Journal of Behavioral Nutrition and Physical Activity*. International Journal of Behavioral Nutrition and Physical Activity, 16(1), pp. 1–13. doi: 10.1186/s12966-019-0841-2.

Damiani, M. L. and Hachem, F. (2017) 'Segmentation techniques for the summarization of individual mobility data', *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(6), pp. 1–22. doi: 10.1002/widm.1214.

Damiani, M. L., Issa, H. and Cagnacci, F. (2014) 'Extracting stay regions with uncertain boundaries from GPS trajectories: A case study in animal ecology', *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 253–262. doi: 10.1145/2666310.2666417.

Dodge, S., Weibel, R. and Lautenschütz, A.-K. (2008) 'Towards a Taxonomy of Movement Patterns', *Information Visualization*, 7(3–4), pp. 240–252.

Domingues, R., Filippone, M., Michiardi, P. and Zouaoui, J. (2018) 'A comparative evaluation of outlier detection algorithms: Experiments and analyses', *Pattern Recognition*. Elsevier Ltd, 74, pp. 406–421. doi: 10.1016/j.patcog.2017.09.037.

Dongarra, J., Grigori, L. and Higham, N. (2020) 'Numerical algorithms for high-performance computational science', *The Royal Society*, 378(2166).

Ellis, K., Godbole, S., Marshall, S., Lanckriet, G., Staudenmayer, J. and Kerr, J. (2014) 'Identifying active travel behaviors in challenging environments using GPS, accelerometers, and machine learning algorithms', *Frontiers in Public Health*, 2(36), pp. 1–9. doi: 10.3389/fpubh.2014.00036.

Ewert, U. (2012) 'Senioren als Fussgänger', *Bern: bfu – Beratungsstelle für Unfallverhütung*, bfu-Fakten(08), pp. 1–16.

Fedorchuk, M. and Lamiroy, B. (2017) 'Binary Classifier Evaluation Without Ground Truth', *Ninth International Conference on Advances in Pattern Recognition (ICAPR-2017)*.

Fillekes, M. P., Kim, E.-K., Trumpf, R., Zijlstra, W., Giannouli, E. and Weibel, R. (2019) 'Assessing Older Adults' Daily Mobility: A Comparison of GPS-Derived and Self-Reported Mobility Indicators', *Sensors*, 19(4551), pp. 1–30. doi: 10.3390/s19204551.

Fillekes, M. P., Röcke, C., Katana, M. and Weibel, R. (2019) 'Self-reported versus GPS-derived indicators of daily mobility in a sample of healthy older adults', *Social Science and Medicine*. Elsevier, 220, pp. 193–202. doi: 10.1016/j.socscimed.2018.11.010.

Fleiss, J. L., Levin, B. and Paik, M. C. (2003) 'The Measurement of Interrater Agreement', in Shewart, W. A. and Wikls, S. S. (eds) *Statistical Methods for Rates and Proportions*. 3rd edn, pp. 598–626. doi: 10.1002/0471445428.ch18.

Fu, Z., Tian, Z., Xu, Y. and Qiao, C. (2016) 'A two-step clustering approach to extract locations from individual GPS trajectory data', *ISPRS International Journal of Geo-Information*, 5(166), pp. 1–17. doi: 10.3390/ijgi5100166.

Gong, L., Sato, H., Yamamoto, T., Miwa, T. and Morikawa, T. (2015) 'Identification of activity stop locations in GPS trajectories by density-based clustering method combined with support vector machines', *Journal of Modern Transportation*. Springer Berlin Heidelberg, 23(3), pp. 202–213. doi: 10.1007/s40534-015-0079-x.

De Graaff, V., De By, R. A. and De Keulen, M. (2016) 'Automated semantic trajectory annotation with indoor point-of-interest visits in urban areas', *Proceedings of the ACM Symposium on Applied Computing*, pp. 552–559. doi: 10.1145/2851613.2851709.

Guidotti, R., Trasarti, R. and Nanni, M. (2015) 'TOSCA: TwO-Steps Clustering Algorithm for personal locations detection', *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*. doi: 10.1145/2820783.2820818.

Hirsch, J. A., Winters, M., Clarke, P. and McKay, H. (2014) 'Generating GPS activity spaces that shed light upon the mobility habits of older adults: a descriptive analysis', *International Journal of Health Geographics*, 13(1), pp. 1–14. doi: 10.1002/9781119199410.ch3.

Huang, Y.-L., Liu, R.-Y., Wang, Q.-S., Van Someren, E. J. W., Xu, H. and Zhou, J.-N. (2002) 'Age-associated difference in circadian sleep-wake and rest-activity rhythms', *Physiology and Behavior*, 76, pp. 597–603. doi: 10.1016/S0031-9384(02)00733-3.

