Passive Wireless Intelligence Tracking System (PasWITS)

An Innosuisse Project – The Swiss Innovation Agency for Innovation Demands
Grant Agreement No. 42193.1 IP-ICT

Technical Report 2022.07

The PasWITS Consortium

Universität Zürich UZH, Communication Systems Group CSG, Zürich, Switzerland
Livealytics AG, Zürich, Switzerland

For more information on this document or the PasWITS project, please contact:

Prof. Dr. Burkhard Stiller
Dr. Bruno Rodrigues
Universität Zürich, CSG@IfI
Binzmühlestrasse 14
CH—8050 Zürich
Switzerland

Phone: +41 44 635 46 81
Fax: +41 44 635 6809
E-mail: [stiller—rodrigues]@ifi.uzh.ch

Document Control

Title:	Final Report
Type:	Public
Editor(s):	Burkhard Stiller, Bruno Rodrigues, Katharina O.E. Müller, Simon Tuck
E-mail:	[stiller—rodrigues—mueller]@ifi.uzh.ch, simon.tuck@livealytics.com
Author(s):	(in alphabetical order) Burkhard Stiller, Bruno Rodrigues, Katharina O. E. Müller Simon Tuck
DOC ID:	PasWITS_Final_Report_2022.pdf

Legal Notices

Contents

1 Executive Summary

Measuring public interest in a particular product or service is highly important for the strategic planning of businesses. Today, secure and non-invasive methods of visualizing an audience's mobility exist, which are key in an increasingly digitized society by passively tracking wireless signals emitted by portable devices. The analysis of these signals enables the extraction of Key Performance Indicators (KPI) for the efficient planning of marketing strategies for business events or campaigns, which in turn improves the offering of a product or service to a particular type of audience. The provision of a solution capable of benchmarking live performance Key Performance Indicators (KPI) of exhibitions or events based on data passively captured from signals emitted from mobile devices defines the essential basis for the "Passive Wireless Intelligence Tracking System" project (PasWITS). Thus, it is possible to obtain insights into how long visitors are engaged in an exhibition and how many visitors and observers are on site. Additional metrics, such as peak times, popular routes, and the duration of certain visits, will become measurable.

PasWITS explores two fundamental points from a business point of view: (1) the passive capture of device data and its identification, and (2) data correlation and visualization, which involves the correlation of devices and users and a prediction of scenarios (*i.e.,* prediction of events and exhibition performance). Also, by adding a dedicated visualization solution to the scenario characterization, it will be possible to predict how a particular event or exhibition might behave based on historical data.

Firstly, PasWITS contributes to the design of a suitable passive collection architecture with different assumptions. Secondly, PasWITS provides a prototypical implementation by correlating these data sources to ensure the precise identification of devices and their trajectories. Thirdly, the visualization of KPIs is based on the processing of collected data. In this regard, the KPI calculation and visualization, while not presenting an in-depth research complexity, have a fundamental business impact, which enables clients to view the performance of their exhibition or event in real-time. The fourth key contribution of PasWITS is the creation of event and display profiles in order to create respective databases for predicting success or failure based on those KPIs observed.

Despite pandemic-related challenges preventing the planned for large-scale testing and evaluations, several positive observations can be made from both a business and an academic point of view. From a business point of view, the Cloud Counter solution has become a product in the livealytics' service portfolio. From an academic point of view, several peer-reviewed publications have been published, in which the UZH team extended with students investigated the theoretical and practical opportunity an operation of sensors carries.

The overall outcome of PasWITS indicates that the construction of a solution, while tackling the multi-object tracking correlation problem, is highly valuable. The passive approach taken and as proposed by PasWITS allows for a higher degree of accuracy in the understanding of how a certain crowd moves and behaves, without influencing the system under evaluation, thus, generating an very useful input on the planning of their efficiency and positioning of public services. Also, improvements disclosed in this report allow livealytics to expand into other customer segments, such as smart cities, tourism, and arts/entertainment.

2 PasWITS

The PasWITS project involved livealytics, the SME focused on ”tracking & measurement solutions provide insights to improve live customer experiences, optimize operational costs, and increase sales.” and the Communication Systems Group CSG, Department of Informatics IfI at the University of Zürich UZH with the expertise on ”excellent research in communications, addressing communication mechanisms for charging, accounting, mobility, security, while considering Telecommunication economics, network management, and highly-decentralized systems as major pillars of today’s communications in the Internet.” Thus, based on an introduction into the combination of selected areas of expertise, the PasWITS project goals are defined, the methodology deployed is summarized, and the structure of this report is provided.

2.1 Introduction

Several methods for visualizing an audience’s mobility by (actively or passively) tracking signals emitted by portable devices carried by humans were proposed. However, they fall short on real-time tracking due to a number of reasons, such as the oscillation and interference of wireless signals or the lack of line-of-sight in image-based methods. Furthermore, real-time satellite-based positioning technologies, *e.g.,* the Global Positioning System (GPS), only work outdoors (given the need for line-of-sight connection).

In this regard, there exist different indoor tracking technologies, ranging from capturing wireless signals emitted by smartphones to the visual camera-based localization of devices. However, in comparison to outdoor tracking, no specific technology has been established as a *de facto* standard due to inefficiencies when used in isolation. For example, indoor tracking based on Wi-Fi signals (*i.e.,* IEEE 802.11 family of protocols) achieves a larger coverage area than tracking based on indoor cameras but suffers from a lack of accuracy in tracking the real-time position of these devices. The use of cameras ensures accurate tracking under line-of-sight conditions but has a limited range and restrictions regarding privacy.

A single type of tracker may not give sufficiently detailed information about a device’s position, often due to unreliability to estimate a position precisely. Thus, the main challenge is the dynamic variations in error types and anomalies caused by the environment, such as multi-path fading and signal attenuation. Therefore, combining measurements from different tracking approaches, such as cameras, Bluetooth sensors, RFID-Tags (Radio Frequency Identifiers), and other sources (*e.g.,* LiDAR), can be used to increase the quality of estimating device positions. Merging data from various nodes into “a sink” allows for the calculation of correlations of these different data sources to estimate the devices’ position more precisely.

2.2 Project Goals

PasWITS had two fundamental goals from a business point of view: (1) the capture of passively device data and its identification, and (2) data correlation and visualization, which involves the correlation of devices and users and prediction of scenarios (*i.e.,* prediction of events and exhibition performance).

2.2.1 Passive Capture of Wireless Signals and Device Identification

The major technical challenges are the unique device identification and the correlation of device(s) to its users considering that users might carry more than one device emitting wireless signals, and these devices can emit passive wireless signals through different sources. Thus, it is possible to correlate the spatial and temporal dimensions through the RSSI signal strength captured from different sources (*e.g.,* Wi-Fi and Bluetooth) measured at a given point in time. Thus, is possible to determine with a higher likelihood the uniqueness of tracked devices. The major benefit of solving these challenges is to increase the precision of the calculated performance metrics based on accurate measurements. Therefore, a solution that allows (a) broadening the number of tracked sources, and (b) enabling correlation, at the analysis stage, the signal sources to uniquely identify the signals emitted by a single device, will increase the accuracy of device detection.

