Development of an unsupervised data-driven detection of GNSS outdoor-indoor transitions.

University of Turku
Department of Computing
Master of Science in Technology Thesis
Robotics and Autonomous Systems
Turku Intelligent Embedded and Robotic Systems (TIERS) Lab
July 2024
Henry Ernesto Ascencio Trejo

Supervisors:
MSc. Paola Torrico Moron
Prof. Tomi Westerlund

UNIVERSITY OF TURKU
Department of Computing

Henry Ernesto Ascencio Trejo: Development of an unsupervised data-driven detection of GNSS outdoor-indoor transitions.

Master of Science in Technology Thesis, 80 p.
Robotics and Autonomous Systems
July 2024

GNSS positioning systems are inside almost every device interested in getting their location in external environments. Recently, these devices have been performing tasks in environments inside infrastructures where they still require to know their location. Different methods exist and keep developing for indoor positioning, and outdoor positioning is practically solved using GNSS technologies. This work focuses on the scenario where devices benefit from a seamless transition between contexts. The seamless transition goal is to have uninterrupted access to the machine's position, ensuring success for the overall objective of the tasks it is performing. A tool helpful in accomplishing a seamless transition is detecting when the device is transitioning from an outdoor environment to an indoor one. With this transition detection, the device can prepare accordingly to avoid problems in challenging surroundings that limit the positioning system's capabilities. The method this thesis proposes involves machine learning to learn the distribution of outdoor data captured when the device interacts with this environment and later raises a flag when the conditions of the measurements change. The intent is to depend less on hard thresholds and adapt better to different locations while overcoming the challenges of data collection and labeling. The strategy relies on one-class support vector machines for their proven effectiveness with novelty and fault detection, along with delay embedding for its suitability to convert time series data to a set of vectors to accomplish the desired target. The resulting algorithm is evaluated in a series of trajectories covering an outdoor-to-indoor transition and portrays the functionality of the methodologies in fulfilling the objective. The evidence shows the potential and advantages of the process to make the detection. It also provides visibility on improvements and additions that can help integrate the algorithm into a general system to leverage the low need for manual configuration and high adaptability.

Keywords: GNSS, Unsupervised Learning, Anomaly Detection, Outdoor-Indoor Transition

Contents

List of Figures

1 Introduction

Positioning is the process of determining the location of an object according to a reference point in a given frame of reference, and it is a fundamental operation in many scientific research, industrial applications, commerce, and daily life [1]. The Global Navigation Satellite System (GNSS) is the most popular positioning system for localization and navigation tasks, especially for those in open-sky areas or outdoors. GNSS not only provides a solution for current problems, but it also has the potential to improve and evolve to power more sophisticated solutions emerging in the near and distant future [2].

One main component of a GNSS positioning system is the receiver. The GNSS receiver collects the signals from the satellites and processes them to provide the position (usually, along with velocity and time) solution to the user. There is a wide range of GNSS receivers available, with two main differential aspects being the quality of the solution they provide and the size of the component. These qualities allow GNSS positioning systems to meet their users' expectations and fit well with the surrounding subsystems in their solutions. Therefore, GNSS technology is the main choice for positioning solutions in outdoor open environments, and it is added to a variety of devices to satisfy the positioning requirements. Often, these devices perform operations also near or inside buildings. GNSS receivers face more difficulty providing accurate position solutions in environments that block or deviate the signals from the satellites [3].

To counter the performance downgrade of the positioning system, users fusion the GNSS information with an auxiliary method that can help to correct or eventually substitute it when the data from the receiver is not reliable anymore [4]. Nowadays, one active area of research is to achieve seamless transitions from outdoor to indoor environments [5]. Seamless transitions aim to exploit the complete capacity and benefits of GNSS positioning systems in outdoor environments, aid their performance with additional sensors while navigating through spaces between the outdoor and indoor frontier, and finally change the positioning system to one more suitable for indoor environments, where users can apply a great variety of technologies, each one with different strengths and weaknesses [6]. One component of seamless outdoor-indoor transition processes is detecting the environment's change. This detection alerts the device when the surrounding environment has changed (for example, from an open-sky area to an urban area with surrounding buildings) to activate actions that minimize the impact on position accuracy for the device. Detecting this transition enables devices to make the right decision on the navigation mode [7] they want to use in these different contexts, empowering their systems to exploit their advantages as much as possible in the right environments and use them efficiently throughout the whole navigation task.

Outdoor-Indoor detection solutions rely on external sensors (such as vision, light, and pressure) to assess the situation and provide feedback on the device's context [8], [9]. But they also rely on evaluating the performance of the GNSS receiver attributes [10]. Integrity monitoring in navigation systems is responsible for providing feedback about the ability of the system to provide accurate and integral solutions [11], and extensive research has explored different methods for its application, adapting to the increase of demand for GNSS systems in multiple applications. Leveraging the studies on GNSS Integrity, methods for outdoor-indoor detection emerge by connecting the faulty and abnormal behavior of the positioning system to the GNSS receiver

performance when entering environments that challenge the satellite signals [12]. Both the growing number of use cases and the wide range of parameters and agents involved in GNSS positioning systems represent a difficulty when building integrity monitoring systems, as the attributes constantly change from day to day, and relying on specific thresholds and measurement ranges limits the adaptability and longevity of the solution. Achieving better adaptability and handling the complexity of the GNSS raw data are objectives that machine learning techniques promise to tackle [13]. Specifically, when detecting faulty and abnormal behavior, machine-learning techniques can excel at learning to classify and differentiate data that complies with the expectations from data that indicates the presence of errors in a malfunctioning system [14].

This thesis presents the development and implementation of a method for detecting outdoor-indoor transitions by monitoring GNSS attributes. The basis of the detection is the capture of regular performance by an unsupervised data learning method to compare later measurements to identify and categorize them as inside or outside of the learned regular behavior. This research aims to design and implement a detection system that relies very little on prior knowledge and configuration to make the detection and can be used well in different environments by focusing solely on the degrading performance of GNSS positioning systems when making the outdoor-indoor transition.

1.1 Related Works

Kuusniemi in [15] covers the terms of integrity monitoring and reliability in satellite navigation with methods that seek to secure confidence in the solution of a positioning system and possible causes of disruption. Reliability testing is a method that determines if the basic postulates are still true in the system or if something has gone wrong with it. Usually, statistical tests are applied for reliability where the

null hypothesis corresponds to a fault-free system and the alternative hypothesis to a system with a fault present. A common approach when using this method is to perform a global test to identify if the measurements followed the assumed distribution, followed by a local test over the measurements to go into detail to identify any outliers and find the origin of the fault. Therefore, fault detection and exclusion schemes apply this process to improve the results of the solution.

Aviation applications use Receiver Autonomous Integrity Monitoring (RAIM) extensively to provide integrity and detect when a satellite failure has occurred [16]. One of the best attributes of RAIM is that it allows the GNSS receiver to detect errors without additional equipment, traditional RAIM can only detect one fault at a time, but improvements now allow for the method to asses multiple faults. Teunissen [17] already demonstrate a procedure applicable to GPS failure and integrity checking by checking the change in the mean of predicted residuals caused by data outliers or sensor failures. Pesonen [18] proposes a Bayesian framework for RAIM that focuses on urban navigation results in an improvement from traditional RAIM but requires a considerable amount of computation resources. Blanch et al. [19] present an advanced RAIM technique that handles multiple faults and a multi-constellation environment, overcoming the limitations of traditional RAIM algorithms. An evaluation of two fault detection and exclusion methods in [20] for GNSS systems working in urban environments presents the difficulty of consistency checks on satellite measurements in these contexts, highlighting the difference in the inconsistency between clean measurements and measurements blocked or deviated. Kim et al. [21] presents a solution relying on neural networks that expands RAIM to overcome its limitation by stating the integrity monitoring task as an anomaly check for unexpected behavior. Besides improvements in accuracy, other works focus on faster integrity checking. Zhang et al. [22] modifies the RAIM algorithm for a difference test statistic and procedure that requires less compute resources and

runs faster on devices. In addition, research in integrity checking in systems that integrate GNSS with other positioning technologies exists because of the increasing need for GNSS technology in urban environments. Sun et al. [23] develop a fault detection and exclusion algorithm in a GNSS and Inertial Measurement Unit (IMU) integrated system, tackling the need to detect multiple faults by combining IMU data with raw satellite measurements. Similarly, in [24], a GNSS and Ultra-wideband (UWB) navigation integrated system includes a fault detection algorithm that focuses on its performance in urban areas. The fault detection frameworks have been expanded to collaborative systems as well. Zhuang et al. [25] present a method based on cooperative reliability messages. This method expands the idea of using prior knowledge of nearby devices to assess the quality of GNSS raw measurements in another device. More recently, [26] introduced a method for slow-changing faults with a dual-threshold method in a GNSS and Internial Navigation System (INS) integrated navigation.

Research reflects vast strategies and methods for developing fault detection systems. Machine learning shows usefulness and advantages for GNSS applications [27] and fault detection systems [28]. A novel design of a fault detection algorithm using a neural network for an integrated navigation system improves the overall performance positioning accuracy in [29]. Biddle et al. [30] uses support vector machines (SVMs) to detect and identify faulty sensors in autonomous vehicle systems. SVMs appear in [31] to classify nominal and faulty phases in a unmanned aerial vehicle (UAV). A variation of SVMs, one-class support vector machines (OC-SVM), provides [32] a solution to identify unknown status and possible faults in chiller systems. Ma et al. [33] study OC-SVM for their fitness and adequacy for novelty detection with time series data, especially when projecting this data in phase space. Guo et al. [34] expand on the idea of phase-space projection to build a fault detection algorithm for a GNSS sub-system filter using a one-class support vector machine.

Another area where machine learning strategies are providing improvements to develop working solutions is outdoor-indoor transition detection in navigation systems, along with algorithms that prioritize thresholds and statistical tests. Zhou et al. in [8] present IODetector for location and context switching in outdoor-indoor environments. This detector is a lightweight sensing service that detects outdoor or indoor environments by taking advantage of sensing resources without assuming prior knowledge. Radu et al. [35] also use lightweight sensors on a smartphone to build a detection system for indoor and outdoor environments. Their method can adapt and learn online using machine learning. Other approaches choose to use GNSS measurements with or without additional sensors. Chen et al [12] use the number of visible satellites in the GNSS receiver to build an indicator of the indoor-outdoor status by performing a satellite existence search. Zhu et al. [36] design a method that extracts geometry distribution, time sequence, and statistical properties features from GNSS measurements to feed a supervised machine learning model to predict indoor-outdoor status. Xia et al. [37] describe four categories for environments in an outdoor to indoor transition (deep indoor, shallow indoor, semi-outdoor, and open outdoor). They show that scenario recognition improves with the availability of more GNSS constellations in the receiver and that using position-independent features for a recurrent neural network model allows high recognition accuracy. Research in [10] introduces a different approach for detecting outdoor, indoor, and transition areas using a time series analysis of different GNSS error statistics. They later use that detection for trajectory estimation and switching between GNSS and VIO for positioning. Siemuri et al. [7] reach for seamless navigation for indoor-outdoor environments. They use an indoor-outdoor detection strategy to choose the most adequate navigation mode.

1.2 Contributions

This work seeks to contribute an alternative solution for detecting the change in environments during that allows the system to act accordingly to reach a seamless transition from outdoor to indoor environments in positioning tasks. Specifically, this work makes the following contributions:

- An analysis of the GNSS receiver's measurements during an outdoor-to-indoor transition and justification of a suitable variable selection that helps to detect the context change.
- The study of applying an unsupervised machine learning method, such as one-class support vector machines, to GNSS data considering the proper treatment of time series data with delay embeddings.
- The design and implementation of an algorithm that raises a flag when detecting the transition between outdoor to close to indoor environments and its respective evaluation on real-world trajectories.

1.3 Structure

The structure of this work consists of the following chapters:

- **Chapter 2** gives a detailed explanation of the technologies involved in developing the method applied in the experiments.
- **Chapter 3** explains how each of the parts inside the algorithm works, the ideas and theory behind their use, and the implementation details to carry out the experiments.
- **Chapter 4** describes the experiments, the tools utilized, and the tasks performed. It gives a deeper look at the results and evidence of how the methods

work and the accomplishments of the algorithm.

- **Chapter 5** discusses the conclusions, additional improvements, and areas of opportunity found with this work.

2 Background

This work presents the design and implementation of a system that explores GNSS measurements in the receiver component to detect the transition from an outdoor to an indoor environment using machine-learning techniques based on the anomalies found in normal behavior distorted by the complications of the surrounding context. This chapter discusses the main topics and technologies that are involved in and support this development.

2.1 GNSS (Global Navigation Satellite System)

The Global Navigation Satellite System, or GNSS, is a collection of multiple satellite navigation systems [38]. The term can also refer to a single global satellite navigation system. GPS (U.S. Global Positioning System) is the most popular global satellite navigation system, but it is not the only one. Other global satellite navigation systems are GLONASS (The Russian Federation Global Navigation Satellite System), the European Galileo System, IRNSS (Indian Regional Navigation Satellite System), and BDS (Chinese BeiDou Navigation Satellite System) [39]. GNSS has been powering many navigation tasks over the years since its introduction, and its use for localization keeps growing. Sometimes, it is considered the undisputed method for determining position in outdoor environments [40]. Applications for GNSS positioning systems vary from big machines (like airplanes) to small devices (such as smartphones) [41]. This wide range of adaptability allows the technology

to appear in many different sectors with multiple options for accuracy, size, and accessibility. GNSS collections contain constellations. These are groups of satellites orbiting the planet to cover it entirely and to provide precise and continuous position information to the users of these constellations. Nowadays, users can leverage multiple constellations at once and compute better position solutions [42]. Besides satellites alone, these navigation systems work with passive receivers on the surface and monitoring networks to fulfill their goals. The main working principle for GNSS positioning systems is the concept of time-of-arrival (TOA) ranging [38]. This technique starts with an emitter sending a signal that includes its known location and specific emission time. Another device (the receiver) captures the signal, computes the time interval (propagation time) from emission to reception, and multiplies it by the signal propagation speed, obtaining the distance between the emitter and itself. At least four measurements (from four different satellites) are necessary to get a full latitude, longitude, and altitude solution [41].

2.1.1 GNSS Receivers

GNSS receivers are the components of a satellite navigation system that compute the solution to determine the user's position [16]. Therefore, they stay with the object of interest and constantly communicate with the collection of satellites in orbit. GNSS technology improvements allow receivers to compute accurate solutions and integrate into smaller systems. The progress on receiver development has also made the component more accessible for mass applications at the trade-off of being less precise with the solution [43]. Generally, all GNSS receivers face the same challenges and work under the same principles. The main problems affecting a receiver are signal blockage, signal attenuation because of the environment's conditions, noise interference, and multipath [38]. Choices in design help overcome these problems, making some receivers more robust and well-adjusted for specific use

cases. Nonetheless, some situations, like multipath in urban environments, require additional action from external systems that complement the receiver performance [44].

A GNSS receiver acquires signals from the satellites that contain replica code phase and replica carrier Doppler phase or frequency [16]. These measurements allow the receiver to compute the transit time and estimate the distance between the satellite and the device. They make this computation after synchronizing the local replica of the code or carrier phase with the one coming from the signal. After achieving the sync, the receiver keeps tracking the signal to maintain an accurate and stable lock. After this signal processing, the receiver gets valuable information to obtain the final solution [38].

The process receivers make before starting computing the position solution requires that the users wait for a complete start and fixation of measurements [45]. A cold start refers to the receiver needing to capture all the information from satellites and evaluate it before it allows signals to go through the following steps in the system. Faster starts are possible by allowing the receiver to save data for the satellites it expects to lock and fix. Warm and hot starts follow this procedure, and their difference is in the amount and type of data they have pre-loaded in the receiver before they start processing incoming signals from the satellites [46].