Hwang, S., VanDeMark, C., Dhatt, N., Yalla, S. V. and Crews, R. T. (2018) 'Segmenting human trajectory data by movement states while addressing signal loss and signal noise', *International Journal of Geographical Information Science*. Taylor & Francis, 32(7), pp. 1391–1412. doi: 10.1080/13658816.2018.1423685.

Kant, E. (1985) 'Understanding and automating algorithm design', *IEEE Transactions on Software Engineering*, 11(11), pp. 1243–1253.

Karami, A. and Johansson, R. (2014) 'Choosing DBSCAN Parameters Automatically using Differential Evolution', *International Journal of Computer Applications*, 91, pp. 1–11. doi: 10.5120/15890-5059.

Kitchin, R. and Tate, N. (2000) 'Spatial Analysis', in *Conducting Research in Human Geography*. London: Routledge, pp. 156–210. doi: 10.4324/9781315661063-14.

Krähenbühl, A. (2014) ‘Vergleich von Methoden zur Detektion von Stopps in Trajektorien. Masterarbeit’, p. 121.

Lamiroy, B. and Sun, T. (2013) ‘Computing precision and recall with missing or uncertain ground truth’, *9th International Workshop, GREC 2011, Seoul, Korea, September 15-16, 2011, Revised Selected Papers*, Springer(7423), pp. 149–162. doi: 10.1007/978-3-642-36824-0_15.

Lee, J. G., Han, J. and Whang, K. Y. (2007) ‘Trajectory clustering: A partition-and-group framework’, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 593–604. doi: 10.1145/1247480.1247546.

Leśniak, A., Danek, T. and Wojdyła, M. (2009) ‘Application of Kalman Filter to Noise Reduction in Multichannel Data’, *Schedae Informaticae*, 17(18), pp. 63–73. doi: 10.2478/v10149-010-0004-3.

Li, M., Gao, S., Lu, F. and Zhang, H. (2019) ‘Reconstruction of human movement trajectories from large-scale low-frequency mobile phone data’, *Computers, Environment and Urban Systems*. Elsevier, 77, pp. 1–10. doi: 10.1016/j.compenvurbsys.2019.101346.

Li, W., Goodchild, M. F. and Church, R. (2013) ‘An efficient measure of compactness for two-dimensional shapes and its application in regionalization problems’, *International Journal of Geographical Information Science*, 27(6), pp. 1227–1250. doi: 10.1080/13658816.2012.752093.

MacEachren, A. M. (1985) ‘Compactness of Geographic Shape : Comparison and Evaluation of Measures’, *Geografiska Annaler. Series B, Human Geography*, 67(1), pp. 53–67.

Marcum, C. S. (2013) ‘Age Differences in Daily Social Activities’, *Research on Aging*, 35(5), pp. 612–640. doi: 10.1177/0164027512453468.

Montoliu, R., Blom, J. and Gatica-Perez, D. (2013) ‘Discovering places of interest in everyday life from smartphone data’, *Multimedia Tools and Applications*, 62(1), pp. 179–207. doi: 10.1007/s11042-011-0982-z.

Moreno, F., Pineda, A., Fileto, R. and Bogorny, V. (2014) ‘SMOT+: Extending the SMOT algorithm for discovering stops in nested sites’, *Computing and Informatics*, 33(2), pp. 327–342.

Nathan, R., Getz, W. M., Revilla, E., Holyoak, M., Kadmon, R., Saltz, D. and Smouse, P. E. (2008) ‘A movement ecology paradigm for unifying organismal movement research’, *PNAS*, 105(49), pp. 19052–19059.

Newson, P. and Krumm, J. (2009) ‘Hidden Markov Map Matching Through Noise and Sparseness’, *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*, pp. 336–343. doi: 10.1145/1653771.1653818.

Nogueira, T. P., Braga, R. B. and Martin, H. (2014) 'An ontology-based approach to represent trajectory characteristics', *Proceedings - 5th International Conference on Computing for Geospatial Research and Application, COM.Geo 2014*, pp. 102–107. doi: 10.1109/COM.Geo.2014.22.

Nogueira, T. P., Martin, H. and Andrade, R. M. C. (2017) 'A statistical method for detecting move, stop, and noise episodes in trajectories', *Proceedings of the Brazilian Symposium on GeoInformatics*, pp. 210–221.

Ohlsson, E. (1998) 'Sequential poisson sampling', *Journal of Official Statistics*, 14(2), pp. 149–162.