Increasing the precision/accuracy of device identification and device-user correlation is a key technical point to enable a competitive business advantage. Therefore, such identification accuracy can be quantified, for example, based on a comparison of experiments in controlled environments using a passive capture approach based only on the 802.11 Wi-Fi protocol family, with the proposed approach combining Wi-Fi and Bluetooth sources and correlating its produced data to avoid MAC-randomization. Hence, it will be possible to determine a competitive advantage where the novel approach allows an increased accuracy percentage to identify unique devices.

2.2.2 Data Correlation, Visualization, and Prediction of Performance

The challenges involve correlating wireless signals to unique devices and mapping these unique devices to their users. Also, the input requires the event or exhibition settings to determine the specific configurations for each customer, as well as external factors that may influence the campaign (*e.g.,* holidays, weather conditions, and others). Another challenge related to the data analysis involves predicting the performance of events or exhibitions based on previous data (*i.e.,* estimating the success or failure probabilities by profiling events and exhibitions). Lastly, visualizing KPI metrics on a dashboard enables real-time performance analysis of the event or exposure, allowing one to make adjustments on the run when possible.

A solution for these challenges offers an opportunity to make marketing investments more efficient. By extracting relevant information from the collected data, it is possible to provide customers with a low-risk investment to promote an exhibition or event and obtain live feedback on its performance. Therefore, solving such challenges is a great opportunity to create a competitive advantage by combining a hardware and software solution capable of extracting relevant information for exhibitors.

2.3 Methodology

The progress of the project followed a balanced approach between development and practice, in which developed prototypes were, at first, individually evaluated. In this sense, the project used a **referenced research** method to investigate and design tracking methods that can be used passively, as well as **applied research** to implement and test tracking methods

individually as well as combined. As such, the project had two clearly stages structured in a way that corresponded to its objectives:

1. **Expand input of passive tracking sources**: the first stage of the project involved widening existing wireless tracking approaches, such as IEEE 802.11 (WiFi) and Bluetooth Low Energy (BLE), to operate passively. Both referenced and applied research were used to tackle specific challenges of these protocols, such as MAC address randomization in the case of WiFi and high imprecision of signals in the case of BLE. At this stage of the project, input sources were expanded beyond WiFi and BLE to include RFID, LiDAR, and Cameras. In general, each individual input source followed the general steps below:

    - **Design and data schema**: based advantages and disadvantages of each source, the prototype was designed to collect data in a passive and GDPR-compliant way (*i.e.,* anonymizing sources). The data schema refers to minimal necessary timestamped spatial coordinates and additional information such as angle of capture, height, rotation, the field of view, and other information that each sensor may provide.
    - **Applied development**: The implementation and application process in use cases was conducted in close partnership with LiveAlytics, which provided hardware and allowed several prototypes to be evaluated with the product on real consumers. This applied research stage proved advantageous for refining the prototypes to the practical needs and challenges faced in an operational stage, such as signal interference or, in some cases, lack of power and connectivity.
    - **Data analysis**: data produced over a period (day or weeks) was analyzed individually to identify each source's advantages and disadvantages in contrast to true cases.

2. **Correlate, predict, visualize**: the second stage corresponds directly to the second objective. The overlap with the first stage, in which different indoor tracking sources were explored, concerns the definition of a data schema with minimal information. The major point was to ensure that different sensors produced comparable data, which involved pre-processing stages in producing spatial information of tracked objects based on a synchronized global clock. Similarly, this stage involved referenced research on the state-of-the-art (near) real-time distributed systems and data fusion approaches. Three major steps were followed at this stage, overlapping with the design of each data source in the first stage:

    - **Data structure and API**: an essential step to realizing the correlation and prediction of scenarios. This step was mostly based on existing definitions from referenced research and applied in practice. Thus, each tracking source should periodically produce timestamped coordinates.
    - **Design of correlation and prediction engine**: involved referenced and applied research to solve the problem of correlating large amounts of data generated by different sensors of different types.

- **Visualization**: this stage tried to answer the question of how to provide a simple and direct understanding that allows a non-technical user to understand the metrics provided by the system. In this sense, the production of the dashboard was also based on referenced and applied research methods.

2.4 Report Outline

This final technical report of PasWITS is structured as follows.

- Section 3 outlines the fundamental concepts concerning each investigated input source and the theoretical basis used in their solution.
- Section 4 presents in a general way the project's modular development methodology, which was organized in data input (involving several sensors), processing backend, which receives data from several sensors and performs data correlation and prediction, and frontend which presents information in an efficient way for users.
- Section 5 offers the results of the various experiments performed during the project, including local evaluations, simulations, and in-field experiments performed along with livealytics.
- Section 6 discusses those key findings, addresses the crucial points and impacts, and outlines challenges faced in the execution of the project.
- Lastly, Section 7 summarizes the work performed and adds considerations along with possible aspects for improvement.

3 Fundamentals

While industrial relevance is shown at first, the fields of application are described secondly to complement in a third part this perspective with their technical and formal foundations this project builds upon and eventually exploits.

3.1 Industry-relevance of PasWITS

Although there are local and international companies in the same scope of internal tracking, it is observed that the main focus of the project (correlation of different data sources) is not explored by these companies. Companies often rely on the sole capture of Wi-Fi signals, it is noted that their tracking method is based on an active mode - requiring that the analyzed device be connected to an access point. As proposed by PasWITS, a passive approach allows understanding of how a certain crowd moves and behaves, without influencing the system under evaluation, generating an extremely useful input on planning their efficiency and positioning of public services. First, PasWITS contributes to implementation by correlating these sources to ensure precise identification of devices and their trajectories. Secondly, the visualization of KPIs is based on processing collected data. In this regard, KPI calculation and visualization, while not presenting an in-depth research complexity, have a fundamental business impact, which enables clients to view the performance of their exhibition or event in real-time. The third key contribution of PasWITS is creating event and display profiles to create a database for predicting success or failure based on the observed KPIs.

3.2 Fields of Application

The concept of the PasWITS project, jointly idealized by UZH and LiveAlytics, targets the indoor localization scenario providing near real-time tracking. In this scenario, multiple use cases are possible in public urban spaces or private events/trade fairs in order to analyze the flow of people and obtain insight into their interest in a particular product or service. That is the target of the concepts designed and evaluated in PasWITS considering the current needs (*i.e.,* customers, use cases) of LiveAlytics. For example:

- **Live events and promotions**: also known as MICE (Meetings, incentives, conferences, exhibitions), in which the goal is to provide, for example, a view on the engagement of visitors, obtain insights of which products/services are more attractive.
- **Retail analytics**: offers the opportunity to obtain important data concerning customer behavior and the performance of different stores and activities. For example, it is possible to provide heatmaps and flowcharts and test the impact of certain promotions or store layouts on customer behavior.
- **Health in public spaces**: a use case that was adjusted during the execution of the project. Tracking customers' behavior indoors allows for increasing health and safety measures such as safe distances, placement of customer-oriented services, and checks based on their vaccination certificates.