2.1.2 GNSS Measurements

Users can access GNSS measurements by interacting with the receiver. All receivers output the final solution to the user, and some also allow users to access intermediate measurements for different purposes [47]. Users might want to implement or research alternatives and improvements for the positioning algorithm [48], identify specific errors [49], or monitor the receiver's performance [50]. Here are definitions of some of the measurements of interest for this thesis:

Pseudorange Measurements

They are the direct result of the signal processing happening in the receiver and the basis for the positioning algorithms [51]. After the receiver locks and syncs with the incoming code phase, it can compute the transmission time of the signal from the satellite [16]. This information times the transmission speed gives the distance the signal has traveled or, in other words, the distance between the satellite and the receiver. The term "pseudo" comes from the synchronism errors between the satellite and receiver clocks. This apparent range is affected by many factors affecting the propagation of the signal in space [38]. For example, delays caused by crossing the atmosphere or by the instruments in the devices, multipath appearing due to the signal hitting other objects before the receiver, and noise appearing in the device [52]. Pseudoranges commonly appear in meters.

It is also frequent to find pseudorange residuals available from the receiver database. They have great use in integrity-checking statistical tests [53]. Pseudorange residuals result after the receiver builds a model that predicts the behavior of pseudorange measurements to estimate them and compares it against the actual pseudorange measurements it gets [54].

Carrier-to-noise density ratio (C/N_0)

Carrier-to-noise density ratio for a GNSS receiver represents a key parameter to measure the quality of an incoming GNSS signal [55]. It measures the ratio between the carrier power and the power of noise in a 1 Hz bandwidth. The fact that the ratio appears as a density over a unit of bandwidth differentiates this measurement from the signal-to-noise ratio and makes it ideal for comparisons due to the independence from the receiver bandwidth [56].

Inside the receiver, this attribute helps to determine if the system can lock and retrieve information from a specific satellite. A higher value indicates the receiver

can successfully extract the navigation data from the signal [57]. On the contrary, a lower value signals that the quality of the satellite signal does not allow for obtaining helpful information [58]. The ratio is a verified source of GNSS signal quality measure [38].

Dilution of Precision (DOP)

A receiver can relate the desired accuracy of position with the accuracy of pseudorange measurements by using Dilution of Precision metrics [16]. DOP provides an indicator for the distribution of the satellites visible by the receiver. A low value indicates a favorable configuration to receive optimal helpful signals, whereas a higher value indicates the opposite. This indicator aims to warn the user of the uncertainty of the final solutions and the confidence the system has in them [59].

Horizontal Dilution of Precision (HDOP) and Vertical Dilution of Precision (VDOP) indicate the quality of satellite geometry in that respective axis. Position Dilution of Precision (VDOP) combines the previous two metrics into one metric that summarizes the 3-D uncertainty. Time Dilution of Precision (TDOP) focuses on the uncertainty in the receiver's clock. An additional Geometry Dilution of Precision (GDOP) metric summarizes all other Dilution of Precision measures and reflects the effect of satellites' geometry on the final position solution [60].

Elevation and Azimuth

Receivers can obtain specific data about the satellites from the signals they acquire. Elevation refers to the degrees of elevation between the satellite's line of sight with the receiver and the earth's surface [61]. Azimuth measures the same interaction but indicates the angle between the satellite and the receiver or which direction to face to find the satellite [62].

GNSS Measurements can refer to individual satellites [63] or collective measure-

ments of visible satellites [64]. They can also depend on the specific position of the receiver and give different values under the same environmental conditions [65].

2.1.3 GNSS Disruptions and Errors

The GNSS signals radio frequency (RF) nature makes them susceptible to phenomena that affect their propagation [66]. Four main disruptions are responsible for downgrading the GNSS receiver performance, they are ionospheric scintillation, interference in the form of jamming and spoofing, signal blockage, and multipath effect [38] and the following paragraph introduce them.

Ionospheric scintillation is a phenomenon that fades the signal due to the eventual irregularities in the ionospheric layer of the earth's atmosphere. This disruption has the effect of diminishing the receiver's ability to track visible satellites for short periods [67]. Interference in GNSS systems appears when undesired and desired RF signals clash before reaching the receiver [68]. This interference can be unintentional by signals from emissions outside the band or intentional by external signals contaminating the respective band in which the GNSS receiver operates. The most common types of interference are jamming and spoofing [69]. The consequences of RF interference are navigation accuracy degradation and the complete loss of receiver tracking [70]. The surrounding environment of a GNSS receiver also has noticeable effects on its performance [71]. Signal blockage disrupts the quality and quantity of signals a receiver can acquire. Signals that face objects during their propagation can be absorbed and never reach the receiver or be reflected and suffer significant attenuation, making them unusable in the device [72]. Advances in GNSS technology allow signals to be functional even if they suffer some form of degradation, something critical in urban environments [73]. Another disruption caused by the receiver's context is multipath. Multipath refers to the multiple paths the signal duplicates take before reaching the receiver when reflected or diffracted [74]. This

deviation of the duplicate signals delivers a delayed signal (compared to the direct signal) and influences the solution negatively [68].

The disruptions presented above are the most considerable contributors to errors in the receiver and lower the solution accuracy. Additional errors can come from erroneous data about the satellite coming in the signal or due to malfunctioning of the satellite clock [16]. Corrections and auxiliary methods complement the GNSS receiver performance to reduce the errors present in the system [75]. Measurement errors are visible in the device by close monitoring of its values [76].

Carrier-to-noise density ratio (C/N_0) disruption in urban environments

Carrier-to-noise density ratio (C/N_0) value partly determines how well the receiver tracks the signals, becoming a key parameter in analyzing the system's performance [77]. Loss of tracking results affects position accuracy [78]. In environments with signal degradation due to foliage or blockage of natural signals from satellites, the carrier-to-noise density variable helps represent the power lost in the transmission [58]. Therefore, C/N_0 also plays a relevant role in spoofing or interference detection [79], [80].

2.2 Anomaly Detection

Anomaly detection refers to the problem of finding patterns in data that do not conform to expected or normal behavior [81]. Anomalies appear as outliers, exceptions, surprises, peculiarities, and even novel observations. The importance of detecting anomalies is the opportunity to identify events or information that require special attention under a specific process or situation. Many analyses aim to look for possible disruptions in the future or further investigations of past problems [82]. In these contexts, anomaly detection is a tool that allows them to focus their efforts on particular interests that are different from what is usual.

Many challenges arise when trying to detect anomalies. The first one is to find an optimal representation of normality or a definition of what is not an anomaly [83]. Sometimes, domain expertise alone is enough to bring this definition. In other situations, the complexity of a process and the involvement of many factors reduce the clarity and easiness of declaring what is ordinary behavior. Secondly, it depends substantially on the process' nature where the detection is or aims to happen [84]. Processes can change continuously over time, modifying their normal behavior and the range of values of their measurements. Some are more robust to support unexpected events than others, where a simple deviation from what is usual does not require special attention. On the contrary, other processes need constant monitoring for slight changes in their critical observations [85]. Finally, anomalies might be unknown to the analysts, and further steps are necessary to effectively target the appropriate variables and systems that help to detect an abnormal situation [86].

Outlier detection and novelty detection are two terms that are closely related to anomaly detection. They share some methods, techniques, and concepts to build solutions. Outlier detection refers to problems containing samples in the data already outside the pre-defined boundary of normality [87]. The main goal is to detect these outside samples during the data processing and use the examples as the basis for future detections. There is no general definition for what constitutes an outlier. It will depend on the use case [88]. Novelty detection aims to detect samples outside the original data boundaries without having information about them beforehand [89]. As observations arrive, the goal is to detect if these observations are novel when comparing them to the existing measurements. Choosing a method for anomaly detection depends on what approach is necessary and the context for our samples [86].

2.2.1 Anomaly Detection and Fault Detection

Technical processes often involve multiple systems working semi or fully automatically. Usually, these systems have limited human supervision and need to operate continuously for a long time. Different devices collect and share data from this process to monitor and help to supervise them. The objective is to ensure the system works optimally and without interruptions as much as possible or necessary. One key component to meet this target is fault detection [90]. Measuring variables and extracting features helps to identify unexpected or unwanted behavior. These detections help to restore the desired state and reduce the consequences of the fault in the system [91].

Much research has explored framing fault detection as an anomaly detection problem [92]–[94]. The main idea is to treat faults as abnormal behavior of the system's measurements or values, considering that not every anomaly might be a fault. One method in this approach is to use the samples acquired from sensors in the system. These observations are indirect measurements of the system's performance [92]. A classical approach to treating these observations is to build the distribution the data follows and later apply thresholds over the boundaries that limit normality [95]. Many existing techniques aim to estimate the data distribution, with the most notable difference being the assumptions they make over the possible distribution. Some systems' measurements or statistics follow assumptions very well, but others require fewer constraints to estimate the distribution accurately [96]. Some steps in the novelty detection process apply to every technique and are fundamental for the applications' success [97]. Primarily, feature extraction acts as an intermediary between raw signals in the system and meaningful variables for the following analysis. These features can be simple or complex, but they must carry relevant information that helps to detect abnormal or faulty behavior [98]. The next step looks into these features and starts asking questions that lead to concrete statements and decisions

over which ones are interesting and necessary to follow closely. A crucial tool to accomplish this goal is to visualize the features and how they change over some time or specific events [99]. After deciding on the features, a method that characterizes their distribution helps to know the probability of particular values and what we can expect from the features under the normal state. Finally, the decision boundary or threshold represents the classification of faulty samples as some measurements separate from the expected behavior [100].

2.2.2 Anomaly Detection and GNSS

One of the most developed methods for outlier detection in GNSS Applications is Reliability Testing [15]. It is essential to detect anomalies that can negatively impact the navigation solution and exclude them from the system. Typical detection for integrity reliability uses statistical tests over the receiver's observables to verify compliance with the initial hypothesis [53]. For these tests, redundancy is fundamental to verifying the consistency between the values inside a sample. RAIM (Receiver Autonomous Integrity Monitoring) is the most known example of satellite fault detection [101]. Initial RAIM methods handle single-failure scenarios using least-squares methods. The approach consists of discarding the outlier (or failure) in a group of measurements. Multiple tests are repeated to discard multiple outliers (if they exist) [102]. More robust methods that handle more than a singular fault due to the demand and usage increment exist nowadays. They can use weighting strategies over the observations, like the Danish Method [103], or greedy and exhaustive searches to exclude faulty satellites [20]. Other solutions rely on machine learning techniques to avoid handcrafted algorithms and rigid thresholds for outlier detection and exclusion of defective measurements [104].

2.2.3 Anomaly Detection Using Machine Learning

Machine Learning includes techniques and algorithms that adjust their parameters according to the data they interact with [105]. These statistical algorithms learn directly from the samples (inputs) they receive to output an adjusted algorithm that will comply with the pre-defined requirements. The main advantage of these methods is that they extract and identify patterns effectively in situations where handmade rules are not enough to accomplish a specific goal [106].

Multiple application domains have benefited from machine learning techniques [107]. These benefits are related to tasks where large amounts of information are available for processing, situations where extracting features and patterns is almost impossible for humans, and scenarios where adaptation is crucial and constant.

Machine Learning solutions rely heavily on the data they use to learn and adjust their parameters [108]. Collecting and inspecting this input data, known as a dataset, is the first and probably the most crucial step toward building the system [109]. Further steps depend on the specific method powering the solution, but some are part of every machine learning workflow. A clear definition of the target or goal for the algorithm is necessary to approach the problem correctly. Data preprocessing helps to ensure raw data is ready and suitable for the next phase. The training phase is the core part of the solution functionality. Training is giving the algorithm the samples and waiting for it to adjust its parameters to reach a specific target or run its complete process [105]. After training, the algorithm has learned what it could from the data and has its method for accomplishing the pre-defined goal [107].

The purpose for the algorithms to learn from data is not only to provide a representation of them to explore it deeply but also to analyze new data coming from the same distribution [110]. That is the reason for assessing the performance of the training phase. The evaluation and testing phase aims to provide meaningful metrics

that tell how close the training phase is to achieving the desired goal [111]. This phase focuses on providing the recently trained algorithm with never-seen samples and obtaining a result to check the solution's performance. These steps result in a model that can help validate assumptions, extract further insights, or analyze new samples [109].

The specifics of the previous workflow depend on which type of machine learning task describes the data and goal better. Supervised and Unsupervised learning are two types of machine learning tasks with specific methods and algorithms to carry out their functions [108]. The difference between the two is the access to previous knowledge about the answer for the solution or goal of the initial dataset (train dataset). Supervised tasks contain samples with features and responses for those features [112]. The algorithm then optimizes to reach those responses, and the evaluation uses new samples to corroborate how close the model response is to the actual response. On the other hand, unsupervised learning refers to problems with no response in the samples, and the main goal is to discover something of interest in the data [108]. The main challenge when building unsupervised learning solutions is to find a proper way to assess their performance [113].

Another task definition is semisupervised learning, and as the name suggests, it applies the supervised approach to just a subset of the samples, given that only those inputs have responses available [114]. In situations where a specific response is dominant or is too expensive to get the responses for all the examples, semisupervised learning strategies help to build working solutions [115].

Framing anomaly detection tasks using the machine learning framework helps emphasize the data and the discoveries available in it [116]. Another advantage is the adaptability for the proper identification of samples. Some solutions have access to data containing examples of normality and abnormality and aim to build a model that can generalize an automatic classification and explore the difference between

them. Other cases only use samples of expected behavior as outliers are challenging to capture [81].

Unsupervised learning methods can help to describe and build representations of usual samples when performing anomaly detection [117]. The principal assumption is that samples recorded under a working context contain little or zero anomalies, and samples outside the representation should be anomalies and require further inspection and attention. Unsupervised methods are helpful when anomalies take new or unexpected values constantly, and it is hard to obtain complete prior knowledge of how they will look specifically in every situation [118].

2.3 One-Class Support Vector Machine (OC-SVM)

Support Vector Machines (SVMs) are a popular technique for solving machine learning problems. They work well for multiple applications and perform exceptionally well in many fields [119]. The basic idea of SVMs is mapping the input data to a high-dimensional feature space and choosing an optimal separating hyperplane as the decision boundary [120]. In the initial implementations, they solve binary classification problems, building a decision boundary that separates positive and negative samples [121]. Subsequent progress allows SVMs to tackle multi-class classification problems and to define non-linear decision boundaries for complex patterns in the data using different kernels [122].

One-class classification problems arise when only samples for one class exist in the dataset, but samples for different labels can exist in the application environment [123]. Usual approaches for classification in machine learning suffer from imbalanced datasets [124]. Thus, in this scenario, it is better to utilize methods specifically tailored to tackle challenges in this context. The goal is to distinguish objects from the one class available from others belonging to other classes. One-class support vector machines (OC-SVMs) exist to provide an alternative for solving these kinds

of problems [125]. They rely on many concepts from SVMs but contain differences that enable them to build decision boundaries on datasets with samples from a single instance [126].

OC-SVM is a kernel-based method that maps the initial training data into a feature space (similar to normal SVMs), finds a hyperplane with a maximum distance from the origin, and separates a fraction of the training samples lying beyond it [127]. Although this is the frequent approach to building OC-SVMs, other solutions propose describing a hypersphere between the outliers and the positive class [128].

When using the first approach, the core parameter to solve the optimization problem is ν. This parameter can take values greater than 0 to less or equal to 1 [121]. ν represents an upper bound on the fraction of outliers and a lower bound on the fraction of support vectors [127]. This solution gives the unique hyperplane with the maximum margin between the data and the origin, putting the hyperplane closer to the origin than any point in the data [120]. The equation proposed to solve the optimization problem in [129] is:

$$\frac{1}{2}\|w\|^2 + \frac{1}{\nu N}\sum_{i=1}^{N}\zeta_i - \rho$$
$$\text{Subject to: } \langle w, \phi(x_i)\rangle \geq \rho - \zeta_i \ , \ i = 1, 2, ..., N \ , \ \zeta_i \geq 0$$

Where, as described in [130]:

- ζ_i are slack variables to model the separation errors.
- $\phi(x_i)$ is the non-linear function to project into the high-dimensional space
- ρ is the distance from the hyperplane to the origin
- w is the normal vector of the hyperplane

OC-SVM models only require samples from the target class, and have the possibility of using non-linear kernels for complex decision boundaries [130]. This make them suitable for problems in anomaly detection [131]. The established approach is

to collect training data corresponding to expected or usual behavior for the process and to label as anomalies the samples that lie on the other side of the hyperplane [132]. Beyond placing categories to the samples, OC-SVM also offers the distance of the samples to the hyperplane in the feature space. This distance can represent a score for further evaluation of how different samples are from the initial training data [133]. By tuning the hyperparameter ν it is possible to control the proportion of abnormal data in the training set and adapt the model to the specific requirement's detection [131].