Ovaska, S. J. and Sztandera, L. M. (2002) *Soft Computing in Industrial Electronics*. Springer-Verlag Berlin Heidelberg GmbH.

Palma, A. T., Bogorny, V., Kuijpers, B. and Alvares, L. O. (2008) 'A clustering-based approach for discovering interesting places in trajectories', *Proceedings of the 2008 ACM symposium on applied computing, SAC '08*, pp. 863–868. doi: 10.1109/ICICTA.2009.569.

Parent, C., Pelekis, N., Theodoridis, Y., Yan, Z., Spaccapietra, S., Renso, C., Andrienko, G., Andrienko, N., Bogorny, V., Damiani, M. L., Gkoulalas-Divanis, A. and Macedo, J. (2013) 'Semantic trajectories modeling and analysis', *ACM Computing Surveys*, 45(4), pp. 1–32. doi: 10.1145/2501654.2501656.

Park, S., Gil, M. S., Im, H. and Moon, Y.-S. (2019) 'Measurement noise recommendation for efficient kalman filtering over a large amount of sensor data', *Sensors*, 19(1168), pp. 1–19. doi: 10.3390/s19051168.

Posada, D. and Buckley, T. R. (2004) 'Model selection and model averaging in phylogenetics: Advantages of akaike information criterion and bayesian approaches over likelihood ratio tests', *Systematic Biology*, 53(5), pp. 793–808. doi: 10.1080/10635150490522304.

Rantanen, T. (2013) 'Promoting Mobility in Older People', *Journal of Preventive Medicine and Public Health*, 46, pp. 50–54. doi: 10.3961/jpmph.2013.46.S.S50.

Rocha, J. A. M. . R., Oliveira, G., Alvares, L. O. and Bogorny, V. (2010) 'DB-SMoT : A Direction-Based Spatio-Temporal Clustering Method', *IEEE Conf. of Intelligent Systems*, pp. 114–119.

Satopää, V., Albrecht, J., Irwin, D. and Raghavan, B. (2011) 'Finding a "Kneedle" in a haystack: Detecting Knee Points in System Behavior', *Proceedings - International Conference on Distributed Computing Systems*, pp. 166–171. doi: 10.1109/ICDCSW.2011.20.

SBB (no date) *HSR-C – Connection to High-Speed Rail Traffic*. Available at: https://company.sbb.ch/en/the-company/projects/national-projects/hsr.html (Accessed: 5 January 2020).

Schneider, C. M., Rudloff, C., Bauer, D. and González, M. C. (2013) 'Daily travel behavior: Lessons from a week-long survey for the extraction of human mobility motifs related information', *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. doi: 10.1145/2505821.2505829.

Schneider, W. and Hennig, A. (2008) *Lexikon Kennzahlen für Marketing und Vertrieb: Das Marketing-Cockpit von A - Z*. 2nd edn. Mannheim: Springer Berlin Heidelberg.

Schubert, E., Sander, J., Ester, M., Kriegel, H.-P. and Xu, X. (2017) 'DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN', *ACM Transactions on Database Systems*, 42(3), pp. 21, article 19. doi: 10.1145/3068335.

Silberholz, J. and Golden, B. (2010) 'Comparison of Metaheuristics', in Gendreau, M. and Potvin, J. (eds) *Handbook of Metaheuristics*. Springer, Boston, MA, pp. 625–640.

Swiss Confederation (2016) 'Demografischer Wandel in der Schweiz: Handlungsfelder auf Bundesebene Bericht des Bundesrates in Erfüllung des Postulats 13.3697 Schneider-Schneiter', pp. 1–86.

Swiss Confederation BFS (2012) 'Freizeit', *ValeurS - Ein Informationsmagazin des Bundesamtes für Statistik*, (1), pp. 1–28.

Thierry, B., Chaix, B. and Kestens, Y. (2013) 'Detecting activity locations from raw GPS data: A novel kernel-based algorithm', *International Journal of Health Geographics*, 12(14), pp. 1–10. doi: 10.1186/1476-072X-12-14.

Tran, L. H., Nguyen, Q. V. H., Do, N. H. and Yan, Z. (2011) 'Robust and Hierarchical Stop Discovery in Sparse and Diverse Trajectories', *Technical report EPFL*, pp. 1–10. Available at: http://infoscience.epfl.ch/record/175473.

Ustun, I. (2016) 'Robust Algorithms for Estimating Vehicle Movement from Motion Sensors Within Smartphones', *Doctor of Philosophy (PhD), Dissertation, Modeling Simul & Visual Engineering, Old Dominion University*, pp. 1–155. doi: 10.25777/8fqh-vc04.