In addition, PasWITS was designed to follow an (1) modular and (2) data-driven approach to (1) allow flexibility in the types of sensors used, and (2) combine the data from different sensors allowing for more accurate tracking. These design steps were directly derived from the project goals, namely (1) the capture of passive device data and its identification, and (2) data correlation and visualization. As such, can be expanded to further use cases where the fusion of data from different sensors is beneficial (*i.e.,* general tracking) making PasWITS not limited to the use cases above. Nonetheless, the approach requires harmonization of data (pre-processing) and the use of a standard data format to correlate tracking data from sensors of different natures.

3.3 Technical Foundations

These foundations encompass communication standards used, camera devices deployed and evaluated, and formal methods exploited to solve the major questions posed.

3.3.1 WiFi IEEE 802.11

The IEEE 802.11 standard specifies a set of protocols for implementing WLAN as presented in Figure 1. The IEEE 802.11 specification focus on the two lowest layers of the OSI model since they incorporate both physical and data link components. The link layer provides a set of rules determining how to access the medium and send data, but the details of transmission and reception are left to the PHY layer (layer 1) [36].

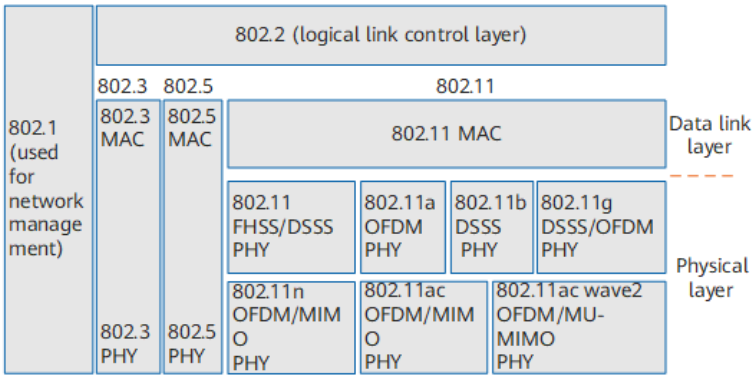

Figure 1: The IEEE 802.11 Family of Protocols

Basic components of 802.11 are stations, access points, wireless medium, and distribution system. Stations are computing devices with wireless network interfaces, which are not necessarily mobile devices. Access Points work as bridges between the wireless medium and the rest of the world. They convert wireless frames to frames that can be transmitted over fixed network connections [36]. Lastly, the wireless medium is the medium the data is transmitted.

PasWITS relies on 802.11 probe-request frames sent by mobile devices to request information on available access points nearby. A probe-request frame (*cf.* Figure 2) consists of various fixed-length header fields and a body of variable length consisting of a varying amount that reveals certain properties about the sender.

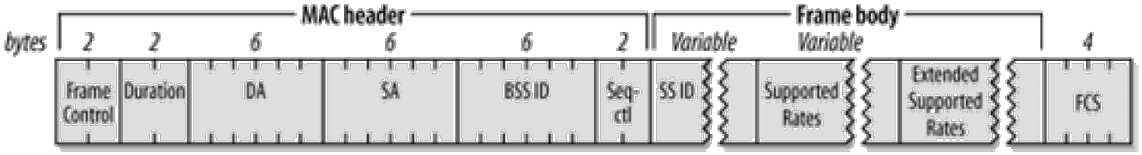

Figure 2: Probe Request Structure [36]

Probe Requests are broadcast in bursts at a high frequency by many devices containing an 802.11 capable network card. Their purpose is to discover access points to associate with [68]. The number of bursts, the number of probe requests contained in each burst, and the exact timing between each packet, depending on several factors such as screen state, charging state, airplane mode, Wi-Fi setting screen open, Bluetooth activation, and proximity of a known network [68].

3.3.2 Bluetooth Low Energy (BLE)

Bluetooth (BT) is defined as a short-range communications system intended to replace the cable(s) connecting portable and/or fixed electronic devices [13]. Operating in the unlicensed 2.4 GHz Industrial, Scientific, and Medical (ISM) frequency band, BT devices typically transmit up to a distance of 10 meters to serve this purpose.

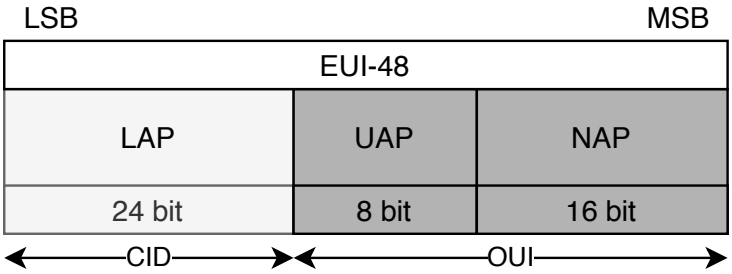

Figure 3: The Composition of BT Addresses [13]

A BT packet is composed of a 24-bit Company ID (CID) and a 24-bit Organizationally Unique ID (OUI) (*cf.* Figure 3). Every BT device has a unique BT address, which is constructed as a 48-bit Extended Unique Identifier (EUI-48) according to the IEEE Standard for Local and Metropolitan Area Networks [43]. The CID is vendor-assigned, whereas the OUI has to be obtained from the IEEE Registrations Authority and is assigned to individual organizations, manufacturers, or vendors of BT technology. For BT networking, parts of the BT address are differentiated by the Lower Address Part (LAP), corresponding to the CID, and the 16-bit non-significant and 8-bit upper address parts, forming the OUI.

The Project Ubertooth is the passive sensing device used for tracking BT packets. Ubertooth is a fully open-source hardware and software package for wireless development [37], suitable for experimentation with both BT Low Energy (BTLE) and classic BT. Ubertooth allows access to lower layers of Bluetooth protocols, which are normally hidden in off-the-shelf Bluetooth modules, being available at low cost.

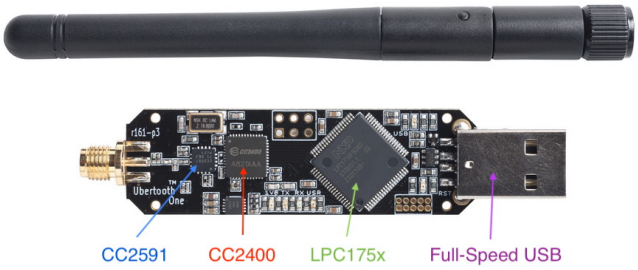

Figure 4: The Ubertooth One Sniffer [40]

BT is a spread spectrum technology, meaning that it moves over a wide range of frequencies during the transmission of data. Most BT monitoring hardware, therefore, implements an array of transceivers to observe all channels used by the BT protocol simultaneously. Ubertooth One used in this paper (*cf.* Figure 4), however, only uses a single transceiver, opting instead to try and hop along with the hopping pattern of ongoing BT connections to eavesdrop data being transmitted [37]. Consequently, only BT classic is fully supported by Ubertooth One.

3.3.3 Ultra-wideband (UWB)

The Ultra-wideband technology was developed following the allocation of the 3.1-10.6 GHz bands by the Federal Communications Commission (FCC) in 2002. The IEEE 802.15.3 UWB standard was initially introduced with the purpose of enabling high throughput, allowing for high data transfer over short ranges. In comparison, the Impulse Radio UWB (IR-UWB) standard was defined in the IEEE 802.15.4 specifications, to allow for highly accurate ranging and positioning. While UWBs high throughput capability has not been adopted by the industry, IR-UWB has seen multiple renewals and amendments, such as IEEE 802.15.4a, which until recently was the most used UWB standard [73].In 2019, IR-UWB localization received renewed interest due to the release of the IEEE 802.15.4z amendment, which aimed to further increase IR-UWB ranging performance and Physical Layer security[21]. Leading to the founding of the Fine Ranging (FiRa) Consortium, backed by industry leaders such as Apple, NXP, Samsung, and Google[32].