2.4 Time Delay Embeddings

Time series data appears in many fields as measurements taken from a process where time is a dimension to consider. Because of the cruciality of these processes, detecting anomalies in the variables that defined them has been a relevant problem to solve historically [134]. Depending on the context, these anomalies can be a single or a group of measurements, indicating a long event or state change that deserves further attention and comparisons pondering the environment at the sampling time [135].

GNSS variables are usually acquired by sampling the measurements periodically. Due to this, strategies from time series analysis helps to examine them and power solutions that require that data for their goals. Although the time dimension is included inherently in the measurements, sometimes it is necessary to focus only on the magnitude of the values while still considering the extraction context [136]. Additionally, some techniques for anomaly detection in time series data require modifications to apply their methods to the temporal samples [137]. Transformations help to adequate the time series data for methods suitable for vectorized data [33]. One advantage of applying these transformations is leveraging the proven algorithms for anomaly detection to solve problems with another type of data [138]. One-

Class Support Vector Machines offer characteristics suitable for fault detection and notable performance in different fields [139]. It is an option worth considering for experimenting and building anomaly detection solutions.

Perkins et al. [33] describe a strategy to apply one-class support vector machines for novelty detection in time series data. They suggest unfolding the time series data into a new projection plane called the phase space using time delay embeddings is the most direct method to adequate the data to train OC-SVM in anomaly detection scenarios. The application of the delay embedding technique aims to reconstruct a picture of the samples in the phase space [140]. It could also represent the dynamics of a system in this new phase space by choosing the appropriate parameters [141]. Specifically, time delay embedding involves the augmentation of a scalar time series into a higher dimension through the construction of a delay vector with a delay lag τ and an embedding dimension m [142]. This method is similar to sampling measurements using a sliding window, with the embedding dimension being the number of measurements in the window and the delay tag the distance between neighboring measurements in time. The operation can be stated in the following way:

Given a time series $X(t)$, with $t = 1, 2, 3, ...N$ representing each time step, and N the total number of time steps in the series, and an embedding dimension m, and time delay τ. It can be unfolded for a sequence of vectors using a time delay embedding process:

$$X_i(t) = [X_i(t_i), X_i(t_i + \tau), X_i(t_i + 2\tau), ..., X_i(t_i + (m-1)\tau)]$$

- $(m-1)\tau$ refers to the window size.
- The difference between $X_i(t)$ and $X_i + 1(t)$ is called stride.

It is important to remark the authors in [33] emphasize using the phase space only in its mathematical form and ignore other meanings used in different fields.

Similarly, in this work, the technique is a helpful tool to eliminate the time dimension and place the measurements in a new projected space where OC-SVMs help solve the problem of novelty detection for fault detection. In this new phase space, usual measurements should appear closer, and abnormal ones should lay in a separate area.

3 Methodology and Implementation

The following chapter exposes the methods to build the solution for outdoor to indoor detection using one-class support vector machines over GNSS data. Along with this, the implementation details complement the ideas for a more concrete description of the tasks. The main goal is to explore a system that can make the transition detections accurately, adapt well to different environments, and reduce the amount of prior knowledge necessary to deploy it. This section explains the steps taken to reach a final working solution and the findings made along the way.

The methodology seeks to provide the techniques to understand what variables (or variable) help to detect the transition using what is available in the receiver and what strategies are suitable for building a system that can detect a transition from an outdoor to an indoor environment without relying on hard thresholds for better adaptation to different contexts.

3.1 Acquiring the GNSS receiver data

The work of this design and implementation proposes to use the data in the GNSS receiver to detect when the system carrying the positioning device moves from outdoors and gets closer to indoor environments. Naturally, the first step is to interact with the receiver and acquire all the helpful data it provides to achieve the detection.

The GNSS receiver chosen to get the data and perform the experiments is the ublox ZED-F9P module. This device is a high-precision multi-band GNSS module

[143]. One advantage of using this receiver is the data availability it has. Beyond the usual receiver data (for example, the position solution), it also allows the output of raw measurements from different observables at specific signal processing stages. Accessing the raw measurements is especially important to analyze variables that can have extended behaviors in the transition phase, contrary to losing a particular variable at the slightest disruption in the solution process. Variables without much processing are of service for inspection even when they suffer from signal-degrading environments and have much greater potential to indicate changes in the surrounding context.

Prior knowledge (presented in Section 2.1) and research [36] help to make an initial screening of the variables to explore. Although the selection at this stage is broad, doing so is favorable for analyzing the behavior of multiple observables related to the solution's performance. The main criterion for selecting a variable for the exploration phase is that it could be affected by the degradation of the incoming satellite signal, whether physically or computationally. Following the signal processing task in the receiver, observables closer to the input area are more explicit in their behavior. Whereas those close to the output usually have gone through correction and estimation processes that hide the abnormalities in the raw signal. This last scenario does not discard every variable at the final stages. Some measurements are specifically to keep track of the performance's quality or geometry between the satellites and the receiver, or they are error calculations of the solutions or intermediate estimates.

Considering the requirements and the receiver's data availability, the variables used for exploring the detection, along with the manufacturer description [144], are presented:

- **Carrier to Noise Density Ratio:** It is an indicator of signal strength. It helps to determine if the receiver can lock on to the carrier signal from a

satellite. Units in db/Hz.

- **Pseudorange residuals:** Refers to the difference between the observable pseudorange and the one computed from the estimated solution. It aims to calculate how close the estimation and the observation are. The data is in meters.

- **Doppler Measurement:** It captures the movement between the antennas on the satellite and the receiver. It also carries information about the rate at which the range between the satellite and the receiver changes. The measurements are either in meters/seconds or Hz.

- **DOP Measurements:** It is a value that indicates the quality of the geometry placement of the satellites, the receiver, and the clock offset. Horizontal, vertical, position, and time dilutions of precision help quantify the quality of the solution. The values are dimensionless.

- **Elevation:** A measurement of the vertical angle between the surface and the satellite. Units in degrees.

- **Azymuth:** A measurement of the horizontal angle between the satellite and the north. Units in degrees.

- **Number of visible satellites:** Amount of satellites interacting with the receiver. Not all of them are useful for the solution. The value is dimensionless.

3.1.1 GNSS Data Acquisition Implementation

The ZED-F9P module has multiple communication ports to connect with a host CPU. It provides UART, SPI, I2C, and USB interfaces [145]. All of them can output different message protocols as interfaces for the data contained in the receiver. Specifically, the receiver uses the National Marine Electronics Association (NMEA)

protocol and u-blox Propetary Protocol (UBX) to build most of its messages and interact with external devices. This implementation uses the USB interface and relies on the UBX protocol to get most of the variables, with some exceptions from the NMEA protocol.

By default, the module outputs general information through its USB port. Modifying the receiver's configuration is necessary to get the defined measurements for exploration. The modification requires modifying the config register in the device and setting proper values to meet the requirements. The recommended and most handy manner of changing the configuration register is to use the U-center tool [146]. This GUI application allows for easy manual setting of values for each item and load to the device. The configuration adopted in this implementation consists of the following:

- Only enabling the USB as a communication port, putting off the rest.
- Change the measurement rate for a specific number of updates per second.
- Activate the output of the messages containing the measurements of interest, allowing one message per epoch.

Once the receiver sends the desired data, the complementary part is to read and parse the messages in the host CPU. For that, a Python script runs to interact with the Serial port and saves the collected data in the system as a file. The core tool in this script, and for reading the messages from the module, is the pyubx2 package. pyubx2 is a Python library to parse messages for u-blox GPS/GNSS devices [147]. It allows effortless interaction with the information given by the device by parsing it into their respective protocol types.

3.2 Analyzing the GNSS variables

Although the receiver provides the necessary variables in its messages, most are unsuitable for direct exploration. The messages only transmit one variable at a time and do not describe the complete state of the receiver at a particular time. Additionally, the device outputs full messages that include a variable of interest but also other variables that are not required for the analysis. Due to this, the next step is to analyze the variables by applying two tasks to the acquired data. The first task is feature extraction, after which only the data of interest will be available for the next part of processing and performing the explorative analysis. After the second task, the objective is to define the variables for the model inferring the transition detection.

The initial challenge when extracting the features is how the device delivers consecutive messages. One sample of the measurements is the value of those measurements at a given time. That sample assumes all the observations happened simultaneously and represents the receiver's state at that particular moment. The state is complete when all the measurements taken together integrate the sample, and when one is missing, it should be null. A measurement from another instant should not be part of a previous or posterior sample, even if the samples miss the value to preserve the integrity of the state. The device does not send all the messages with all the variables in one transfer. Instead, it sends consecutive messages corresponding to a particular moment and waits until the next epoch to repeat the delivery. Thus, after reading the file containing the extracted data, it is necessary to build batches containing measurement groups for distinctive moments by leveraging the messages' cycle.

The batches of messages are filtered and prepared for the second challenge, which is to evaluate the adequacy of the variables for transition detection using explorative analysis. For this task, visual inspection of the measurements' behavior is crucial.

The idea is to recognize observables that react to the context environment change by inspecting how the time series changes (or not) while conducting an outdoor to an indoor transition. Once initial potential is recognized, statistical descriptors can help find patterns or consistencies between different scenarios. Even though after completing this task, there is no evaluation of how well the variable could work to detect the transition, it is relevant to select those attributes with potential and deserve closer inspection and continue the process.

3.2.1 Feature Extraction Implementation

The batches start with the ID of the first message, setting it as the mark for the completion of the cycle. A new list saves all the following messages. If that list already exists, it goes into another one before replacing it. After this process, we ensure that all the messages in one batch contain all the measurements describing the receiver's state at a particular time.

The goal of feature extraction is to get the exact attributes for the analysis from the messages in each batch. The starting point for achieving this objective is to create a Satellite object for holding the values of interest. A crucial attribute of the Satellite object is the satellite ID. The ID consists of the GNSS constellation number together with the satellite number. This object, specifically the satellite ID, helps keep track of existing satellites in the receiver messages.

The core feature extraction process starts by reading each batch of messages (receiver's state at a given time) alone. Then, for each batch, it goes through every message inside it. Once inside the messages, it is possible to extract the designated variables for analysis using the name they have in the manufacturer's documentation. A temporary list stores the variables in a specific order to later build a Satellite object using these variables as parameters for its attributes. If the satellite ID already exists in the general satellite storage, then only the new features

from that message are passed to this satellite. Otherwise, a new satellite goes into the general storage. Special attention is necessary for the collective attributes (they describe group characteristics) as they are collected once and then redistributed to every satellite in the batch. The process result is a list with many lists containing the batches. These batch lists have the satellites the receiver got at that particular time represented by their attributes. In terms of rows and columns, the rows are satellites, and the columns are the values of the observables for each satellite.

3.2.2 Explorative Analysis Implementation

The analysis looks at the receiver's state during the context transition through the variables captured in the data. Because the attention is on the receiver as a whole, the variables that describe individual satellite characteristics should help to reflect an overall picture of the interaction between the receiver and all the satellites. Summarizing the individual measurements also aims to have a value describing or representing the observable from the receiver's perspective. Reducing the number of features is positive when modeling the detection as long as the values are meaningful and carry prediction potential. Together with the summarization values, other variables already describing the complete picture of the receiver in their specific field are evaluated.

Statistical descriptors help to summarize the individual measurements. The descriptors chosen to represent the receiver's state better were the mean and the standard deviation. The reasons behind this selection consider the statistical but also the functional interpretation. The mean represents the average value across all the satellites and will move towards where most values are. The sensitivity to sparse values will not represent a problem as long as it is stable. In practical terms, if the context transition causes a considerable change in the measurement's magnitude, the number of visible satellites, or both, the mean will capture and reflect that

effect. In another way, the standard deviation will capture how dispersed the values from a group of satellites are. The focus here is to inspect how much the variation in the values changes after the transition. For example, we can explore if, with a large number of visible satellites, we can observe a larger sparsity in outdoor environments (receiving signals from poor and good satellite geometry) than in close-to indoor environments with less visible satellites giving degrading signals.

The function running the data pre-processing receives the file with the tabular measurement data and groups the measurements using the batch number. Then, for each sub-group, it takes the columns from the individual satellite measurements and computes the mean and the standard deviation. Finally, it merges these values with the collective receiver measurements and produces a single row representing the receiver state at that particular time.

Each batch is equivalent to a unit of time in the receiver's performance. The next step for the analysis is to plot each variable as a function of time in a scatter graph. These plots allow for a deeper inspection of the performance of each variable during the experiments. A visual examination helps to analyze the behavior of the observables clearly when conducting an outdoor-to-indoor transition. The expectation is to find changes that indicate the receiver has gone from one context to another. It is important to remark that the exact detection might not be reachable at this stage. Nonetheless, a clear or growing differentiation of the measurements when exiting the preferred outdoor location will indicate the potential for automatic transition detection. The analysis provides variables useful for further exploration of the detection model and insights into the strategy for the complete functionality.

3.3 Building the Transition Detection Model using OC-SVM and Delay Embeddings

The work in this design and implementation aims to use a machine learning model that gets the receiver's state at a specific time as input and determines if the device has transitioned from an outdoor to an indoor environment. The goal is to have a system simple to build and deploy that can adapt effortlessly to new contexts by relying on the data it captures. Using a one-class support vector machine model helps to reach the goal as it learns the distribution even from small samples in the training data. Pairing this model type with a projection of the input data in the phase space using delay embeddings allows it to take advantage of its characteristics for novelty detection to analyze time series data.

One of the perks of using data learning algorithms is their ability to generalize, meaning that they handle samples that follow the same distribution from the training input even when they have not seen them before. Nonetheless, to achieve notable generalization performance in contexts with high variation and sparsity in the data encountered, it is necessary to feed large amounts of data to get the complete representation of the expected distribution as closely as possible. This situation poses a challenge when data collection and labeling become difficult and expensive, and it is almost impossible to get most of the variety to ensure acceptable performance. Multiple strategies have been implemented in research to collect training data for detecting outdoor to indoor transitions using a GNSS receiver. However, the nature of the GNSS technology poses many variables that change the context for the measurements the model can receive from time to time. Due to this, this methodology aims to provide an alternative focusing on this challenge by selecting an algorithm that can perform well with a low number of training samples and does not require labeling them.

The procedure to assess and evaluate the model performance includes a previous step to collect data from known environments. Using data from the two states the model has to differentiate helps to analyze how well the model is doing and if the strategy is working as expected. An ideal scenario will be a model that correctly classifies the data from the usual or outdoor environments as the positive class and data from a different context (closer to indoors) as the negative class. This step also helps to discriminate variables to decide which observable is the best to utilize for the detection.

The steps to find the suitable strategy to build the detection model start with projecting the time series data of all the features into the phase space using delay embeddings. After that, the data is in vector form and suitable for training the one-class support vector machine models. The training will use each feature from the data of known environments to judge which produces better results to differentiate the different contexts. After analyzing the results, the implementation of the complete training and detection strategy will begin using data from scenarios where the device is performing a movement that involves transitioning from an outdoor environment to an indoor one. At the end of these steps, a method will emerge to provide a system that detects outdoor to indoor transitions using GNSS receivers' measurements.

3.3.1 Projecting the time series into the phase space using delay embeddings

The idea to project the time series data into a new plane, as mentioned in [33], is to make it suitable for using one-class support vector machines. Instead of having a continuous signal that keeps extending over time, a sliding window captures the state of the signal (embedding), placing that representation in the phase space using other dimensions. The data becomes a set of vectors that can serve as the model

input.

The method utilized to perform the projection is time delay embedding. This process consists of "delaying" the measurements of the time series to construct a higher-dimensional vector from them. The sliding window samples some measurements (embedding dimension) separated by a particular time between them (time delay) and keeps repeating for all available data. As a result, each window becomes a point in the phase space. In complex systems, the projection can have physical interpretations. In the context of this work, the projection will help to place the points over a plane to use a machine learning model and identify outliers.