Vazquez-Prokopec, G. M., Stoddard, S. T., Paz-Soldan, V., Morrison, A. C., Elder, J. P., Kochel, T. J., Scott, T. W. and Kitron, U. (2009) 'Usefulness of commercially available GPS data-loggers for tracking human movement and exposure to dengue virus', *International Journal of Health Geographics*, 8(68), pp. 1–11. doi: 10.1186/1476-072X-8-68.

Voelcker-Rehage, C., Godde, B. and Staudinger, U. M. (2011) 'Cardiovascular and coordination training differentially improve cognitive performance and neural processing in older adults', *Frontiers in Human Neuroscience*, 5, pp. 1–12. doi: 10.3389/fnhum.2011.00026.

Wegener, I. (1989) *Effiziente Algorithmen für grundlegende Funktionen*. 2nd edn. Springer Fachmedien Wiesbaden GmbH.

WHO (2017) '10 Priorities towards a decade of healthy ageing', pp. 1–15.

Xiang, L., Gao, M. and Wu, T. (2016) 'Extracting stops from noisy trajectories: A sequence oriented clustering approach', *ISPRS International Journal of Geo-Information*, 5(29), pp. 1–18. doi: 10.3390/ijgi5030029.

Xiao, N. (2016) *GIS Algorithms: Theory and Applications for Geographic Information Science & Technology, GIS Algorithms: Theory and Applications for Geographic Information Science & Technology*. Edited by R. Rojek, M. Oldfield, K. Haw, and R. Leigh. SAGE Publications Ltd. doi: 10.4135/9781473921498.

Yan, Z., Parent, C., Spaccapietra, S. and Chakraborty, D. (2010) *The Semantic Web: Reserach and Applications - 7th Extended Semantic Web Conference, ESWC 2010, Lecture Notes in Computer Science*. Edited by L. Aroyo, G. Antoniou, E. Hyvönen, A. ten Teije, H. Stuckenschmidt, L. Cabral, and T. Tudorache. Springer-Verlag Berlin Heidelberg.

Zadeh Monajjemi, P. P. (2013) 'A Clustering-Based Approach for Enriching Trajectories With Semantic Information Using VGI Sources', *Master's Thesis Faculty of Geo-Information Science and Earth Observation of the University of Twente, The Netherlands*.

Zhang, W., Wang, X. and Huang, Z. (2019) 'A system of mining semantic trajectory patterns from GPS data of real users', *Symmetry*, 11(889), pp. 1–12. doi: 10.3390/sym11070889.

Zhang, Y., Liu, Z. L. and Song, M. (2015) 'ChiNet uncovers rewired transcription subnetworks in tolerant yeast for advanced biofuels conversion', *Nucleic Acids Research*, 43(9), pp. 4393–4407. doi: 10.1093/nar/gkv358.

Zhao, Q., Shi, Y., Liu, Q. and Fränti, P. (2015) 'A grid-growing clustering algorithm for geo-spatial data', *Pattern Recognition Letters*, 53, pp. 77–84. doi: 10.1016/j.patrec.2014.09.017.