In April 2021 Apple released the AirTag, which utilizes a combination of the previously mentioned Bluetooth Low Energy (BLE), Global Positioning System (GPS) and IR-UWB to allow any object, attached to the tag, to be localized across multiple orders of magnitude: across town (GPS), across a house (BLE), and across a table (UWB), as depicted in Figure 5.

In comparison to BLE, UWB is a wideband technology that utilizes a broader spectrum of frequencies, but at lower power and for multiple short bursts, called pulses [78]. The output power of a UWB pulse is regulated to limit interference with other protocols, essentially: ”the higher the rate, the lower the transmission power per pulse.[78]” As such, the protocols frame transmissions are split across pulses and usually integrate with the noise on the spectrum (*cf.* Figure 6).

UWB is therefore promising in terms of precise indoor localization. However, optimal reliability and precision, in line-of-sigh scenarios, are only possible from a distance of 20 m

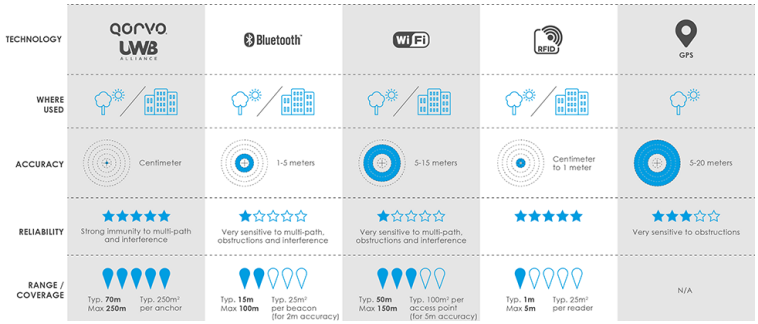

Figure 5: Qorov's Accuracy and Range Comparisons of Available Technologies [74]

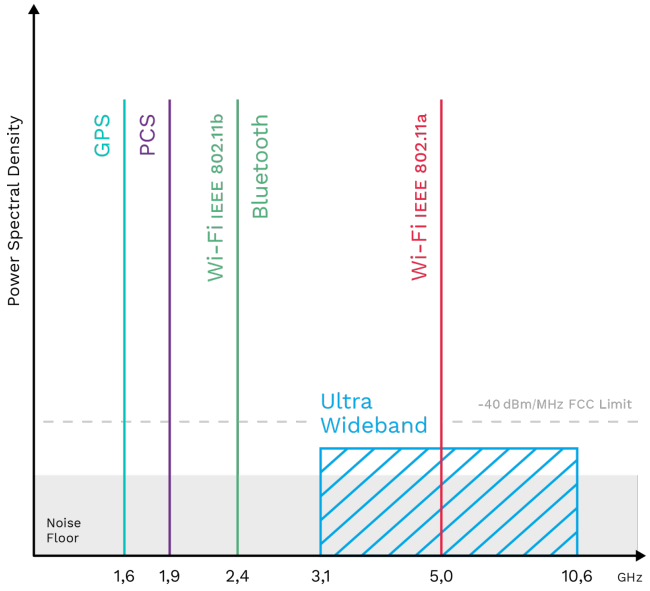

Figure 6: UWB Frequency Context [53]

up to 70 m [46, 74, 98]. This could explain Apple's AirTag localization protocol switching, from BLE to UWB, as soon as it is within a 2 m range of the targeted tag. The proximity of the tag ensures accurate localization even with obstructions such as cushions, chairs, or wardrobes. This would allow for two tracking scenarios. The first scenario, utilizing four permanent anchors to triangulate the position of all UWB tags within the predefined area and layout. The second scenario could utilize UWB tags as beacons, sending out messages to show an item's presence, allowing the precise localization of lost or hidden items, even within unknown area layouts.

3.3.4 Light Detection And Ranging (LiDAR)

LiDAR is a sensing technology that observes ranges (*i.e.,* distances) backscattered by a laser, in which a sensor is used to measure the backscattered light in the environment [61]. Thus, the distance can be estimated by measuring the time required for the laser light to return, given the speed of light. The basic principle for LiDAR originated in the early 1960s, and

military institutions drove its technical developments for measuring distances and weapon guidance [69].

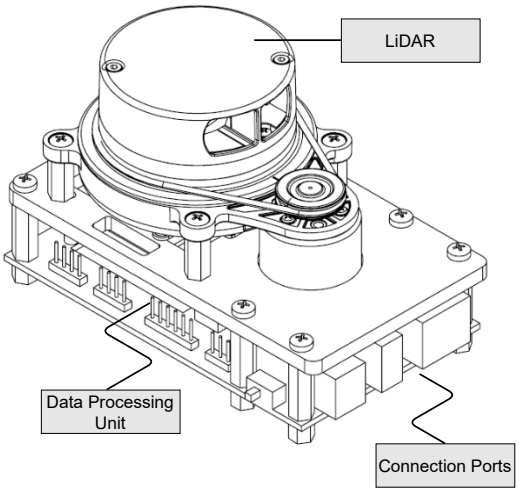

Figure 7: Slamtec Mapper M1M1 LiDAR Device [77]

The 2D LiDAR device used in PasWITS is the Slamtec Mapper M1M1 [77] (*cf.* Figure 7). It is a 2D LiDAR able to perform 7,000 measurements per second and achieve a maximum range distance of 20 m with a 5 cm resolution in indoor and outdoor environments. A Data Processing Unit (DPU) processes data in real-time, outputs a high-precision map, and poses with a maximum data mapping area of 90,000 m^2. Further, it provides a 10/100 Mbps Ethernet port, an 802.11a/b/g/n/ac WiFi module, and Universal Serial Bus (USB) for communication. Thus, it is suitable for tracking visitors in presentation stands without introducing deployment complexities.

In addition, PasWITS used the 3D LiDAR Intel Realsense L515, which adds the depth dimension for an indoor environment. The Intel Realsense L515 LiDAR is a small device that is convenient and easy for mobility. It is the world's smallest high-resolution LiDAR camera, only 100 grams. Without ambient light, It can capture accurate depth data in the range of 0.25m to 9m [47], which is ideal for indoor environment detection use cases.

Figure 8: Intel L515 3D LiDAR [47]

This type of camera (*cf.* Figure 8) has a wide depth field of view, $70° \times 55°$ ($\pm 3°$). For an indoor environment, it is wide enough to cover the detected areas in this project. The

detected image by LiDAR has a clear edge based on depth which is beneficial for object detection. However, drawbacks still exist. The most influential concern is the ambient light which greatly interferes with the detection range. Infrared light from sunshine could degrade the resolution of detected images and affect the measurement results. Thus, the extent of light affection is also considered and measured in this project.