Using the examples in other research [33], choosing time delay embeddings comes from having a method to represent a normal state from the receiver's measurements. The intention is to reproduce the device's state in a high-dimension plane where samples containing typical values lay together, allowing the model to learn its distribution straightforwardly. In contrast, samples from abnormal situations or deviations from the normality should be far from the others in the new projected plane. Concretely, for the use case of outdoor to indoor transition detection, the embedding process should help to capture the state of outdoor measurements with points staying close when sampled. Putting observations close to indoors or not in outdoors away from the initial open-sky ones.

Applying time delay embedding is simple but requires special attention when selecting the necessary parameters. Given a time series variable, one can extract a set of vectors containing samples from the time series separated by a specific time between them. The size of each vector in the set is called the embedding dimension m. Another way to describe this parameter is to see it as the amount of measurements inside the sliding window. The name embedding dimension refers to the number of dimensions of each individual vector in the set. The name for the separation between each sample in the vector is time delay τ. The separation refers

to how much time is between each measurement and is constant in each vector.

Selecting adequate values for these two embedding parameters is crucial. Some applications benefit from a particular choice to unfold the behavior or a whole representation of a system using one observable. In this work, the purpose of the representation is to portray the system's state at a particular time and context. Thus, the selection of the embedding parameters will surge considering the sampling rate of the messages coming from the receiver and the speed to analyze and deliver a response from the measurements. At the same time, the time it can take the device to notice a change in the environment will be part of the considerations. The initial approach is to use an embedding dimension large enough to capture one second of measurements in a vector and a time delay equal to the sampling rate of the receiver. The implementation of the time delay embedding consists of grouping the desired measurements in an array and placing all these vectors into a bigger one.

A final aspect of the projection is visualizing the resulting embedding. For lower dimensions, it is possible to plot it directly, and for higher dimensions, a dimensionality reduction technique helps to watch the new set of vectors. A visual inspection also adds insights and valuable information to understand the behavior of the variable.

3.3.2 Training a one-class support vector machine model for detecting outdoor to indoor transitions

Building the embeddings completes the data processing stage. The output of the data processing is ready to supply the inputs for modeling a novelty detection model that is useful for detecting the moment of the transition from outdoor environments to indoor ones. This work intends that the system computes the boundaries for detection from the data directly, relying less on prior knowledge or hardcoded thresholds. Two core challenges appear to accomplish this goal. The first is to evaluate the

suitability of the data-learning methods for the specific use case, and the second is to integrate the model into a complete process that analyses data during a transition.

The first strategy is collecting data from known outdoor and close indoor locations to evaluate how well the model behaves in this specific application. Measurements that reflect changes or differences between the two environments should help the model create a functional decision boundary and discriminate the values from different sources. This task will also help to study the limitations on the amount of data for training and the impact of hyperparameters on the data at hand.

The second strategy consists of acquiring data from actual trajectories while going from outdoor to close to indoor locations. The goal here is to study and decide on the complementary factors necessary to obtain the detection. The result is a series of steps to integrate modeling and detecting while considering all the data phases. This strategy will reflect aspects complementary to the detection alone but with a considerable impact on the final transition flag decision.

One key feature this thesis explores is to avoid the requirement of labels in the data for training the model. One part of solving that problem is to use a model like a one-class support vector machine, which only requires samples from the known target. The other is building the strategy to prepare the model before the receiver-carrying system needs to use it. The proposal to solve this second part is to collect training data while the device is known to be in an outdoor environment. This acquiring phase ideally would be after the receiver has fixed satellites and can compute acceptable position solutions. The detector system will dedicate a small amount of time to capture and process data in this scenario and prepare it to train the model. After that time, it will use that data for training, resulting in a model that has learned how the outdoor environment data behaves. The model will start to make inferences on new samples to evaluate whether they fall inside or outside the decision boundary for common (outdoor) measurements. Consecutive

negative samples should appear so that false detections or short transitions do not affect the final decision. Therefore, the method in this work can adapt to multiple locations. Nonetheless, it also carries other challenges and details that depend on precise scenarios. For example, the amount of data for training might not be a fixed parameter, and some scenarios need longer times to capture how the outdoor measurements behave.

Training and evaluation OC-SVM

The core of the detection system is the one-class support vector machine model. The objective of using this tool is to detect different data from the training samples. The algorithm projects the data into a feature space and finds a hyperplane that acts as a decision boundary between the positive and negative samples. The core elements to explore when training the model are the following:

- **A kernel function** that controls how the model behaves in the feature space. The gamma hyperparameter defines how much a singular training sample influences the solution. A high gamma value puts the decision boundary much closer to the support vectors, and a low value creates a more linear boundary.
- **The parameter** ν is particular to one-class SVMs. This value corresponds to the fraction of new regular samples outside the boundary.

The decision of the metric to evaluate the performance of the preliminary stage (using known environments) is to use accuracy. This metric will help to know how many correct detections on the data outside of the training samples the model has done correctly, given that the model only outputs one decision. A good performance exists when the model can classify most of the samples from the negative samples accordingly. Most of the effects of false detection are mitigated by waiting for consecutive sustained detections and not only single events.

OC-SVM Implementation

The package scikit-learn [148] provides the algorithm to build OC-SVM models. The script creates an instance and fits it using the training data. Once the model fits the samples, the same instance allows inference on new data. Positive labels go to data inside the decision boundary, and negative ones go to new samples dissimilar from the training data. For accuracy computation, the script grabs the number of negative detections and divides it by the number of samples in the indoor known environment.

Implementing the complete transition detection system

After working on the previous phases, putting them all together allows using the model for transition detection while the device is working in a real-world scenario. The intention is to provide the advantages of this approach without affecting the operation of the device carrying the positioning system. The steps for the transition detection system in outdoor to indoor trajectories are the following:

1. Collect raw data and process it to extract features that represent the receiver state.

2. Apply delay embedding to the time series data to create a new set of vectors.

3. Train a one-class support vector machine with data from a short window after the receiver confirms having fixed data.

4. After training, use the model with new data from the receiver.

5. Raise a flag after classifying consecutive samples as negative. The flag indicates that the device is no longer in the outdoor environment and has moved to a new one (closer to the indoor).

4 Experiments and Results

4.1 Hardware and Software

The equipment for the experiments consists mainly of the GNSS receiver and the computer host to process the device's messages. In addition, in the scenarios where trajectories are collected, a visual-inertial tracking camera is also part of the equipment to capture auxiliary data to corroborate the movement and location of the device.

The GNSS receiver is the ZED-F9P-02B-00 shown in Fig. 4.1, manufactured by u-blox [143]. The receiver is on Sparkfun's GPS-RTK-SMA breakout board. It provides an SMA connector to attach an antenna. The device connects to the host computer using a USB cable.

(a) Front View (b) Angle View

Figure 4.1: Pictures of the GNSS Receiver: ZED-F9P-02B-00

The visual-inertial tracking camera is the Intel RealSense T265 [149] shown in Fig. 4.2. The onboard fish-eye cameras and the Inertial Measurement Unit power the tool to provide position-tracking in almost any context. The second experiment uses this functionality to get the device's x and y coordinates to help verify the trajectory of the GNSS receiver.

(a) Front View (b) Angle View

Figure 4.2: Pictures of the Visual-interial tracking camera Intel RealSense T265

The software tools run in the host computer, and none has special requirements to run. The following is a detailed description of the tools and their usage:

- Customizing the GNSS receiver's configuration and messages: ***u-blox's u-center*** [146]
- Inspecting the GNSS receiver's performance: ***PyGPSClient*** [150]
- Capturing and parsing the GNSS's receiver messages in the host computer: ***pyubx2*** [147]
- Data processing tasks: ***NumPy*** [151]
- Data visualization: ***Matplotlib*** [152]
- Statistical computations and modeling: ***SciPy*** [153], ***Scikit-learn*** [148]

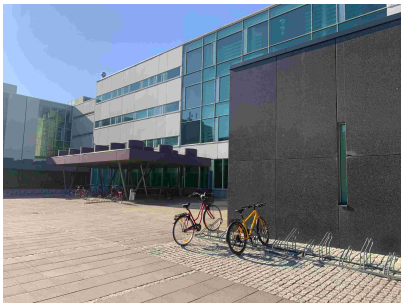
- Verifying delay embedding computations: ***GTDA (giotto-tda)*** [154]
- Extracting pose coordinates from the visual-inertial tracking camera: ***pyre-alsense2*** [155]

4.2 Locations

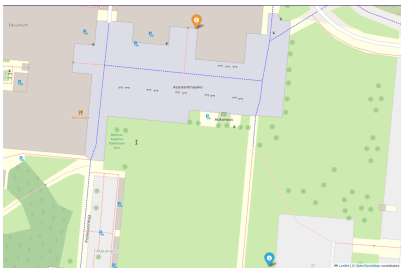
The places to conduct the experiments are suitable for following trajectories that start in an open-sky location and end close to the entrance of a building. The most important aspect is to simulate how the device will behave if an application or use case were working on an outdoor location and started transitioning to indoor.

- **Location 1**: Starts at an open-sky area in the middle of some buildings. Trees exist only on one side of the trajectory. At the end, reaches a roofed place in front of the building's entrance.

(a) Location 1 Starting Point



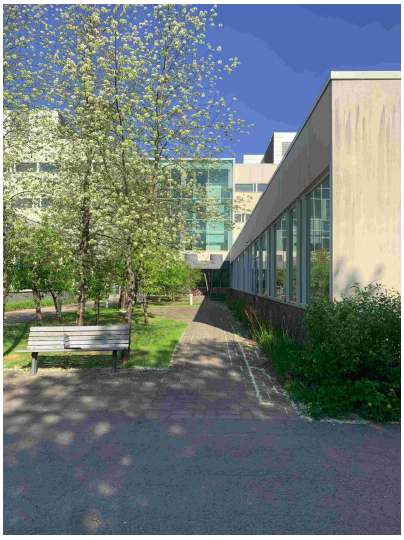
(b) Location 1 Finishing Point



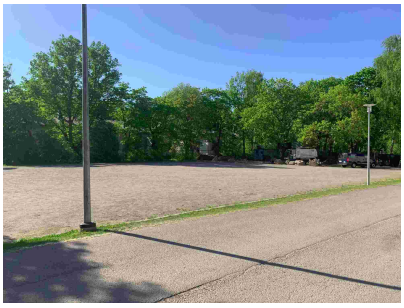
(c) Location 1 Map

Figure 4.3: Pictures of Location 1. In (c) the starting point is the blue marker, and the finishing point is the orange one.

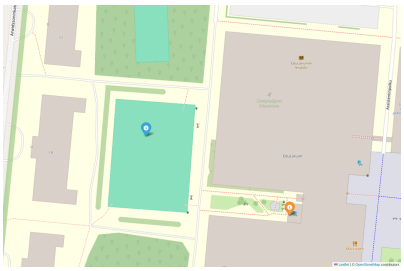
- **Location 2**: Starts in a parking lot without cars around it. Low height buildings and tress are around the starting place. It encounters trees and a building before reaching the entrance door.



(a) Location 2 Finishing Point



(b) Location 2 Starting Point



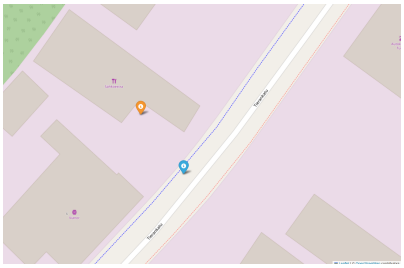
(c) Location 2 Map

Figure 4.4: Pictures of Location 2. In (c) the starting point is the blue marker, and the finishing point is the orange one.

- **Location 3**: It starts at the edge of the sidewalk and approaches slowly to the building entrance. Only low buildings surround the trajectory.

(a) Location 3 Starting Point

(b) Location 3 Finishing Point



(c) Location 3 Map

Figure 4.5: Pictures of Location 3. In (c) the starting point is the blue marker, and the finishing point is the orange one.

4.3 Experiment 1: Exploring the best feature and the model performance for detecting the transition

This experiment consists of using Location 1, visible in Fig. 4.3, to capture data from known environments. First, the receiver is in an open-sky location, representing the outdoor state. The data is collected containing messages with all the features selected for inspection. In this scenario, no movement happens, and all the measurements correspond to the identical location. Then, the receiver is in a place close to the entrance of the building, representing the almost indoor state or final stage in the transition phase. By doing this, we ensure having two sets of data from known locations and it allows us to assess the model's classification.

The first step in this experiment is to parse and extract the features from the

messages. After that, a visualization of the features over time, like the graph in Fig. 4.6, helps to explore their behavior in the two different scenarios.

The features portray the two states of the receiver using its measurements. The expectation is that variables suitable for outdoor to indoor transition detection will look very different in these two states' comparisons. Although there are many ways in which the features can show the difference, the main outlook appears to be the magnitude the values take. In the case of the carrier-to-noise density ratio, the difference in the value for the two states becomes visible clearly, for both the mean and the standard deviation of the visible satellites at that moment. In a lesser amount, it is noticeable with pseudorange residuals and the number of satellites. Variables, such as the DOPs, Elevation and Azymuth, are lost once the receiver gets too close to the building. This indicates that these variables are not suitable for the detection. Finally, the Doppler effect also shows difference in the measurements, but this variable is sensitive to the receiver's movement and this can cause false detection for scenarios where the device is in motion.

The next step is projecting the most promising feature using delay embedding to observe if the difference continues in the phase space, as seen in Fig. 4.7. This step also helps to prepare the data for training the model. The strongest difference between the locations is shown with C/N_0. When projected in the phase space, the observations are in separate places. This reaffirms the strong indication that this variable can help to detect the transition.

The model trains with a partial amount of the outdoor state samples. The reasons for this are: not fitting the model with too much data and simulating the initial run of the device when it starts functioning in the open-sky area. The model stops training after fitting all the selected outdoor data. The main limitation of this approach is that if the values of the variable change considerably after some time, even if the device is not moving, the captured distribution will not reflect the

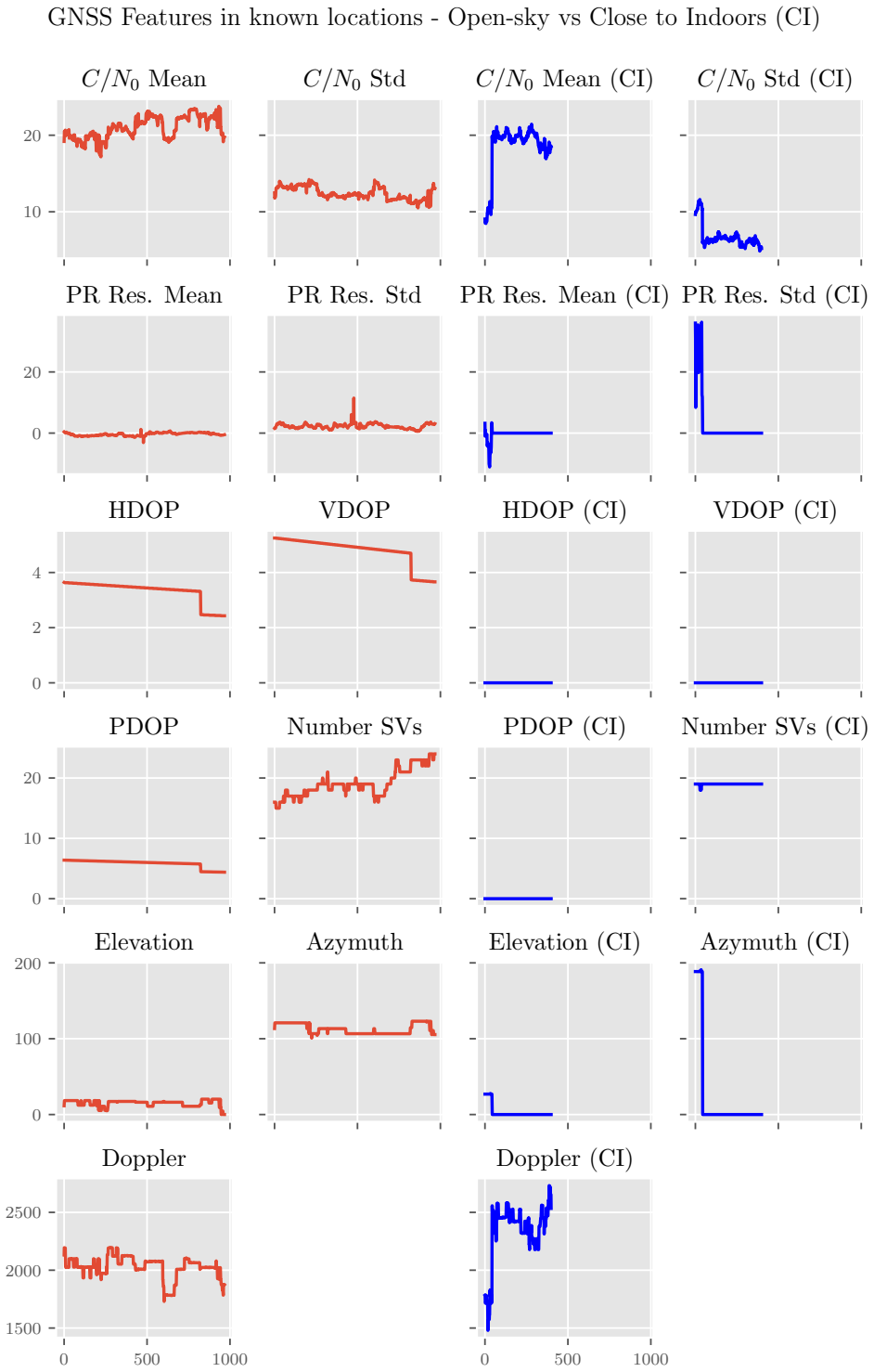


Figure 4.6: Measurements values over time for all the captured features from the GNSS Receiver. In red the observations from the open-sky location. In blue the ones from the close to indoor location.