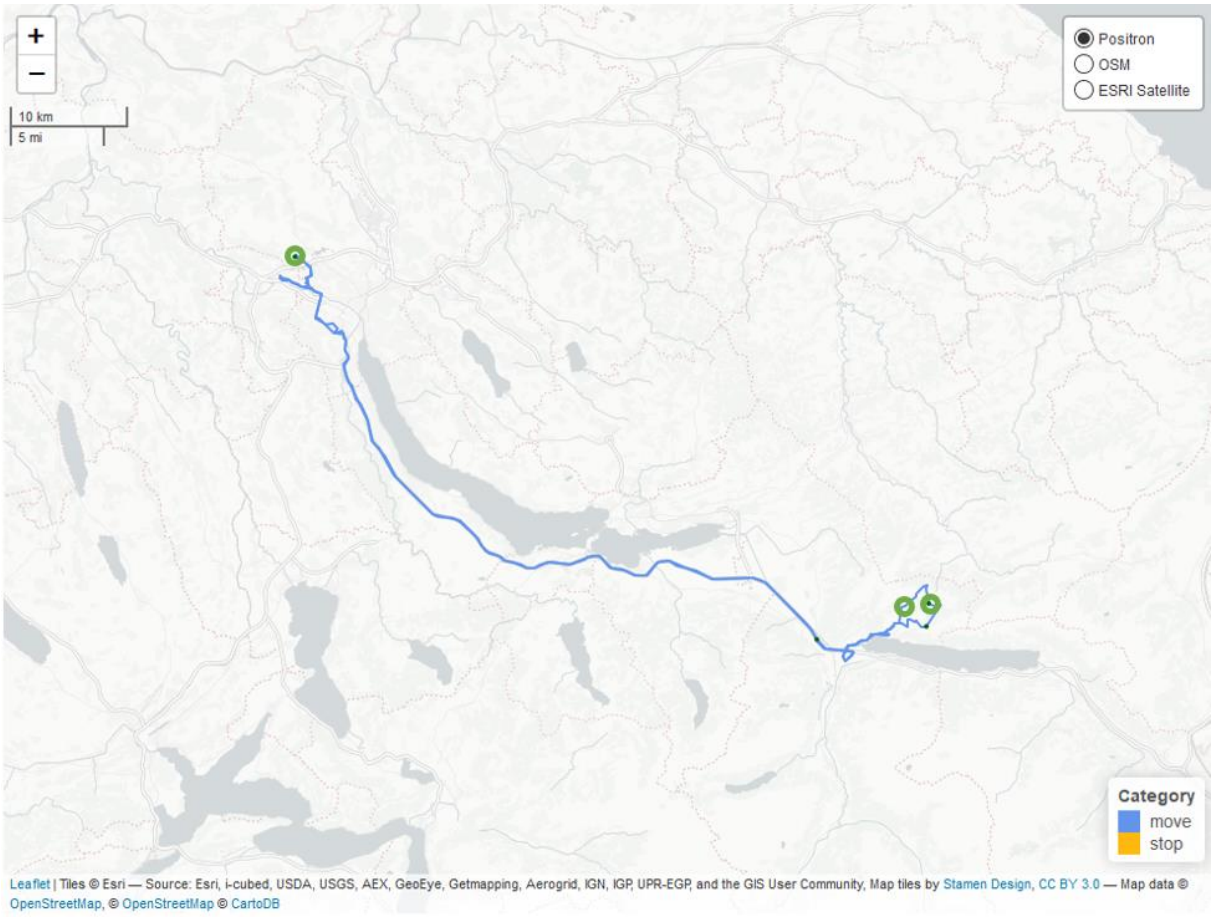
Zheng, Y. (2015) 'Trajectory Data Mining: An Overview', *ACM Transactions on Intelligent Systems and Technology*, 6(3), pp. 1–41. doi: 10.1145/2743025.

Zimmermann, M., Kirste, T. and Spiliopoulou, M. (2009) *Finding Stops in Error-Prone Trajectories of Moving Objects with Time-Based Clustering, Intelligent Interactive Assistance and Mobile Multimedia Computing - Communications in Computer and Information Science 53*. Edited by D. Tavangarian, T. Kirste, D. Timmermann, U. Lucke, and D. Versick. Rostock: Springer-Verlag Berlin Heidelberg.