In contrast to tracking solutions such as WiFi-based [75] and Bluetooth-based [76], LiDAR tracking offers the following **benefits**:

- **Precision:** LiDAR scanners present a high precision in comparison to WiFi-based and Bluetooth-based solutions. *E.g.,* [75] implements a WiFi-based system, which achieved a deviation of 1.1 m at a maximum distance of 10.8 m under good conditions since Received Signal Strength Indicator (RSSI) values are not reliable, whereas LiDAR devices can achieve a precision of 5 cm.
- **Capturing objects, not devices:** While the Android OS has enabled Medium Access Control (MAC) address randomization by default since version 10 [63], Apple has extended MAC address randomization with iOS 14 [45]. Thus, device fingerprinting will become increasingly difficult, not affecting LiDARs. Moreover, unlike Bluetooth or WiFi tracking, it allows tracking objects without an intelligent device.
- **Privacy:** Data protection regulations have become increasingly strict, which ”light” adheres to automatically. Tracking via WiFi or Bluetooth data could be complicated because the MAC address is classified as personal data according to the GDPR [20]. Thus, LiDAR tracking does neither process nor even hold information concerning tracked object identities.
- **Line of Sight:** LiDAR sensors can provide accurate results over long distances (up to 200 m in vehicles) [55]. Hence, LiDAR’s are often used in vehicles in combination with other sensors [90].

However, LiDAR techniques show **drawbacks** due to their dependency on the physical properties of light:

- **Hidden objects:** Due to the principle of operation, a LiDAR scanner cannot detect objects that are behind other objects. This limitation can be compensated by a second scanner capturing objects from a different angle.
- **Large data sets:** Depending on the resolution of the scanner (and also the aperture angle and the use case), the number of data points collected can become extensively large and data points deriving from static objects that are usually not of interest. Henceforth, the post-processing of LiDAR’s data can be more extensive than that of other measurement techniques (*e.g.,* WiFi-based [75] and Bluetooth-based [76]).
- **Robustness:** The performance of a LiDAR scanner strongly decreases in heavy rain or fog, which can reduce the detection rate of objects by up to 50% [30, 55].

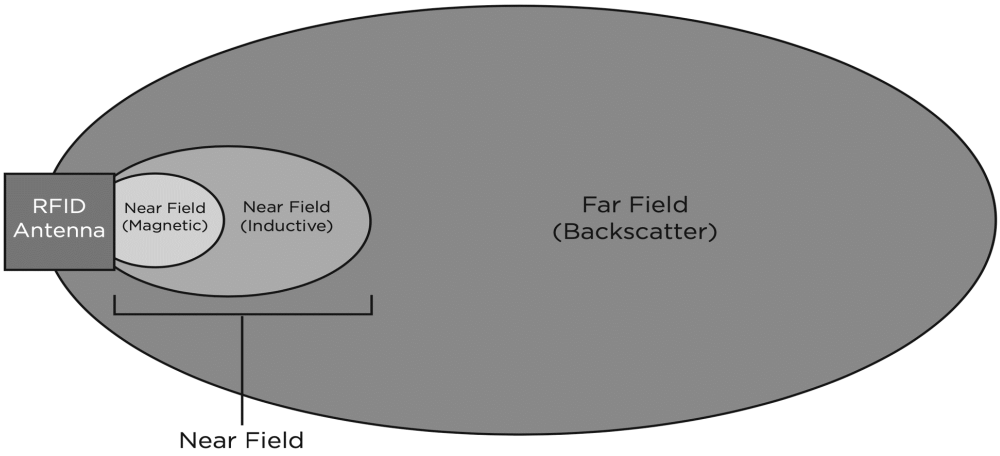

Figure 9: RFID Far-Field and Near-Field

3.3.5 Radio-Frequency Identification (RFID)

RFID (Radio Frequency Identification) refers to a technology, in which digital data is encoded in RFID tags and captured by one or multiple readers through radio waves. RFID belongs to the group of auto-ID technologies and it enhances them by allowing tags to be read without a line of sight. Depending on the type of RFID the reading range is up to or slightly above $20\ m$ [31]. Since data acquisition can be performed without human intervention, RFID tags' operation is efficient. Due to no line-of-sight requirements, the placement of tags on objects can be performed with fewer limitations, thus improving flexibility. In general, RFID tags are categorized as follows [50]:

- **Active Tags** receive energy from their power supply to operate, they can autonomously transmit data to the reader and cover greater distances than passive tags.
- **Passive Tags** do not operate on an energy source but receive power from the signal originating from the reader. They consist of a chip (with a unique identifier and memory), an antenna, and a support or container. Once a reader passes the tag, the respective radio frequency activates the microchip inside the tag to generate the energy needed to operate.
- **Semi-passive/Active Tags** are equipped with an energy source, which is used to power the microchip or even another device, such as sensors, but this does not power the transmitter. To be able to transmit information, the tag still must be within range of the reader/antenna.

Passive tags can be divided into *Near-Field RFID* and *Far-Field RFID* types, depending on the frequency band used to communicate and the Electromagnetic (EM) spectrum. (*cf.* Figure 9). In the near-field region, the interaction between components is dominated by the magnetic field generated by the antenna, which induces an electric current in the tag by inductive coupling and allows the chip to be activated. Tags of this type are part of the Low Frequency (LF) and High Frequency (HF) classes. In contrast, in the far-field region type the interaction of components is dominated by the EM field created by the antenna. The

RFID tag resonates with the frequency of the EM field and the current generated activates the chip. Tags of this type are part of the Ultra High Frequency (UHF) class.

RFID systems are attractive because of their relatively lower cost and technical potential [8]. For example, RFID has a clear advantage over Bluetooth in deployment cost (competitive tag cost), security (given one-way communication), and long-term maintenance cost [24, 34, 76]. Within the RFID domain are different types of RFID tags, namely, passive and active tags. The activeness refers to the need for a power source. In other words, an active tag can only be functional when a power source is supplied, whereas a passive tag can operate on its own without power [2]. Due to no power source, passive tags are less costly, have longer usage, and come in smaller sizes.

3.3.6 3D Camera

In the context of the analysis of people's movements, capacity, and engagement, a Xovis PC2 Camera [97] is utilized. Additionally, Xovis 3D cameras are equipped with a 3D sensor and two wide-angle lenses, which perceive the scene from different perspectives, achieving a precise depth map or 3D image of the entire scene. Both hardware and software components of this technology have to be compliant with the EU General Data Protection Regulation (GDPR) [89], and additional data protection laws, such as Federal Act on Data Protection (FADP) [29], which are applicable in countries of the European Union. Thus, the camera in use offers four levels of privacy protection: While level 0 of privacy has no restrictions, only a still image is shown in level 1 with insights on a person's path. Level 2 disables the video stream's functionality, allowing only path tracking of visitors without revealing their identities. In level 3, no tracked person paths' are visible anymore, which makes it compelling for use in places where strict privacy protection regulations apply.