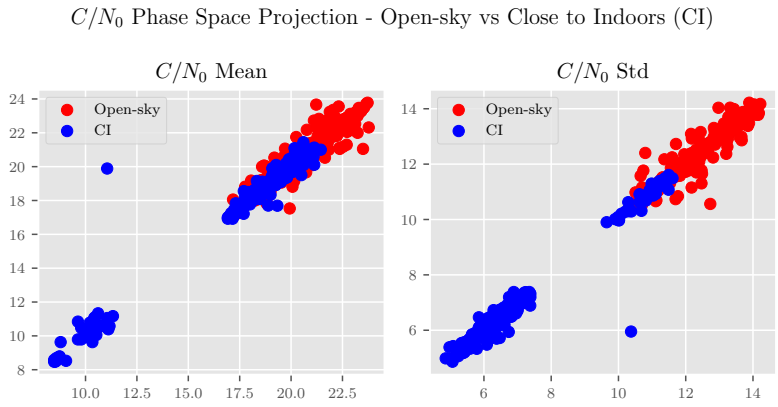


Figure 4.7: Projection of the C/N_0 measurements in the phase space using delay embedding. On the left each point represents the mean C/N_0 value of all the visible satellites at a given time. On the left each point represent the standard deviation C/N_0 value for the same satellites

receiver's state properly in that environment.

By limiting the amount of training data, the strategy tries to reduce the time that is necessary to have the system ready for usage and to avoid putting limitations on the general system performance. Because the model learns at the moment the device is in the field performing its overall task, the goal of not creating challenges or interruptions that can affect the primary objective of the device is key for the development of the algorithm.

The model hyperparameters for training are in shown in Table 4.1. The lower value for ν represents the low percentage of measurements that correspond to outliers or abnormal points in the training sample, and the high values of gamma were selected to increase the influence of a single sample, given the known certainty of the class to which the training samples do and don't belong. Finally, the model uses the RBF kernel function for its proven performance [156].

The evaluation of the resulting model using the hyperparameters in Table 4.1 is shown in Table 4.2. This results reflect the classification of the remaining outdoor data and the samples from the close indoor location. Correct classification is categorizing as negative the data from the close to indoor samples, and as positive the

Hyperparameter	C/N_0 **Mean**	C/N_0 **Std**
ν	0.05	0.05
Kernel function	RBF	RBF
Gamma	0.4	1.8
Stopping tolerance	0.001	0.001

Table 4.1: Hyperparameters for training a One-Class SVM model on known outdoor data

data from the outdoor samples. The accuracy is really low when using the C/N_0 Mean values, even if the model uses more than half of the training samples. When looking at the plot in Fig. 4.7, is understandable that this is the case, because many observations from the close to indoor location are overlapping the ones from the outdoor scenario. An important aspect in this case are the measurements that are far away from the rest. On the other hand, when using the C/N_0 Standard Deviation values, the model does better and with much less training samples. Again, this is very noticeable in the graph comparing the projections, as seen in Fig. 4.7.

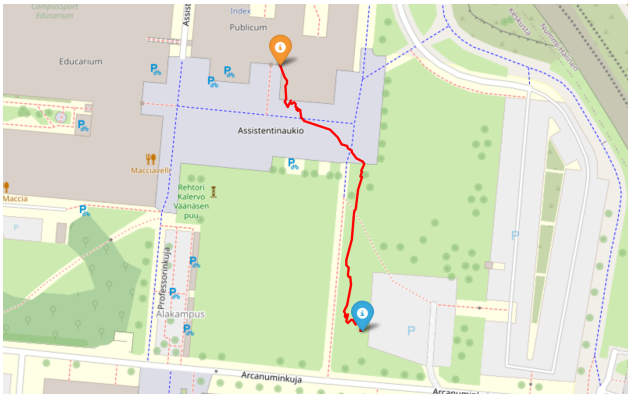
	% of outdoor data for training	**Accuracy**
C/N_0 **Mean**	51	0.52
C/N_0 **Std**	31	0.74

Table 4.2: Results for the One-Class SVM model trained on known outdoor data. The inference was made on data outside of the training samples. Accuracy means classifying correctly observations from an outdoor context as positive, and close to indoor samples as negative.

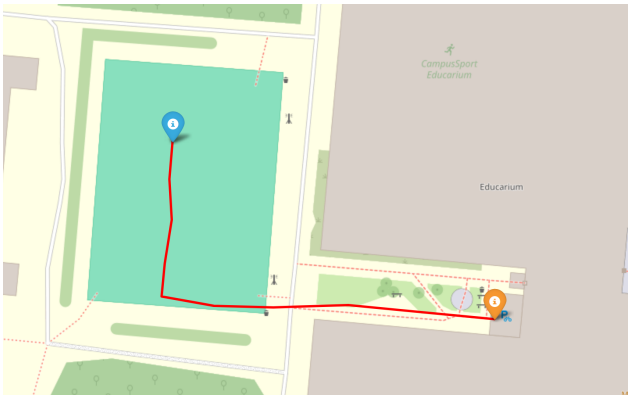
4.4 Experiment 2: Model performance for detecting the transition during a complete trajectory

This experiment uses the three locations shown in Fig. 4.2 and takes the receiver to follow three trajectories in each place. The paths for each trajectory are visible in Fig. 4.8. It builds upon the findings of the first experiment described Section 4.3.

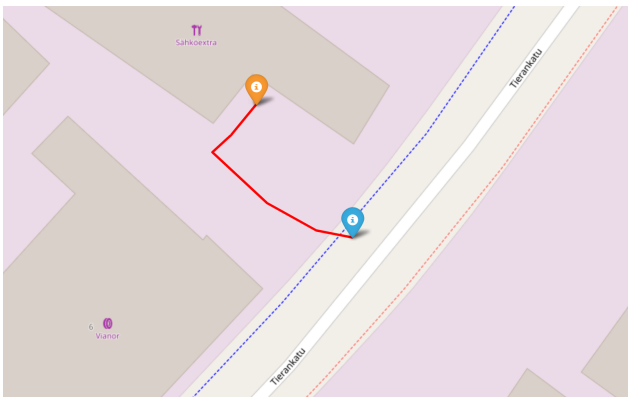
In addition to training a model, this experiment also provides a test of the system that raises a flag when it detects the receiver has gone under an outdoor to indoor transition.



(a) Location 1 Trajectory



(b) Location 2 Trajectory



(c) Location 3 Trajectory

Figure 4.8: Trajectories map for Experiment 2. The blue mark is the starting point, the orange one is the ending. Map image generated with Folium [157] and Map data copyrighted OpenStreetMap contributors and available from [158]

.

First, the receiver's data is collected while following each of the defined trajectories. Only the carrier-to-noise density feature is fully extracted and stored. As an initial step, both the mean and standard deviation value are examined to decide on which one to use to better identify the transition. After plotting their behavior during each trajectory displayed in Fig. 4.9, and carrying a visual inspection, the mean computation of C/N_0 provides a clearer picture of the signal degradation at the end of the trajectories, where is when the device is transitioning and getting closer to an indoor location. At the same time, the visual-inertial camera tracking saves the coordinates to follow the movement as an external verifying tool. The camera coordinates and the position from the receiver will help place the measurements in a more precise location in the trajectory.

Next, using the samples, a further analysis process projects the features, trains the model on an initial sample, and classifies the rest of the measurements to see if they are inside or outside the decision boundary. The embedding process uses a dimension of value 3, and time delay of 1. The selection corresponds to having a sliding window context of 1 second (because the receiver sampling rate is 3 Hz), and consecutive measurements being part of the same set. The plots in Fig. 4.10 show the C/N_0 values mean of the visible messages in its time series, and delay embedding form. The colors represent 10 seconds windows to easily see where the measurements fall in the projection. Given that the original embedding contains 3 dimensions, this visualization uses Principal Component Analysis (PCA) to reduce it to two and still obtain a good look of the behavior of the observations in the phase space.

The training strategy consists of using a limited time window at the start of the collection, in this case, 30 seconds. This selection is not fixed and can change according to each use case. By inspecting visually the 10-second windows and their behavior on the trajectories, the value of 30 seconds appeared enough to capture the

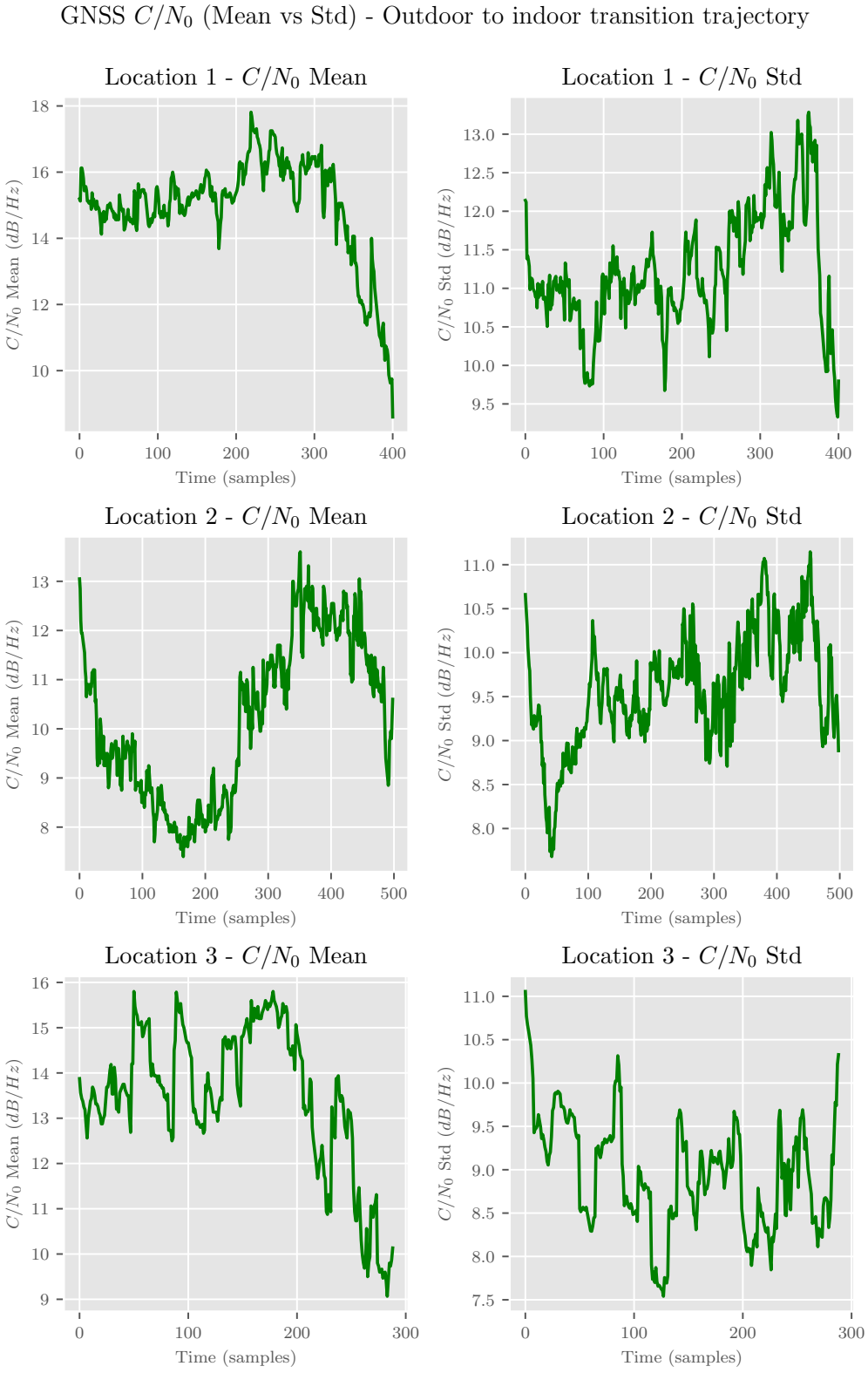


Figure 4.9: Behaviour of the C/N_0 Mean and Standard Deviation measurements in each of the locations.

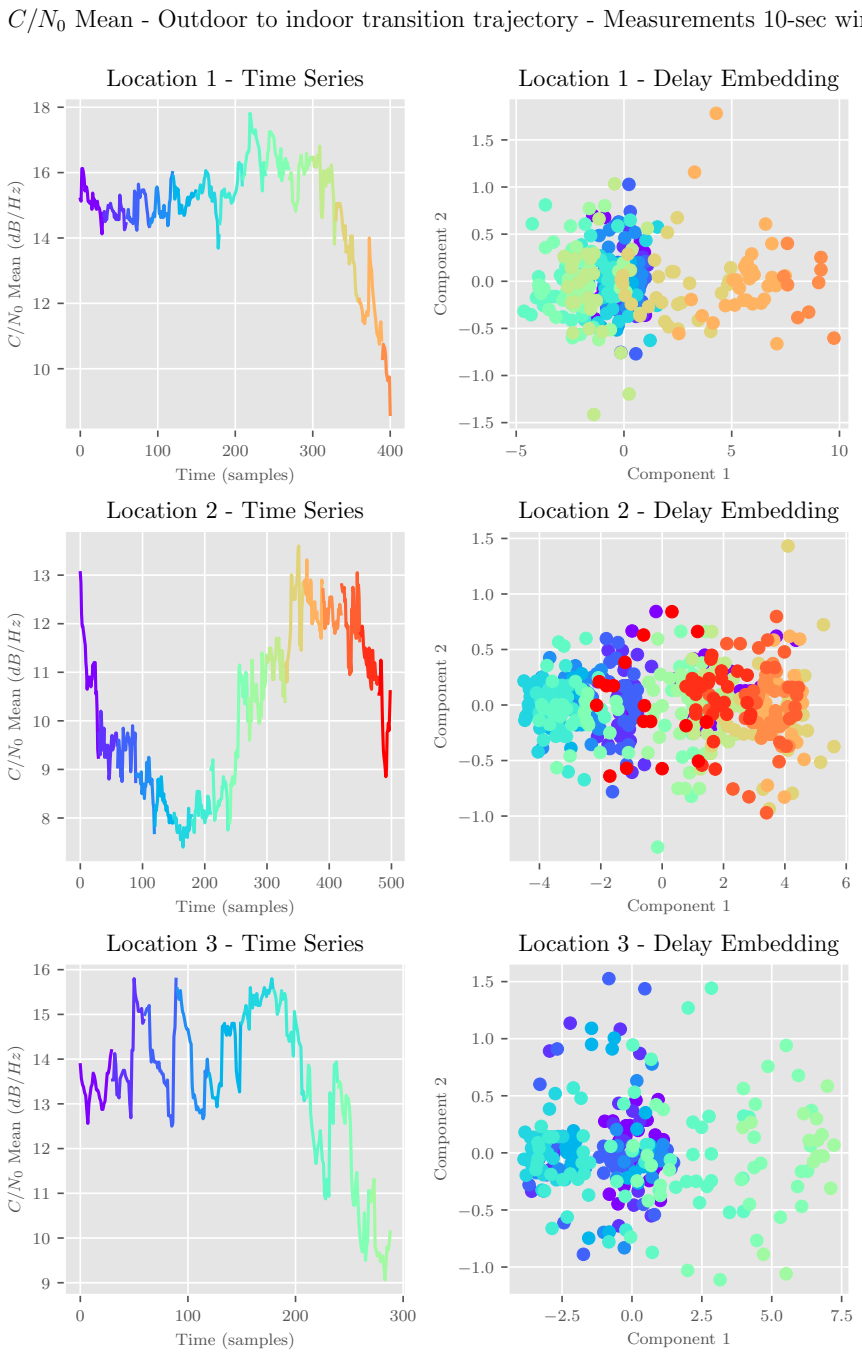


Figure 4.10: Behaviour of the C/N_0 Mean measurements in each of the locations. On the right the time series representation. On the left the projection in the phase space. Each color represent a 10 seconds-window measurements. Plots on the right are the PCA projection of the delay embeddings.

regular performance of the receiver in these scenarios and a good initial selection to continue the experiments. The existing samples from after the initial 30 seconds are only used for classification and not training. The training process is identical in the three trajectories.