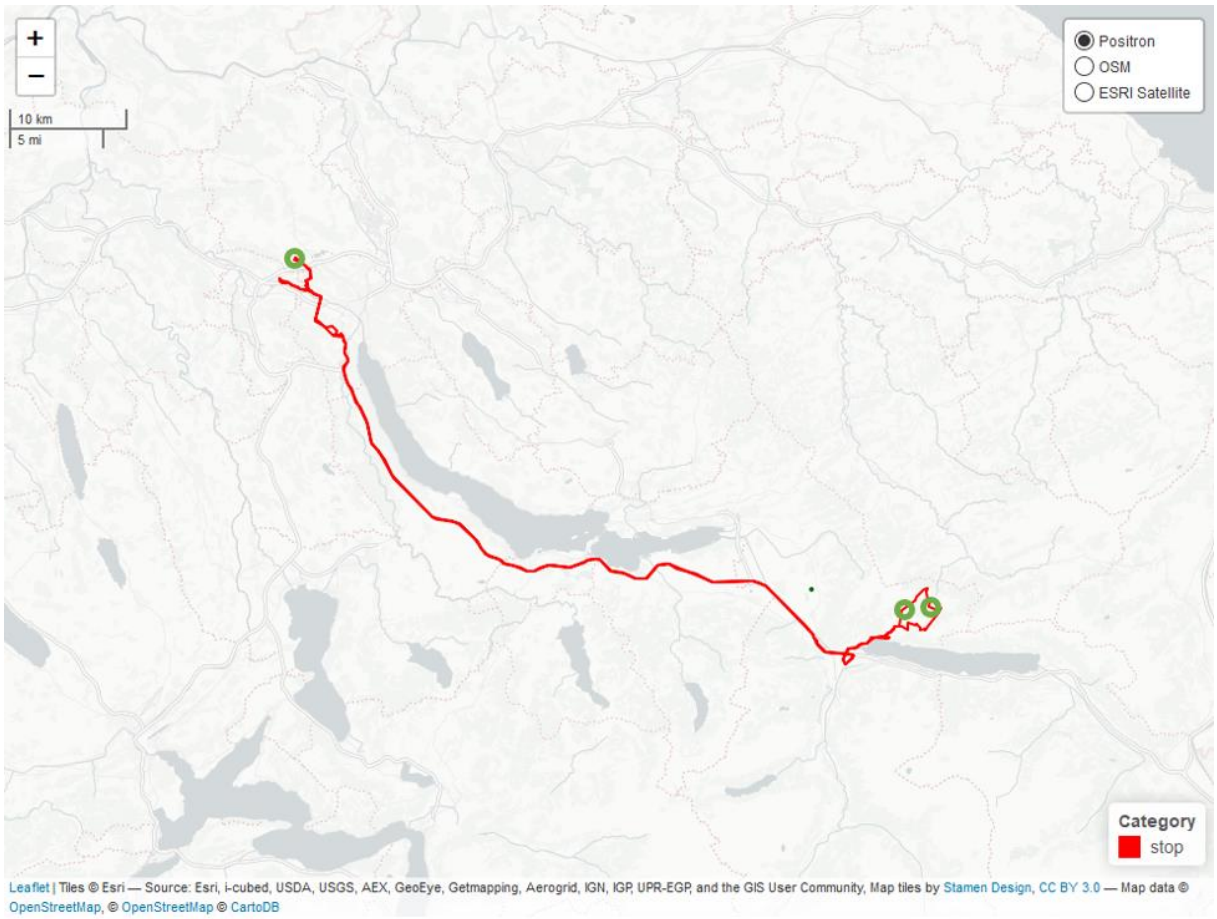
9 Appendix

9.1 Visualisations of Algorithms on Map

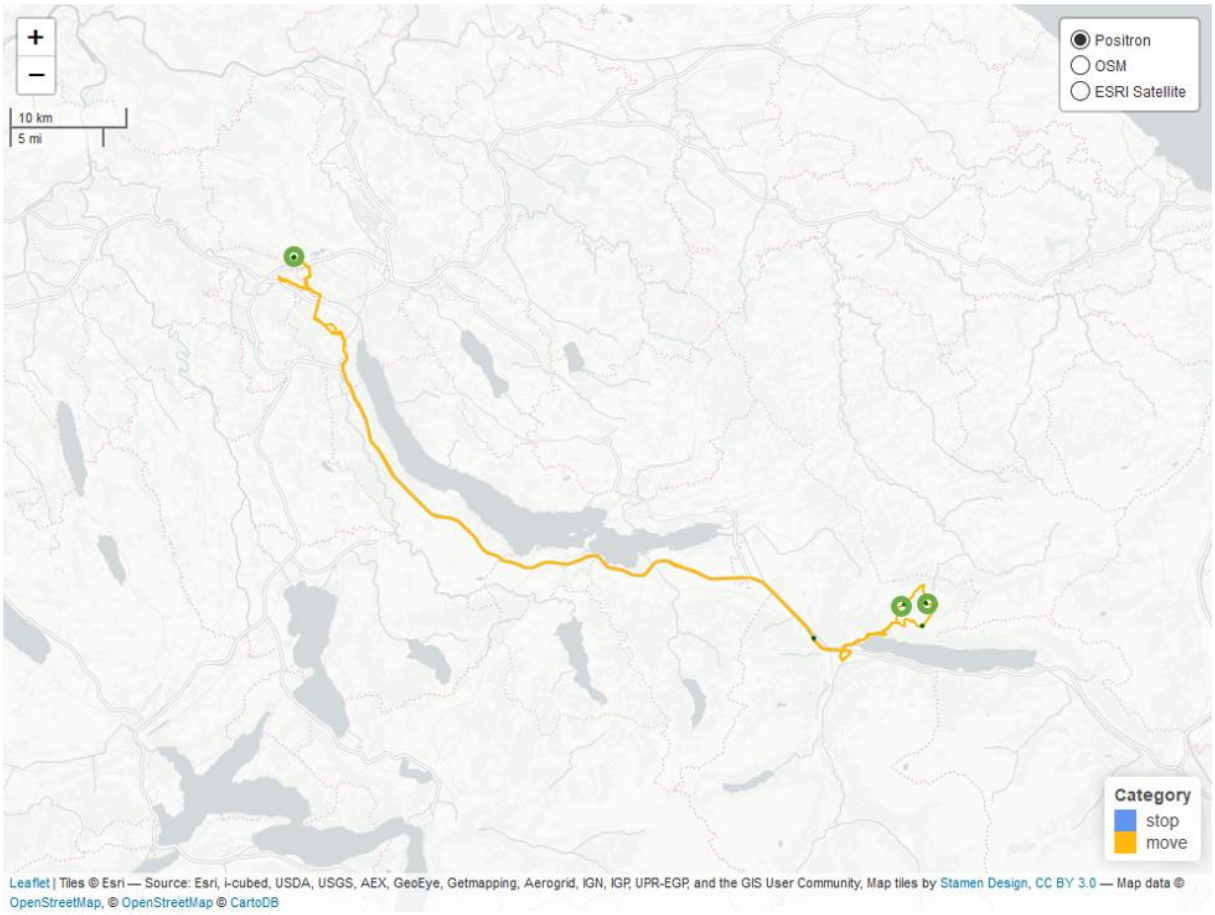
Appendix 9.1 shows the visualised results for one exemplary day. In bright green circles, the stops of the manually labelled ground truth were inserted in the maps. The centres of the stops, that the algorithms detected are coloured in small dark green points.