3.3.7 Multi-lateration

Multi-lateration defines the process of geometrically estimating an object's position in space through distance measures to at least three points. It is called trilateration a case where exactly three points are used. Mathematically, this corresponds to solving the following non-linear system, with (x_i, y_i, z_i) the position of the i-th point, (x, y, z) the position of the object, and d_i the distance of the object to the i-th point. For planar problems, this can be simplified further, leading to the following system in two variables:

$$\begin{aligned}(x-x_1)^2+(y-y_1)^2+(z-z_1)^2&=d_1^2\\(x-x_2)^2+(y-y_2)^2+(z-z_2)^2&=d_2^2\\(x-x_3)^2+(y-y_3)^2+(z-z_3)^2&=d_3^2\end{aligned} \tag{1}$$

For planar problems, this can be simplified further, as is illustrated in Figure 10. This leads to the following system in two variables.

$$\begin{aligned}(x-x_1)^2+(y-y_1)^2&=d_1^2\\(x-x_2)^2+(y-y_2)^2&=d_2^2\\(x-x_3)^2+(y-y_3)^2&=d_3^2\end{aligned} \tag{2}$$

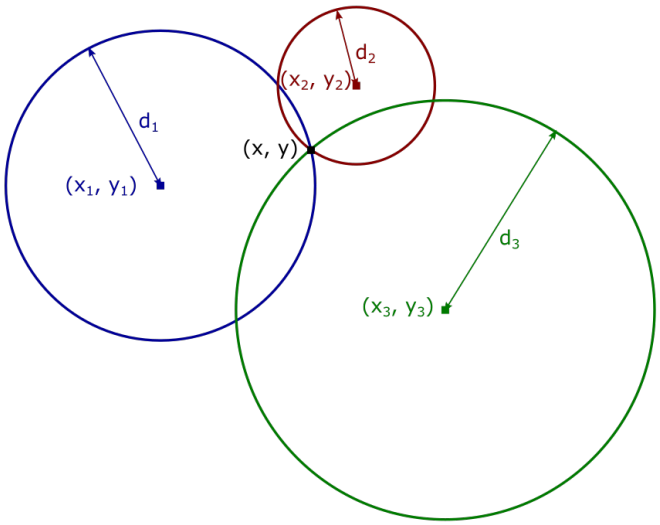

Figure 10: A Planar Trilateration Problem

This system is then often linearized by subtracting the last equation from the other two, leading to the following determined linear system of equations [59].

$$\begin{aligned} 2(x_3 - x_1)x + 2(y_3 - y_1)y &= d_1^2 - d_3^2 + x_3^2 - x_1^2 + y_3^2 - y_1^2 \\ 2(x_3 - x_2)x + 2(y_3 - y_2)y &= d_2^2 - d_3^2 + x_3^2 - x_2^2 + y_3^2 - y_2^2 \end{aligned} \tag{3}$$

The solution is reached analogously for any higher-order multilateration problems. In practice, distance measures are often imperfect, and the calculation of a solution for Equation 2 using a non-linear optimization leads to better results.

3.3.8 The Log-Distance Path Loss Model

The Log-Distance Path Loss Model is a popular model for radio signal decay over distance [3]. It models the finding that a logarithmic function can approximate the decay of a signal over distance. With $RSS(d)$ the received signal strength at distance d, d_0 a reference distance, n the path-loss coefficient and $\mathcal{X}_\sigma$ a zero-mean Gaussian random variable, it can be defined as follows:

$$RSS(d) = RSS(d_0) - 10n \log\left(\frac{d}{d_0}\right) + \mathcal{X}_\sigma \tag{4}$$

In practice, the reference distance d_0 is often set to 1 meter, and noise is ignored for the calculation, simplifying the model even further. With RSS_C, the received signal strength at 1 meter, it can then be expressed as follows:

$$RSS(d) = RSS_C - 10n\log(d) \tag{5}$$

RSS_C is dependent on each device and has to be calibrated for. The path-loss coefficient n is a factor that depends on the environment. For free-space, it is often chosen at $n = 2$.

3.3.9 Kalman Filtering

The Kalman Filter describes a type of Bayesian filter. Bayesian filters "probabilistically estimate a dynamic system's state from noisy observations" [33]. With the state at time t represented as random variable s_t, it can be expressed mathematically as finding the probability distribution over s_t, which we then call *belief* $Bel(s_t)$. With sensor observations over time $z_1, z_t, ..., z_t$, the belief is defined as follows:

$$Bel(s_t) = p(s_t|z_1, z_2, ..., z_t) \tag{6}$$

Bayes' theorem tells us that this belief can also be expressed in the following way [56]:

$$p(s_t|z_1, z_2, ..., z_t) = p(s_t|z) = \frac{p(z|s_t) * p(s_t)}{p(z)} \tag{7}$$

$p(z|s_t)$ is the probability of making the observations z, given that we are in state s_t. $p(s_t)$ is the probability of being in the state s_t before our knowledge of the observations z. Finally, $p(z)$ is the probability of observing z without any restrictions by the state s_t. This theorem is often expressed in the more high-level way

$$\text{posterior} = \frac{\text{likelihood} * \text{prior}}{\text{evidence}} \tag{8}$$

The Kalman Filter is then nothing more than a Bayesian filter that assumes the probability distributions to be Gaussian. It is further detailed in [52, 56].

With

s_k The state at time k

z_k The observation at time k

F_k The state transition model at time k

H_k The observation model at time k

Q_k The process noise covariance at time k

R_k The observation noise covariance at time k

we can define the state at time k as

$$s_k = F_k s_{k-1} + w_k, w_k \sim \mathcal{N}(0, Q_k) \tag{9}$$

and the observation at time k

$$z_k = H_k s_k + v_k, v_k \sim \mathcal{N}(0, R_k) \tag{10}$$

3.3.10 The Kalman Filter Algorithm

We can now define the Kalman Filter algorithm as an iterative algorithm in two steps: the predict step and the update step [52] [56]. In the following, the hat operator "$\wedge$" will denote an estimate of a variable, the superscript "$-$" will denote a predicted (prior) and the superscript "$+$" an updated (posterior) estimate.

During the **Predict Step**, we calculate the *predicted state estimate* $\hat{s}_k^-$ and the *predicted error covariance* P_k^-

$$\hat{s}_k^- = F_k s_{k-1}^+ \tag{11}$$

$$P_K^- = F_k P_{k-1}^+ F_k^T + Q_k \tag{12}$$

We correct the predictions using our measurements during the **Update Step**. We first calculate the *measurement residual* $\tilde{y}_k$ and its covariance S_k.

$$\tilde{y}_k = z_k - H_k \hat{s}_k^- \tag{13}$$

$$S_k^- = H_k P_k^- H_k^T + R_k \tag{14}$$

We can then calculate the *Kalman gain* K_k.

$$K_k = P_k^- H_k^T S_k^{-1} \tag{15}$$

Finally, we can compute the *updated state estimate* $\hat{s}_k^+$ and the *updated error covariance* P_k^+

$$\hat{s}_k^+ = \hat{s}_k^- + K_k \tilde{y}_k \tag{16}$$

$$P_k^+ = (I - K_k H_k) P_k^- \tag{17}$$

3.3.11 Similarity Algorithms

The correlation of multiple data sources is a fundamental aspect based on the position reported by different input vectors, including their respective timestamps. Correlating tracking sources based on known similarity algorithms are summarized in Table 1.