The training hyperparameters selected for this model are shown in Table 4.3. This selection results from observing the performance of the parameters on a specific sample. The main focus is to create a boundary that covers most, if not all, of the training samples, but still allowing some degree of flexibility for small variations around them. The low value of ν indicates the model the almost non-existent probability of samples outside the defined positive class, and the relatively low gamma value helps to provide a more reaching boundary on the extremes.

Hyperparameter	**Value**
Kernel function	RBF
Gamma	0.05
ν	0.01
Training samples (time)	30 seconds

Table 4.3: Hyperparameters for training a One-Class SVM model on a 30 second sample during the transition trajectories in the selected locations

The training result is a model delimiting a decision boundary that determines samples coming from an open-sky (outdoor) location, and those that don't. The model captures the distribution from the training samples and classifies as positive measurements similar to those inside the boundary. Concretely, the model will capture the distribution of the 30 second window samples and data that fall inside of it in the phase space will be classified as positive, and those that fall outside as negative. Fig. 4.11 shows the model results over the trajectories data represented by the decision boundary plotted in the PCA projection of the delay embedded samples. The resulting model function encloses most of the training samples and defines what is considered to be a normal measurement, samples outside the boundary will be considered negative, or in this case, observations from a close to indoor location.

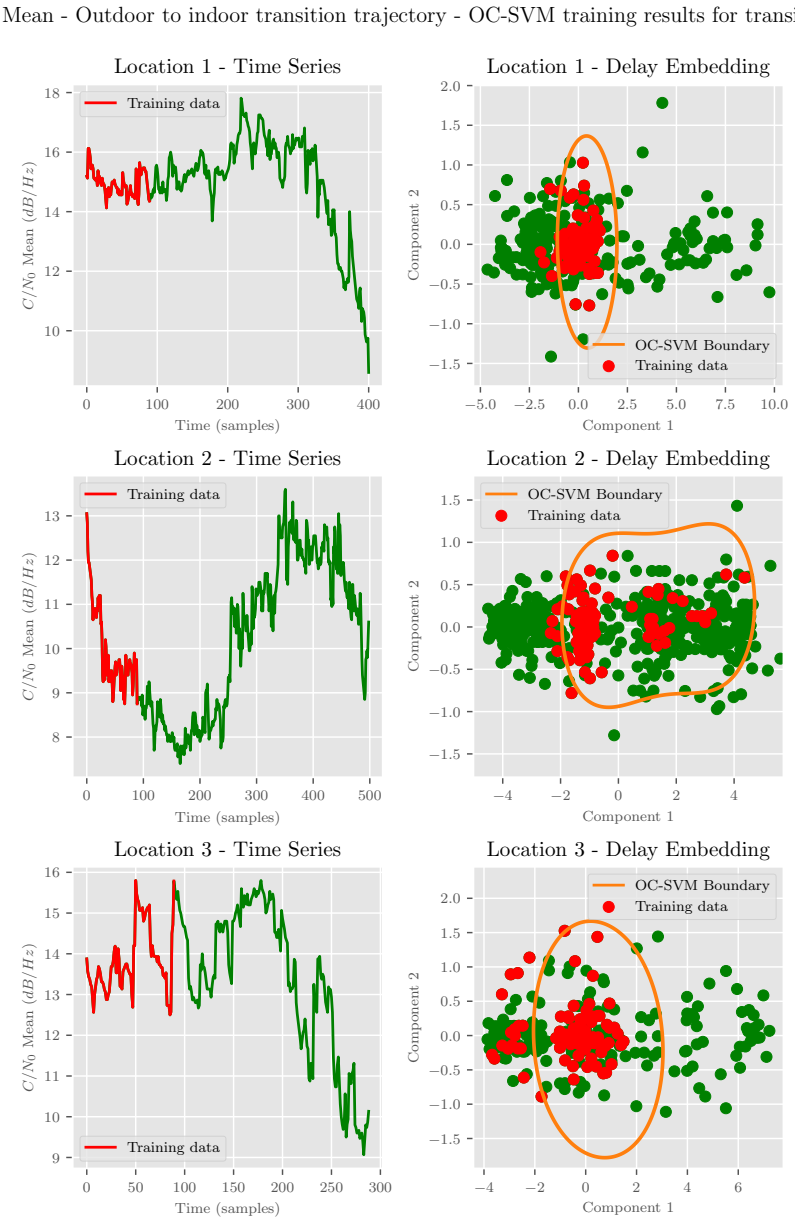


Figure 4.11: Training data and results using C/N_0 Mean measurements with a OC-SVM model. The observations in red are the training data, and in green the rest of the data. The decision boundary corresponds to the distribution learned by the model. Plots on the right are the PCA projection of the delay embeddings.

An algorithm (1) that raises a flag at the transition moment works as the real-case application system to evaluate the second experiment. The main blocks of the algorithm are model training and inference. In addition, it checks if the values of the carrier-to-noise density are higher or lower than the maximum value in the training samples because the values of this observable get lower when approaching indoor places. The algorithm counts the number of detections, and after a considerable number of them, it raises the flag for the transition at a particular time.

Fig. 4.12 presents the results for the trajectories in location 2 and 3. For location 2 the signal degrades very soon even before the device gets close to the building. One reason for this behavior are the surrounding obstacles around the device after going forward a few meters. This signal behavior causes the transition flag to raise very soon, and then to deactivate as the signal goes back to normal levels after continuing advancing another few meters. When the device gets close to the building the signal starts degrading again, and the flag raises to indicate the transition. It is noticeable that the signal does not recover after approaching the indoor context.

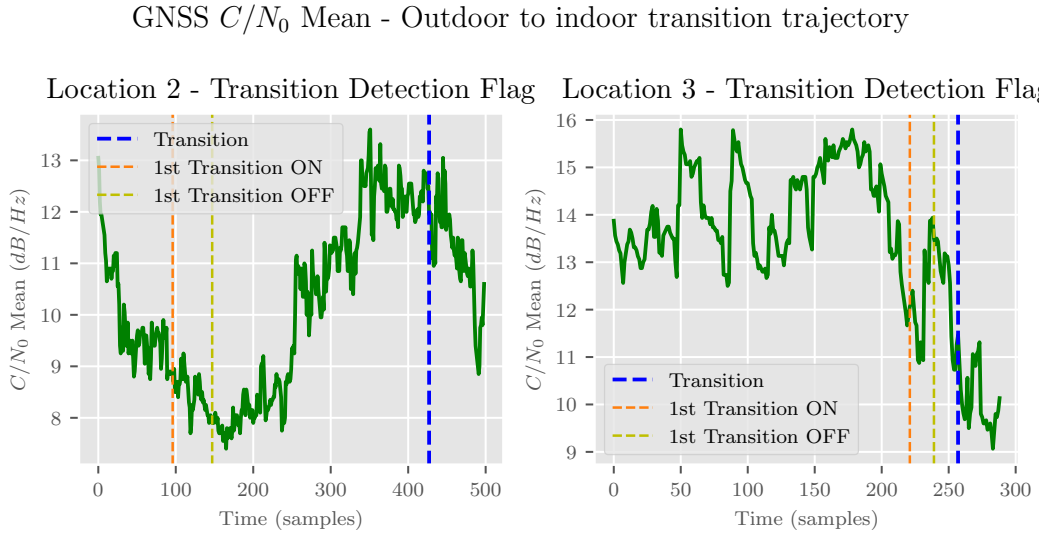


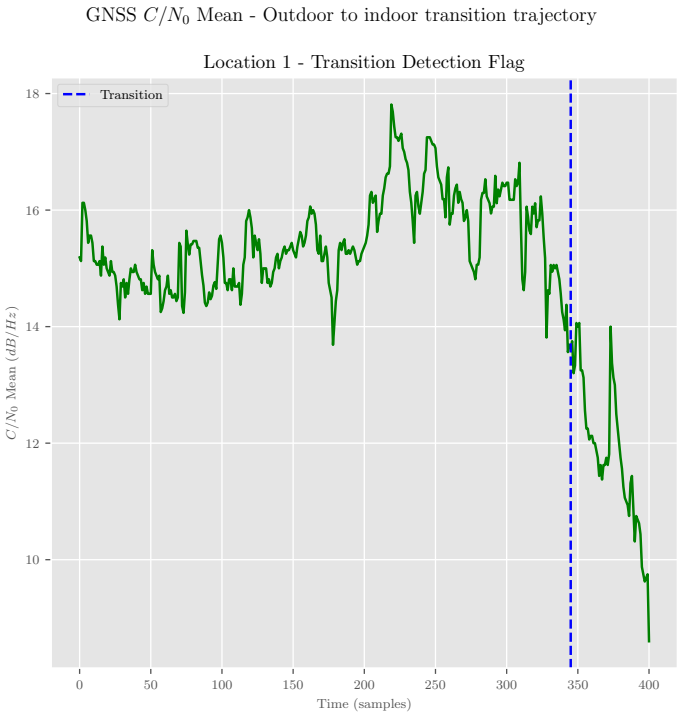
Figure 4.12: Results of the transition detection for Location 2 (left) and Location 3 (right). They show the last detection made by the algorithm using the blue dotted line. The first flag activation is shown by the orange dotted line, and the moment of the deactivation of the first flag is shown by the brown dotted line.

Listing 1 Transition Detection Algorithm for an incoming stream of measurements

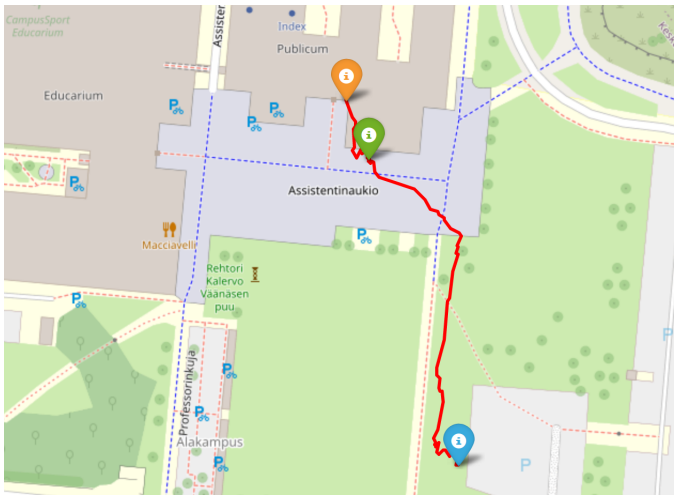
```
Require: NewMeasurement
Require: TrainingSamplesSize
Require: WindowSize
Require: DetectionSamplesSize
  isModelReady = False
  SamplesCount ← 0
  SamplesWindowSet ← Queue[WindowSize]
  TrainSet ← []
  OldState ← 0
  NegTransCount ← 0
  PosTransCount ← 0
  CurrentState ← 0
  while True do
    SamplesWindowSet.appendleft(NewMeasurement)
    if SamplesCount < TrainingSamplesSize then
      TrainSet.append(NewMeasurement)
    else
      if isModelReady = False then
        TrainSetEmbedded ← embedTrainSet(TrainSet)
        model ← trainModel(TrainSetEmbedded)
        isModelReady ← True
      else
        if NewMeasurement < max(TrainSet) then
          detection ← model.predict(SamplesWindowSet)
          if detection > 0 then
            NegTransCount ← 0
            PosTransCount ← PosTransCount + 1
            if NegTransCount = DetectionSamplesSize then
              CurrentState ← 1
            end if
          else
            NegTransCount ← NegTransCount + 1
            if NegTransCount = DetectionSamplesSize then
              CurrentState ← 2
              TransitionMark ← SamplesCount
            end if
          end if
        end if
      end if
    end if
    SamplesCount ← SamplesCount + 1
  end while
```

For location 3 the algorithm goes through a similar scenario. The signal gets a sudden drop, followed by an upward trend after it. This behavior activates the transition flag and deactivates it as soon as the signal goes to normal levels again. Finally, when the signal drops again the transition flag switch on, and because the signal keeps getting closer to the indoor environment it does not go up again.

The results indicate at which point the algorithm flags the transition happening. For location 1, Fig. 4.13 (a) shows the moment when the transition flag activates from the time series perspective. It is noticeable that the flag raises after the signals starts degrading. In Fig. 4.13 (b) the same transition flag appears but as a green mark. The trajectory is built from the GNSS receiver's position solutions. Here, the green mark is located very close to the building (which is the final destination), correctly indicating that the transition is happening and the device is close to enter an indoor environment.



(a) Location 1 - Transition Detection result in time series



(b) Location 1 - Transition Detection flag location in map. The green flag represents the detection flag.

Figure 4.13: Results of the transition detection for Location 1. Trajectory map image generated with Folium [157] and Map data copyrighted OpenStreetMap contributors and available from [158]. (a) shows the moment at which the transition flag is activated using the blue dotted line. (b) shows the trajectory map and places a green mark at the moment the transition flag is activated. The blue mark is the starting point and the orange the finishing one.

5 Conclusion

The motivation for this work was to provide a new solution for detecting environment transitions in the seamless positioning problem. The main focus was on adapting to different locations without extensive specific configurations and on exploiting the advantages of machine learning while reducing the cost of collecting and labeling data and the computational load for the devices. The research and development in this work tackle the previous challenges by providing evidence for using an unsupervised data learning method with GNSS measurements. As part of this development, the evaluation of GNSS observables to find a suitable variable that can help to detect the transition, such as the carrier-to-noise density ratio, and the usage of delay embedding on this data to prepare samples for training one-class support vector machines provide initial steps onto the exploration of this approach for more robust and complete systems. The proposed algorithm gives acceptable results for detecting the transition between outdoor to close to the indoor environment in Location 1's trajectory. The substantial factor for this result is that the carrier-to-noise density ratio successfully captures the degradation of the signal while getting closer to a building. Nonetheless, in Locations 2 and 3, the algorithm struggles to provide more consistency in the detection, and the degradation suffers multiple local minima and maximums that make it difficult to determine the transition as one specific moment. Surrounding infrastructure or tall objects appearing after the initial capture of outdoor data are the probable cause of the degradation of the signal before reaching

the building. A longer time capturing the outdoor data or adding supplementary margin to extend the boundary could be solutions to help in these situations. These results show the limitations, and the potential for using this method in cases where systems need to change or turn on or off their positioning systems according to the environment they are in, specially in scenarios where that detection is challenging due to the difficulty on choosing the right threshold, not having the necessary data to feed complex alternatives, or even those methods not being able to perform in a unknown environments for their models.