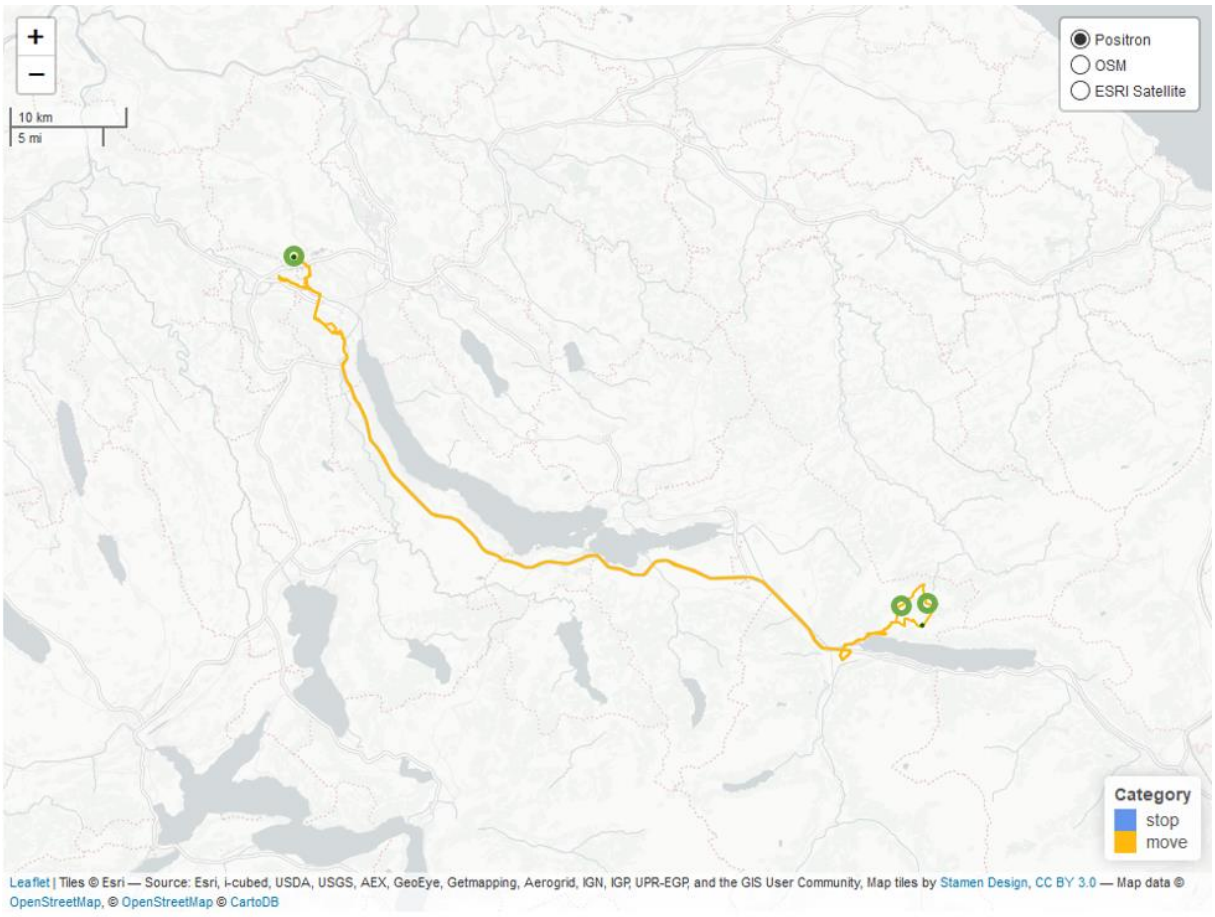
Appendix Figure 9.1 Visualisation of CandidateStops algorithm (pre- and post-processing delivered the same results)