Equation 18 defines the Euclidean distance, calculated by taking the square root of the sum of the squared pair-wise distances of every dimension. Distances (x, y) reported by different tracking sources within a rolling time window (based on the timestamp of each source) are compared in a simple Cartesian plane. The smaller the distance, the larger the correlation is. Equation 19 is similar to the Euclidean distance but calculates the distance between two vectors by considering the diagonal path ("beeline") between two points, *i.e.,* considering a grid line.

As with a slightly different approach, the cosine similarity measures the orientation of two n-dimensional sample vectors irrespective of their magnitude. It is calculated by the dot product of two numeric vectors, and it is normalized by the product of the vector lengths such that output values close to 1 indicate high similarity. The Pearson correlation coefficient (*cf.*

Table 1: Overview of Similarity Algorithms. Based on [35]

Algorithm	Short Description	Equation
Euclidean	Distance between 2 points in a multidimensional space. Takes the square root of the sum of the squared pair-wise distances of every dimension	$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$ (18)
Manhattan	Similar to the Euclidean distance but the distance is calculated based on grid lines, taking the shortest diagonal path	$\sum_{i=1}^{n}\|x_i - y_i\|$ (19)
Cosine	Measures the orientation of two sample vectors irrespective to their magnitude. Calculated by the dot product of two numeric vectors	$\frac{x \cdot y}{\|\|x\|\| \cdot \|\|y\|\|}$ (20)
Pearson	Obtained via a Least-Squares fit and a value of 1 represents a positive relation, -1 a negative relation, and 0 indicates the absence of relation	$\frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2(y_i - \overline{y})^2}}$ (21)

Equation 21) is the most widely used measure for linear relationships between two standard distributed variables and, thus, is often just called the “correlation coefficient”. Usually, the Pearson coefficient is obtained via a least-squares fit. A value of 1 represents a perfect positive relationship, -1 defines a perfect negative relationship, and 0 indicates the absence of a relationship between these variables.

4 Design and Prototype

PasWIT's design starts with an overview, followed by the standard data structure defined. While the date input is refined in all levels of detail, the Fusion Data Tracking System (FITS) is determined as the proposed architecture. Finally, the data consumption perspective is taken and detailed.

4.1 PasWITS Overview

PasWITS is based on the correlation of different passively emitted signals in events and exhibitions to extract KPIs in relation to the public interest of a product or service to a particular type of audience. Considering that different sources of signal emissions (*e.g.,* Wireless 802.11b or 802.11g and Bluetooth Low Energy - BLE) have different advantages and disadvantages, the combination of these in the same medication scenario allows for reducing its disadvantages. Figure 11 illustrates the different architectural functions of the project in which there are three different modules: input, back-end, and front-end.

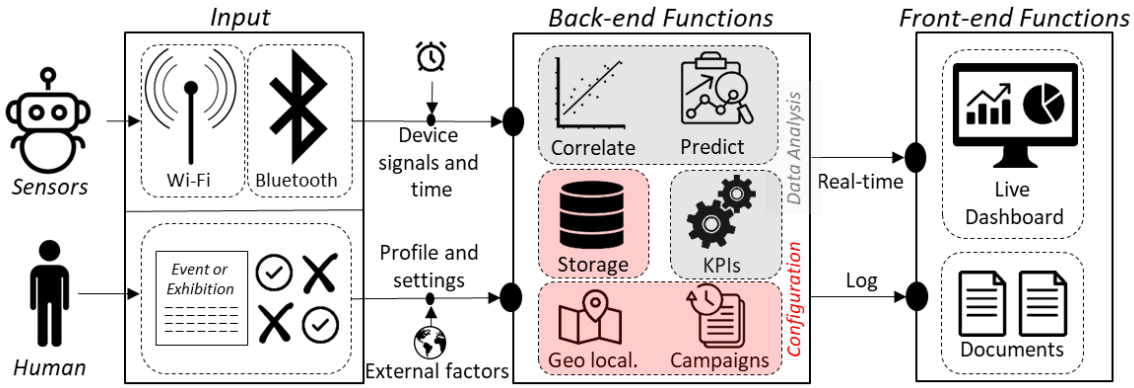

Figure 11: PasWITS Architectural Functions

- **Input**: requires both the positioning and configuration of the different sensors used, as well as manual input about details of the event or exhibition. Depending on the physical characteristics of the place where the event or exhibition takes place (open, closed, positioning of the target stand), different sensors can be used to measure the public's interest in the stand.
- **Back-end**: considers the operation in two modes: streaming and post-processing. In the streaming operation mode, an API (*e.g.,* HTTP/MQTT) would be available so sensors can send data to be correlated in a rolling time window. In post-processing, the data is correlated in a single batch for the event or display under analysis. In both modes, KPIs (*e.g.,* visibility, engagement, interaction rate, bounce rate) are calculated and the analyzed data is stored with the configured characteristics forming the base of prediction of events with similar characteristics.

- **Front-end**: allows viewing KPIs and event configurations in a real-time dashboard and generating logs and documents about the performance of the event or exhibition. In addition to configuring sensors that often involve specific frameworks and procedures for deployment, the front-end is the main means of user interaction to configure event details and view their performance through data obtained from the backend.

4.2 Standard Data Structure

Different sensors use different formats to describe their measurements. A typical data standard (*cf.* Table 2), and data transformation steps, are necessary to correlate measurements obtained from sensors of different types.

Location. Tracking is achieved by sensors that are installed in the location at certain positions. Defined as an area where the PasWITS is installed to track the presence and movement of visitors. A location is modeled by a coordinate system representing the area covered by the sensors. A location's data structure is defined by a name, an optional external identifier, and, most importantly, a list of physically installed sensors at the location.

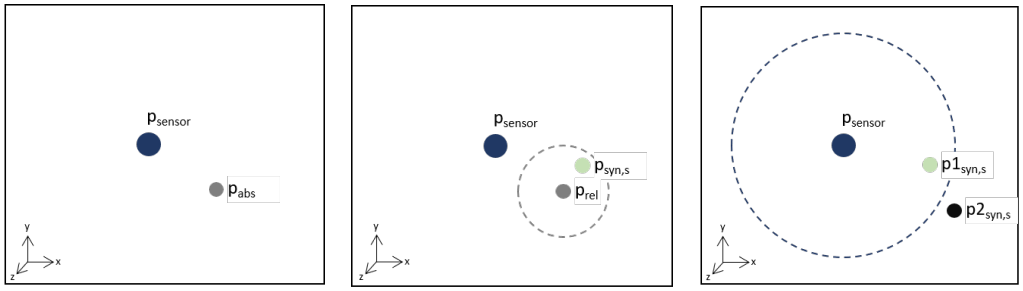

Figure 12: Exemplary Synthetic Measurement Generation Process. Left: True Sensor Position (p_sen), Center: Relative Coordinates (p_{rel}) and Synthetically Generated Measured Position ($p_{syn,s}$). Right: Two Randomly Generated Measurements; One ($p2_{syn,s}$) being Rejected

Sensor. At a pre-defined location, sensors are installed to capture visitors currently present in the location. Each sensor is assumed to have an internal coordinate system, which is used to express the *X*, *Y*, and *Z* coordinates of visitors they have tracked. To combine the measurements of multiple sensors, their measurements must be translated from their *relative* coordinate system to a common, *global* coordinate system. The common coordinate system of sensors is the location's coordinate system. To facilitate this transformation, information about the sensor's position and orientation within the location are required. Therefore, the sensor data structure contains the *X*, *Y*, and *Z* coordinates of the sensor's origin position related to the location's coordinate system. Further, its orientation is captured in the *Pitch*, *Yaw*, and *Roll* properties, which represent the rotation offset in comparison to the location's coordinate system orientation. This information enables the calculation of the coordinate offsets and rotations. For instance, consider a sensor installed on the ceiling, facing down, which means the sensor's z-axis points to the floor as opposed to the location's z-axis, which points in the opposite direction. This particular setup would translate to a roll of 180 degrees.