5.1 Future Works

The complications the developed algorithm found during the last trajectories indicate that a more robust approach is necessary to account for multiple transitions during the whole task. In addition, improvements in the data capture elements could also benefit the system as the selected measurement will portray better the actual state of the situation, and a deeper exploration of strategies to select the hyperparameters model could increase performance on specific scenarios. Although this thesis focuses on using only one variable to perform the detection, further work can explore integrating multiple variables and combinations of many models to vote and have a more complete decision. The use and positive performance of delay embeddings with GNSS measurements can also indicate the benefits of applying Topological Data Analysis to tackle the problem of detecting transitions and GNSS positioning challenges in general. Finally, a more detailed exploration of integrating the transition detecting system into a complete system will help exploit the benefits and focus on aiding the overall goal.

References

[1] W. Gao, S.-W. Kim, H. Bosse, *et al.*, "Measurement technologies for precision positioning", *CIRP annals*, vol. 64, no. 2, pp. 773–796, 2015.

[2] D. Egea-Roca, M. Arizabaleta-Diez, T. Pany, *et al.*, "Gnss user technology: State-of-the-art and future trends", *IEEE Access*, vol. 10, pp. 39 939–39 968, 2022.

[3] S. M. Asaad and H. S. Maghdid, "A comprehensive review of indoor/outdoor localization solutions in iot era: Research challenges and future perspectives", *Computer Networks*, vol. 212, p. 109 041, 2022.

[4] K. Krishnaswamy, S. Susca, R. McCroskey, *et al.*, "Sensor fusion for gnss denied navigation", in *2008 IEEE/ION Position, Location and Navigation Symposium*, IEEE, 2008, pp. 541–551.

[5] J. A. Navarro, M. E. Parés Calaf, *et al.*, "Towards seamless indoor-outdoor positioning: The iopes project approach", in *International archives of the photogrammetry, remote sensing and spatial information sciences: XXIV ISPRS Congress, Commission IV (volume XLIII-B4-2020)*, 2020, pp. 313–319.

[6] R. F. Brena, J. P. García-Vázquez, C. E. Galván-Tejada, D. Muñoz-Rodriguez, C. Vargas-Rosales, J. Fangmeyer, *et al.*, "Evolution of indoor positioning technologies: A survey", *Journal of Sensors*, vol. 2017, 2017.

[7] A. Siemuri, M. Elsanhoury, K. Selvan, P. Välisuo, H. Kuusniemi, and M. S. Elmusrati, "Seamless navigation for indoor-outdoor positioning using gnss-aided uwb/wifi/imu system", in *Proceedings of the 36th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2023)*, 2023, pp. 2616–2623.

[8] P. Zhou, Y. Zheng, Z. Li, M. Li, and G. Shen, "Iodetector: A generic service for indoor outdoor detection", in *Proceedings of the 10th acm conference on embedded network sensor systems*, 2012, pp. 113–126.

[9] U. Lipowezky, "Indoor-outdoor detector for mobile phone cameras using gentle boosting", in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*, IEEE, 2010, pp. 31–38.

[10] E. Angelats, A. Gorreja, P. F. Espín-López, M. E. Parés, E. S. Malinverni, and R. Pierdicca, "Enhanced seamless indoor–outdoor tracking using time series of gnss positioning errors", *ISPRS International Journal of Geo-Information*, vol. 13, no. 3, p. 72, 2024.

[11] P. B. Ober, "Integrity prediction and monitoring of navigation systems.", 2004.

[12] K. Chen and G. Tan, "Satprobe: Low-energy and fast indoor/outdoor detection based on raw gps processing", in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, IEEE, 2017, pp. 1–9.

[13] A. Siemuri, H. Kuusniemi, M. S. Elmusrati, P. Välisuo, and A. Shamsuzzoha, "Machine learning utilization in gnss—use cases, challenges and future applications", in *2021 International Conference on Localization and GNSS (ICL-GNSS)*, IEEE, 2021, pp. 1–6.

[14] D. Leite, A. Martins Jr, D. Rativa, J. F. De Oliveira, and A. M. Maciel, "An automated machine learning approach for real-time fault detection and diagnosis", *Sensors*, vol. 22, no. 16, p. 6138, 2022.

[15] H. Kuusniemi, *User-level reliability and quality monitoring in satellite-based personal navigation*. Tampere University of Technology, 2005.

[16] J. J. Spilker Jr, P. Axelrad, B. W. Parkinson, and P. Enge, *Global positioning system: theory and applications, volume I*. American Institute of Aeronautics and Astronautics, 1996.

[17] P. Teunissen, "Quality control in integrated navigation systems", *IEEE aerospace and electronic systems magazine*, vol. 5, no. 7, pp. 35–41, 1990.

[18] H. Pesonen, "A framework for bayesian receiver autonomous integrity monitoring in urban navigation", *Navigation*, vol. 58, no. 3, pp. 229–240, 2011.

[19] J. Blanch, T. Walker, P. Enge, *et al.*, "Baseline advanced raim user algorithm and possible improvements", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, no. 1, pp. 713–732, 2015.

[20] L.-T. Hsu, H. Tokura, N. Kubo, Y. Gu, and S. Kamijo, "Multiple faulty gnss measurement exclusion based on consistency check in urban canyons", *IEEE Sensors Journal*, vol. 17, no. 6, pp. 1909–1917, 2017.

[21] D. Kim and J. Cho, "Improvement of anomalous behavior detection of gnss signal based on tdnn for augmentation systems", *Sensors*, vol. 18, no. 11, p. 3800, 2018.

[22] K. Zhang and P. Papadimitratos, "Fast multiple fault detection and exclusion (fm-fde) algorithm for standalone gnss receivers", *IEEE Open Journal of the Communications Society*, vol. 2, pp. 217–234, 2021.

[23] R. Sun, J. Wang, Q. Cheng, Y. Mao, and W. Y. Ochieng, "A new imu-aided multiple gnss fault detection and exclusion algorithm for integrated navigation in urban environments", *GPS Solutions*, vol. 25, pp. 1–17, 2021.

[24] P. Zabalegui, G. De Miguel, J. Mendizabal, and I. Adin, "Innovation-based fault detection and exclusion applied to ultra-wideband augmented urban gnss navigation", *Remote Sensing*, vol. 15, no. 1, p. 99, 2022.

[25] C. Zhuang, H. Zhao, S. Hu, X. Meng, and W. Feng, "A novel gnss fault detection and exclusion method for cooperative positioning system", *IEEE Transactions on Vehicular Technology*, vol. 72, no. 4, pp. 4697–4712, 2022.

[26] G. Zhao, Z. Jiang, J. Wang, and S. Gao, "An improved lm-gn assisted ins/gnss dual-threshold fault detection algorithm", 2024.

[27] A. Siemuri, K. Selvan, H. Kuusniemi, P. Valisuo, and M. S. Elmusrati, "A systematic review of machine learning techniques for gnss use cases", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 58, no. 6, pp. 5043–5077, 2022.

[28] G. Ciaburro, "Machine fault detection methods based on machine learning algorithms: A review", *Mathematical Biosciences and Engineering*, vol. 19, no. 11, pp. 11 453–11 490, 2022.

[29] X. Hu, Y. He, and D. Xu, "A fault-tolerant integrated navigation method for intelligent vehicles", *Arabian Journal for Science and Engineering*, vol. 47, no. 11, pp. 15 015–15 025, 2022.

[30] L. Biddle and S. Fallah, "A novel fault detection, identification and prediction approach for autonomous vehicle controllers using svm", *Automotive Innovation*, vol. 4, no. 3, pp. 301–314, 2021.

[31] M. Bronz, E. Baskaya, D. Delahaye, and S. Puechmore, "Real-time fault detection on small fixed-wing uavs using machine learning", in *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*, IEEE, 2020, pp. 1–10.

[32] A. Beghi, L. Cecchinato, C. Corazzol, M. Rampazzo, F. Simmini, and G. A. Susto, "A one-class svm based tool for machine learning novelty detection in hvac chiller systems", *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 1953–1958, 2014.

[33] J. Ma and S. Perkins, "Time-series novelty detection using one-class support vector machines", in *Proceedings of the International Joint Conference on Neural Networks, 2003.*, IEEE, vol. 3, 2003, pp. 1741–1745.

[34] C. Guo, F. Li, Z. Tian, W. Guo, and S. Tan, "Intelligent active fault-tolerant system for multi-source integrated navigation system based on deep neural network", *Neural Computing and Applications*, vol. 32, pp. 16 857–16 874, 2020.

[35] V. Radu, P. Katsikouli, R. Sarkar, and M. K. Marina, "A semi-supervised learning approach for robust indoor-outdoor detection with smartphones", in *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, 2014, pp. 280–294.

[36] Y. Zhu, H. Luo, Q. Wang, *et al.*, "A fast indoor/outdoor transition detection algorithm based on machine learning", *Sensors*, vol. 19, no. 4, p. 786, 2019.

[37] Y. Xia, S. Pan, W. Gao, *et al.*, "Recurrent neural network based scenario recognition with Multi-constellation GNSS measurements on a smartphone", *Measurement*, vol. 153, 2020, ISSN: 0263-2241.

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

[39] B. Hofmann-Wellenhof, H. Lichtenegger, and E. Wasle, *GNSS–global navigation satellite systems: GPS, GLONASS, Galileo, and more*. Springer Science & Business Media, 2007.

[40] R. Mautz, "Combination of indoor and outdoor positioning", in *1st International Conference on Machine Control & Guidance*, 2008, pp. 1–9.

[41] D. Gebre-Egziabher and S. Gleason, *GNSS applications and methods*. Artech House, 2009.

[42] B. Bonet, I. Alcantarilla, D. Flament, C. Rodriguez, and N. Zarraoa, "The benefits of multi-constellation gnss: Reaching up even to single constellation gnss users", in *Proceedings of the 22nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2009)*, 2009, pp. 1268–1280.

[43] S. Verhagen, D. Odijk, P. J. Teunissen, and L. Huisman, "Performance improvement with low-cost multi-gnss receivers", in *2010 5th ESA workshop on satellite navigation technologies and European workshop on GNSS signals and signal processing (NAVITEC)*, IEEE, 2010, pp. 1–8.

[44] A. Angrisano *et al.*, "Gnss/ins integration methods", *Dottorato di ricerca (PhD) in Scienze Geodetiche e Topografiche Thesis, Universita'degli Studi di Napoli PARTHENOPE, Naple*, vol. 21, 2010.

[45] M. Paonni, M. Anghileri, S. Wallner, J.-A. Avila-Rodriguez, and B. Eissfeller, "Performance assessment of gnss signals in terms of time to first fix for cold, warm and hot start", in *Proceedings of the 2010 International Technical Meeting of The Institute of Navigation*, 2010, pp. 1051–1066.

[46] A. Grenier, E. S. Lohan, A. Ometov, and J. Nurmi, "A survey on low-power gnss", *IEEE Communications Surveys & Tutorials*, 2023.

[47] M. S. Grewal, A. P. Andrews, and C. G. Bartone, "Gnss receiver design and analysis", in *Global Navigation Satellite Systems, Inertial Navigation, and Integration*. 2020.

[48] X. Gong, S. Gu, Y. Lou, F. Zheng, M. Ge, and J. Liu, "An efficient solution of real-time data processing for multi-gnss network", *Journal of Geodesy*, vol. 92, pp. 797–809, 2018.

[49] S. Feng and W. Ochieng, "A difference test method for early detection of slowly growing errors in gnss positioning", *The Journal of Navigation*, vol. 60, no. 3, pp. 427–442, 2007.

[50] K. Gunning, T. Walter, and P. Enge, "Multi-gnss constellation anomaly detection and performance monitoring", in *Proceedings of the 30th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2017)*, 2017, pp. 1051–1062.

[51] M. P. Marco Rao Gianluca Falco, "Gnss solutions: Code tracking and pseudoranges. how can pseudorange measurements be generated from code tracking?", *Inside GNSS*, 2012.

[52] N. Viandier, D. Nahimana, J. Marais, and E. Duflos, "Gnss performance enhancement in urban environment based on pseudo-range error model", in *2008 IEEE/ION Position, Location and Navigation Symposium*, IEEE, 2008, pp. 377–382.

[53] P. Zabalegui, G. De Miguel, A. Pérez, J. Mendizabal, J. Goya, and I. Adin, "A review of the evolution of the integrity methods applied in gnss", *IEEE Access*, vol. 8, pp. 45 813–45 824, 2020.

[54] I. Sbeity, C. Villien, B. Denis, and E. V. Belmega, "Lstm-based gnss localization using satellite measurement features jointly with pseudorange residuals", *Sensors*, vol. 24, no. 3, p. 833, 2024.

[55] M. Z. H. Bhuiyan, S. Söderholm, S. Thombre, L. Ruotsalainen, M. Kirkko-Jaakkola, and H. Kuusniemi, "Performance evaluation of carrier-to-noise density ratio estimation techniques for beidou bl signal", in *2014 Ubiquitous Positioning Indoor Navigation and Location Based Service (UPINLBS)*, IEEE, 2014, pp. 19–25.

[56] K. Borre, I. Fernández-Hernández, J. A. López-Salcedo, and M. Z. H. Bhuiyan, *GNSS software receivers*. Cambridge University Press, 2022.

[57] M. S. Braasch and A. Van Dierendonck, "Gps receiver architectures and measurements", *Proceedings of the IEEE*, vol. 87, no. 1, pp. 48–64, 1999.

[58] E. Ghizzo, A. G. Pena, J. Lesouple, C. Milner, and C. Macabiau, "Assessing gnss carrier-to-noise-density ratio estimation in the presence of meaconer interference", in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2024, pp. 8971–8975.

[59] W. Huihui, Z. Xingqun, and Z. Yanhua, "Geometric dilution of precision for gps single-point positioning based on four satellites", *Journal of Systems Engineering and Electronics*, vol. 19, no. 5, pp. 1058–1063, 2008.

[60] R. B. Langley *et al.*, "Dilution of precision", *GPS world*, vol. 10, no. 5, pp. 52–59, 1999.

[61] R. R. Heselton, "Elevation effects on gps positional accuracy", Ph.D. dissertation, Virginia Tech, 1998.

[62] B. R. Rao, *GPS/GNSS Antennas*. Artech House, 2013.

[63] J. J. Z. J. Sanz Subirana and M. Hernández-Pajares, *Gnss basic observables*, `https://gssc.esa.int/navipedia/index.php/GNSS_Basic_Observables`, [Online].

[64] M. Tahsin, S. Sultana, T. Reza, and M. Hossam-E-Haider, "Analysis of dop and its preciseness in gnss position estimation", in *2015 International conference on electrical engineering and information communication technology (ICEEICT)*, IEEE, 2015, pp. 1–6.

[65] A. Angrisano, C. Gioia, S. Gaglione, and G. Del Core, "Gnss reliability testing in signal-degraded scenario", *International Journal of Navigation and Observation*, vol. 2013, no. 1, p. 870 365, 2013.

[66] M. M. Hoque and N. Jakowski, "Ionospheric propagation effects on gnss signals and new correction approaches", *Global navigation satellite systems: signal, theory and applications*, vol. 10, p. 30 090, 2012.

[67] P. M. Kintner, B. M. Ledvina, and E. De Paula, "Gps and ionospheric scintillations", *Space weather*, vol. 5, no. 9, 2007.

[68] P. Craven, R. Wong, N. Fedora, and P. Crampton, "Studying the effects of interference on gnss signals", in *Proceedings of the 2013 International Technical Meeting of the Institute of Navigation*, 2013, pp. 893–186.

[69] N. Spens, D.-K. Lee, F. Nedelkov, and D. Akos, "Detecting gnss jamming and spoofing on android devices", *NAVIGATION: Journal of the Institute of Navigation*, vol. 69, no. 3, 2022.

[70] C. Vegni and A. Neri, "Gnss interference: Effects and solutions", in *Proceedings of the Ion 2015 Pacific Pnt Meeting*, 2015, pp. 454–469.

[71] A. Hussain, F. Akhtar, Z. H. Khand, A. Rajput, and Z. Shaukat, "Complexity and limitations of gnss signal reception in highly obstructed enviroments", *Engineering, Technology & Applied Science Research*, vol. 11, no. 2, pp. 6864–6868, 2021.