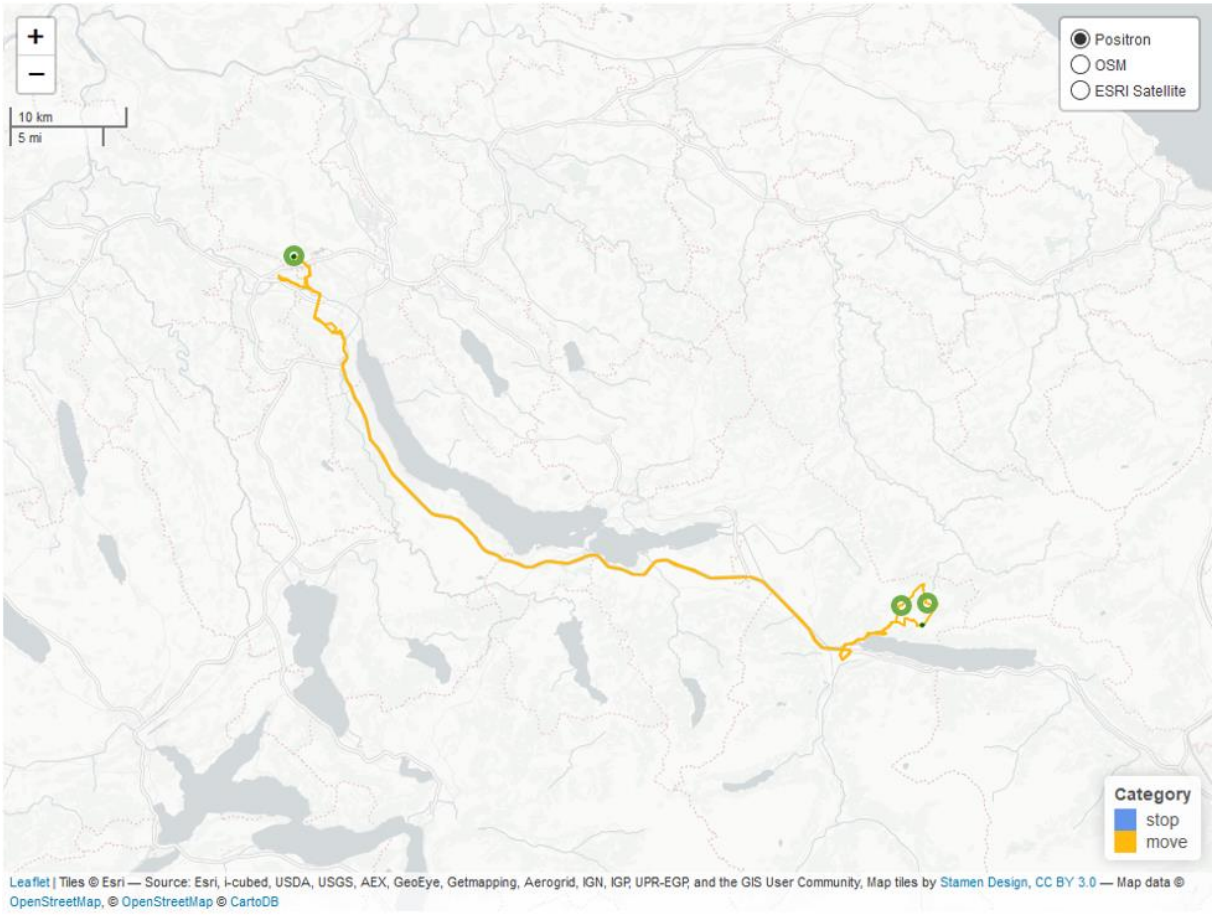
Appendix Figure 9.2 Visualisation of POSMIT algorithm (pre- and post-processing delivered the same results)



Appendix Figure 9.3 Visualisation of MBGP algorithm, T_{min} = 900 s,
(pre- and post-processing delivered the same results)



Appendix Figure 9.4 Visualisation of MBGP algorithm, T_{min} = 2100 s,
(pre- and post-processing delivered the same results)



Appendix Figure 9.5 SOC algorithm (pre- and post-processing delivered the same results)

10 Personal Declaration

I hereby declare that the submitted thesis is the result of my own, independent work. All external sources are explicitly acknowledged in this thesis.

Winterthur, 30.04.2020

Elena Ebert