[72] J. Zidan, E. I. Adegoke, E. Kampert, S. A. Birrell, C. R. Ford, and M. D. Higgins, "Gnss vulnerabilities and existing solutions: A review of the literature", *IEEE Access*, vol. 9, pp. 153 960–153 976, 2020.

[73] B. Ben-Moshe, E. Elkin, H. Levi, and A. Weissman, "Improving accuracy of gnss devices in urban canyons.", in *CCCG*, 2011, pp. 511–515.

[74] M. Smyrnaios, S. Schn, M. Liso, and S. Jin, "Multipath propagation, characterization and modeling in gnss", *Geodetic sciences-observations, modeling and applications*, pp. 99–125, 2013.

[75] M. Wang, P. Yu, and Y. Li, "Performance analysis of gnss/ins loosely coupled integration systems under gnss signal blocking environment", in *E3S Web of Conferences*, EDP Sciences, vol. 206, 2020, p. 02 013.

[76] K. S. Jacobsen and M. Dähnn, "Statistics of ionospheric disturbances and their correlation with gnss positioning errors at high latitudes", *Journal of Space Weather and Space Climate*, vol. 4, A27, 2014.

[77] R. B. Langley, "Gnss receiver system noise", *GPS World*, 1997.

[78] G. MacGougan, G. Lachapelle, R. Nayak, and A. Wang, "Overview of gnss signal degradation phenomena", in *Proceedings of the International Symposium Kinematic Systems and Geodesy, Geomatics and Navigation*, 2001.

[79] V. Dehghanian, J. Nielsen, and G. Lachapelle, "Gnss spoofing detection based on receiver c/no estimates", in *Proceedings of the 25th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2012)*, 2012, pp. 2878–2884.

[80] A. Jafarnia Jahromi, A. Broumandan, J. Nielsen, and G. Lachapelle, "Gps spoofer countermeasure effectiveness based on signal strength, noise power, and c/n0 measurements", *International Journal of Satellite Communications and Networking*, vol. 30, no. 4, pp. 181–191, 2012.

[81] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey", *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[82] V. Hodge and J. Austin, "A survey of outlier detection methodologies", *Artificial intelligence review*, vol. 22, pp. 85–126, 2004.

[83] J. M. Estévez-Tapiador, P. García-Teodoro, and J. E. Dı, "Measuring normality in http traffic for anomaly-based intrusion detection", *Computer Networks*, vol. 45, no. 2, pp. 175–193, 2004, ISSN: 1389-1286. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128604000064.

[84] R. Foorthuis, "On the nature and types of anomalies: A review of deviations in data", *International journal of data science and analytics*, vol. 12, no. 4, pp. 297–331, 2021.

[85] Z. Hameed, Y. Hong, Y. Cho, S. Ahn, and C. Song, "Condition monitoring and fault detection of wind turbines and related algorithms: A review", *Renewable and Sustainable energy reviews*, vol. 13, no. 1, pp. 1–39, 2009.

[86] N. Suri, N. M, and G. Athithan, *Outlier Detection: Techniques and Applications: A Data Mining Perspective* (Intelligent Systems Reference Library). Springer International Publishing, 2019, ISBN: 9783030051273. [Online]. Available: https://books.google.fi/books?id=wTSDDwAAQBAJ.

[87] I. Ben-Gal, "Outlier detection", in *Data Mining and Knowledge Discovery Handbook*, O. Maimon and L. Rokach, Eds. Boston, MA: Springer US, 2005, pp. 131–146.

[88] A. Smiti, "A critical overview of outlier detection methods", *Computer Science Review*, vol. 38, p. 100 306, 2020, ISSN: 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2020.100306. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013720304068.

[89] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A review of novelty detection", *Signal processing*, vol. 99, pp. 215–249, 2014.

[90] D. Miljković, "Fault detection methods: A literature survey", in *2011 Proceedings of the 34th international convention MIPRO*, IEEE, 2011, pp. 750–755.

[91] R. Isermann, "Supervision, fault-detection and fault-diagnosis methods—an introduction", *Control engineering practice*, vol. 5, no. 5, pp. 639–652, 1997.

[92] T. Brotherton and R. Mackey, "Anomaly detector fusion processing for advanced military aircraft", in *2001 IEEE Aerospace Conference Proceedings (Cat. No. 01TH8542)*, IEEE, vol. 6, 2001, pp. 3125–3137.

[93] D. Tolani, M. Yasar, S. Chin, and A. Ray, "Anomaly detection for health management of aircraft gas turbine engines", in *Proceedings of the 2005, American Control Conference, 2005.*, IEEE, 2005, pp. 459–464.

[94] J. De Kleer and B. C. Williams, "Diagnosing multiple faults", *Artificial intelligence*, vol. 32, no. 1, pp. 97–130, 1987.

[95] E. Eskin, "Anomaly detection over noisy data using learned probability distributions", 2000.

[96] A. J. Izenman, "Review papers: Recent developments in nonparametric density estimation", *Journal of the american statistical association*, vol. 86, no. 413, pp. 205–224, 1991.

[97] L. Tarassenko, D. A. Clifton, P. R. Bannister, S. King, and D. King, "Novelty detection", *Encyclopedia of structural health monitoring*, 2009.

[98] Z. Ouyang, X. Sun, and D. Yue, "Hierarchical time series feature extraction for power consumption anomaly detection", in *Advanced Computational Methods in Energy, Power, Electric Vehicles, and Their Integration: International Conference on Life System Modeling and Simulation, LSMS 2017*

and International Conference on Intelligent Computing for Sustainable Energy and Environment, ICSEE 2017, Nanjing, China, September 22-24, 2017, Proceedings, Part III, Springer, 2017, pp. 267–275.

[99] S. Gajjar and A. Palazoglu, "A data-driven multidimensional visualization technique for process fault detection and diagnosis", *Chemometrics and Intelligent Laboratory Systems*, vol. 154, pp. 122–136, 2016.

[100] Y. Xiao, H. Wang, W. Xu, and J. Zhou, "Robust one-class svm for fault detection", *Chemometrics and Intelligent Laboratory Systems*, vol. 151, pp. 15–25, 2016.

[101] S. Pullen and M. Joerger, "Gnss integrity and receiver autonomous integrity monitoring (raim)", *Position, navigation, and timing technologies in the 21st century: integrated satellite navigation, sensor systems, and civil applications*, vol. 1, pp. 591–617, 2020.

[102] S. Hewitson and J. Wang, "Gnss receiver autonomous integrity monitoring (raim) for multiple outliers", *The European Journal of Navigation*, vol. 4, no. 4, pp. 47–57, 2006.

[103] T. Krarup, "Gotterdammerung over least squares adjustment", in *Proc. 14th Congress of the International Society of Photogrammetry*, vol. 3, 1980, pp. 369–378.

[104] S. Zair, S. Le Hégarat-Mascle, and E. Seignez, "Outlier detection in gnss pseudo-range/doppler measurements for robust localization", *Sensors*, vol. 16, no. 4, p. 580, 2016.

[105] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006, vol. 2, pp. 1122–1128.

[106] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects", *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[107] T. Mitchell, *Machine Learning*. McGraw Hill, 1997.

[108] G. James, D. Witten, T. Hastie, R. Tibshirani, *et al.*, *An introduction to statistical learning*. Springer, 2013, vol. 112.

[109] A. Burkov, *Machine learning engineering*. True Positive Incorporated Montreal, QC, Canada, 2020, vol. 1.

[110] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, `http://www.deeplearningbook.org`.

[111] J. D. Kelleher, B. Mac Namee, and A. D'arcy, *Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies*. MIT press, 2020.

[112] T. Jiang, J. L. Gradus, and A. J. Rosellini, "Supervised machine learning: A brief primer", *Behavior therapy*, vol. 51, no. 5, pp. 675–687, 2020.

[113] T. Wuest, D. Weimer, C. Irgens, and K.-D. Thoben, "Machine learning in manufacturing: Advantages, challenges, and applications", *Production & Manufacturing Research*, vol. 4, no. 1, pp. 23–45, 2016.

[114] X. Zhu and A. B. Goldberg, *Introduction to semi-supervised learning*. Springer Nature, 2022.

[115] J. E. Van Engelen and H. H. Hoos, "A survey on semi-supervised learning", *Machine learning*, vol. 109, no. 2, pp. 373–440, 2020.

[116] R. K. Dwivedi, A. K. Rai, and R. Kumar, "A study on machine learning based anomaly detection approaches in wireless sensor network", in *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, IEEE, 2020, pp. 194–199.

[117] D. F. Oliveira, L. F. Vismari, J. R. de Almeida, *et al.*, "Evaluating unsupervised anomaly detection models to detect faults in heavy haul railway operations", in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, IEEE, 2019, pp. 1016–1022.

[118] D. Guthrie, L. Guthrie, B. Allison, and Y. Wilks, "Unsupervised anomaly detection.", in *IJCAI*, 2007, pp. 1624–1628.

[119] C. J. Burges, "A tutorial on support vector machines for pattern recognition", *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[120] H. J. Shin, D.-H. Eom, and S.-S. Kim, "One-class support vector machines—an application in machine fault detection and classification", *Computers & Industrial Engineering*, vol. 48, no. 2, pp. 395–408, 2005.

[121] B. Schölkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[122] L. H. Hamel, *Knowledge discovery with support vector machines*. John Wiley & Sons, 2011.

[123] S. S. Khan and M. G. Madden, "One-class classification: Taxonomy of study and review of techniques", *The Knowledge Engineering Review*, vol. 29, no. 3, pp. 345–374, 2014.

[124] S. Kotsiantis, D. Kanellopoulos, P. Pintelas, *et al.*, "Handling imbalanced datasets: A review", *GESTS international transactions on computer science and engineering*, vol. 30, no. 1, pp. 25–36, 2006.

[125] L. M. Manevitz and M. Yousef, "One-class svms for document classification", *Journal of machine Learning research*, vol. 2, no. Dec, pp. 139–154, 2001.

[126] S. Alam, S. K. Sonbhadra, S. Agarwal, and P. Nagabhushan, "One-class support vector classifiers: A survey", *Knowledge-Based Systems*, vol. 196, p. 105 754, 2020.

[127] K.-R. Müller, S. Mika, K. Tsuda, and K. Schölkopf, "An introduction to kernel-based learning algorithms", in *Handbook of neural network signal processing*, CRC Press, 2018, pp. 4–1.

[128] D. M. Tax and R. P. Duin, "Data domain description using support vectors.", in *ESANN*, vol. 99, 1999, pp. 251–256.

[129] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution", *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[130] A. Bounsiar and M. G. Madden, "One-class support vector machines revisited", in *2014 International Conference on Information Science & Applications (ICISA)*, IEEE, 2014, pp. 1–4.

[131] X. Zhang, C. Gu, and J. Lin, "Support vector machines for anomaly detection", in *2006 6th World Congress on Intelligent Control and Automation*, IEEE, vol. 1, 2006, pp. 2594–2598.

[132] K.-L. Li, H.-K. Huang, S.-F. Tian, and W. Xu, "Improving one-class svm for anomaly detection", in *Proceedings of the 2003 international conference on machine learning and cybernetics (IEEE Cat. No. 03EX693)*, IEEE, vol. 5, 2003, pp. 3077–3081.

[133] M. Amer, M. Goldstein, and S. Abdennadher, "Enhancing one-class support vector machines for unsupervised anomaly detection", in *Proceedings of the ACM SIGKDD workshop on outlier detection and description*, 2013, pp. 8–15.

[134] H.-S. Wu, "A survey of research on anomaly detection for time series", in *2016 13th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, IEEE, 2016, pp. 426–431.

[135] A. Sgueglia, A. Di Sorbo, C. A. Visaggio, and G. Canfora, "A systematic literature review of iot time series anomaly detection solutions", *Future Generation Computer Systems*, vol. 134, pp. 170–186, 2022.

[136] C. Cassisi, P. Montalto, M. Aliotta, A. Cannata, A. Pulvirenti, *et al.*, "Similarity measures and dimensionality reduction techniques for time series data mining", *Advances in data mining knowledge discovery and applications*, pp. 71–96, 2012.

[137] M. Braei and S. Wagner, "Anomaly detection in univariate time-series: A survey on the state-of-the-art", *arXiv preprint arXiv:2004.00433*, 2020.

[138] U. Yokkampon, S. Chumkamon, A. Mowshowitz, R. Fujisawa, and E. Hayashi, "Anomaly detection using support vector machines for time series data", *Journal of robotics networking and artificial life*, 2021.

[139] R. B. Amir, S. T. Gul, and A. Q. Khan, "A comparative analysis of classical and one class svm classifiers for machine fault detection using vibration signals", in *2016 International Conference on Emerging Technologies (ICET)*, IEEE, 2016, pp. 1–6.

[140] N. H. Packard, J. P. Crutchfield, J. D. Farmer, and R. S. Shaw, "Geometry from a time series", *Physical review letters*, vol. 45, no. 9, p. 712, 1980.

[141] F. Takens, "Detecting strange attractors in turbulence", in *Dynamical Systems and Turbulence, Warwick 1980: proceedings of a symposium held at the University of Warwick 1979/80*, Springer, 2006, pp. 366–381.

[142] E. Tan, S. Algar, D. Corrêa, M. Small, T. Stemler, and D. Walker, "Selecting embedding delays: An overview of embedding techniques and a new method using persistent homology", *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 33, no. 3, 2023.

[143] u-blox, *Zed-f9p module product page*, https://www.u-blox.com/en/product/zed-f9p-module, [Online].

[144] u-blox, *Zed-f9p interface description document*, https://cdn.sparkfun.com/assets/f/7/4/3/5/PM-15136.pdf, [Online].

[145] u-blox, *Zed-f9p integration manual document*, https://content.u-blox.com/sites/default/files/ZED-F9P_IntegrationManual_UBX-18010802.pdf, [Online].

[146] u-blox, *U-center software product page*, https://www.u-blox.com/en/product/u-center, [Online].

[147] S. Consulting, *Pyubx2: An original python 3 parser for the ubx, nmea and rtcm3 protocols*, https://github.com/semuconsulting/pyubx2, [Online].

[148] L. Buitinck, G. Louppe, M. Blondel, *et al.*, "API design for machine learning software: Experiences from the scikit-learn project", in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[149] Intel, *Robust visual-inertial tracking from a camera that knows where it's going*, https://www.intelrealsense.com/visual-inertial-tracking-case-study/, [Online].

[150] S. Consulting, *Pygpsclient: A free, open-source, multi-platform graphical gnss/gps testing, diagnostic, device configuration application*, https://github.com/semuconsulting/PyGPSClient, [Online].

[151] C. R. Harris, K. J. Millman, S. J. van der Walt, *et al.*, "Array programming with NumPy", *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. DOI: 10.1038/s41586-020-2649-2. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2.

[152] J. D. Hunter, "Matplotlib: A 2d graphics environment", *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007. DOI: 10.1109/MCSE.2007.55.

[153] P. Virtanen, R. Gommers, T. E. Oliphant, *et al.*, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python", *Nature Methods*, vol. 17, pp. 261–272, 2020. DOI: 10.1038/s41592-019-0686-2.

[154] G. Tauzin, U. Lupo, L. Tunstall, *et al.*, *Giotto-tda: A topological data analysis toolkit for machine learning and data exploration*, 2020. arXiv: 2004.02551 [cs.LG].

[155] Intel, *A python library for for intel realsense depth cameras*, https://github.com/IntelRealSense/librealsense/tree/master/wrappers/python, [Online].

[156] A. F. Rochim, K. Widyaningrum, and D. Eridani, "Performance comparison of support vector machine kernel functions in classifying covid-19 sentiment", in *2021 4th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, 2021, pp. 224–228. DOI: 10.1109/ISRITI54043.2021.9702845.

[157] python-visualization, *Folium*, version 0.11.0, Dec. 28, 2020. [Online]. Available: https://python-visualization.github.io/folium/.

[158] OpenStreetMap contributors, *Planet dump retrieved from https://planet.osm.org*, https://www.openstreetmap.org, 2017.