Table 2: Data Structure

Property	Description
Location	
name	Name of the location (*e.g.,* Room number)
external_identifier	Field to store an id of an external system
sensors	List of sensors installed at the location
Sensor	
sensor_identifier	Unique identifier of the sensor
type	Sensor type (*e.g.,* RFID, Wi-Fi, Camera)
measurement_unit	Length unit of the measurement results (*e.g.,* 'm', 'cm', 'mm')
x_origin	X-coordinate of the sensor position, expressed in the location's coordinate system
y_origin	Y-coordinate of the sensor position, expressed in the location's coordinate system
z_origin	Z-coordinate of the sensor position, expressed in the location's coordinate system
yaw	XY-plane rotation angle compared to the location's coordinate system orientation
pitch	XZ-plane rotation angle compared to the location's coordinate system orientation
roll	YZ-plane rotation angle compared to the location's coordinate system orientation
Measurement	
object_identifier	Identifies the object captured by the sensor
sensor_identifier	Identifier of the sensor which recorded the measurement
x	Measured x-coordinate (sensor internal coordinate system)
y	Measured y-coordinate (sensor internal coordinate system)
z	Measured z-coordinate (sensor internal coordinate system)
timestamp	Timestamp the measurement was recorded
Prediction	
object_identifier	Identifies the object captured by the sensor
x	Predicted x-coordinate (location coordinate system)
y	Predicted y-coordinate (location coordinate system)
z	Predicted z-coordinate (location coordinate system)
timestamp	Timestamp the measurement was recorded

Measurement. Each sensor produces measurements it captures in its reading range. A measurement contains a timestamp, the sensor's id, an object id, and the measured X, Y, and Z coordinates. Depending on the sensor type, the object ID is assigned by the sensor itself or determined by the object. A camera sensor, for instance, uses its indexing scheme to identify objects, whereas wireless-based sensors usually use MAC addresses. The coordinates are expressed relative to the sensor's internal coordinate system.

Prediction. Based on the measurement data collection, the system identifies unique visitors by matching object IDs captured and assigned by the different sensors. It fits them into clusters, mapping back to the unique real-world visitors. The system predicts the path or timeline a visitor traveled in the period covered by the sensor measurements for each identified visitor. The timeline is represented by timestamped position estimates (*x, y, z*), expressed in the location's coordinate system.

4.3 Data Input

The input refers to all types of data sources and configurations required to correlate multiple data sources, as well as, to predict the position and movement of certain objects. Terms such as object and device are often depending on the input source as these are based on different tracking methods. For example, Bluetooth and Wireless tracking methods are based on signals emitted passively by mobile phones whereas vision-based technologies (*e.g.,* LiDAR) track moving objects that correspond to moving persons within their tracking range.

4.3.1 WiFi — ASIMOV

As mentioned in the section on fundamentals (*cf.* Subsection 3.3.1), mobile phones are constantly probing nearby access points by sending probe-request packets. These packets contain important information concerning the physical address of the phone (MAC address) but are constantly switched by the mobile operating systems in an approach termed MAC randomization.

ASIMOV circumvents such challenges by using available localization and Information Elements (IE) to determine whether traffic captured originates from the same device passively or not. Thus, enabling the tracking of devices even when they use a MAC address randomization. For this, the prototype ASIMOV was implemented. Unlike previous deanonymization approaches, such as *NiFi [18]* and *Wobly [60]*, ASIMOV does not rely solely on specific data fields that are not assumed to be stable over time or universally equal from device to device. On the contrary, ASIMOV uses a combined Received Signal Strength Indicator (RSSI) value-based localization and the IE transmitted in every IEEE 802.11 probe request frame. ASIMOV is entirely passive and can determine how many devices are present and track over time a single device in the area covered.

The ASIMOV's approach consists of a step to distinguish devices, divided into *Data Gathering* and *Data Analysis*. Both processes are started by the user, who manages devices and the processes through an intuitive management interface. The data gathering process orchestrates components to obtain data from available devices, aggregate these data, and store relevant data. This process starts with monitor nodes being configured to capture devices' signals within a specified area. Monitor nodes dump Probe Requests (PR) received at the specified Network Interface Card (NIC), extract relevant data fields, such as MAC address, timestamps, RSSI values, and IEs, and return them to the Sync node. Sync nodes aggregate and store data from monitors in a shared database for posterior analysis, interacting with the Interface on one side and with the Monitor nodes on the other side.

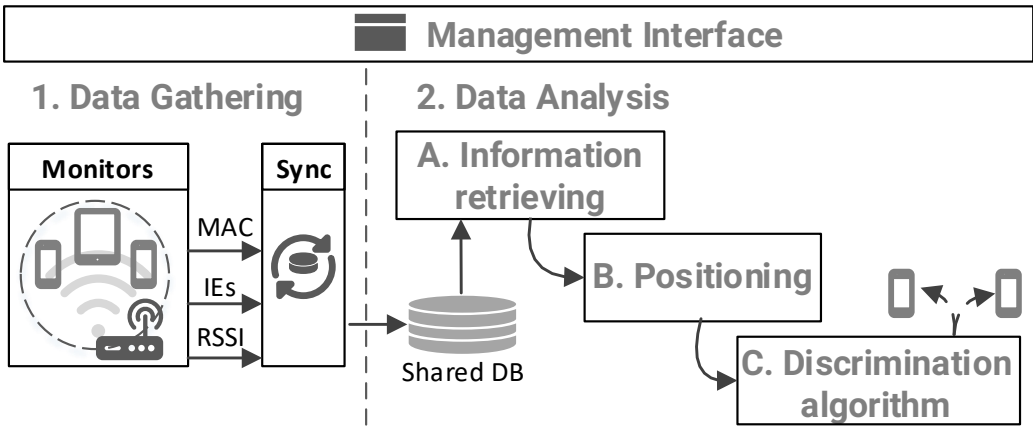

Figure 13: ASIMOV Architecture and Process to Distinguish MAC-randomized Devices

A. Information Retrieving The data analysis process is composed of three steps. In the information retrieving step, the system loads data from the shared database for posterior analysis: all packets are reassembled, such that for each PR, the RSSI values are measured at different monitor nodes are stored. ASIMOV uses a multi-step approach, that combines signal strength-based device localization with IEs. It, therefore, combines an external feature, with a feature, that is content-based and device-specific. Instead of relying solely on probe request features, and struggling with the potentially small variance of IEs, ASIMOV tries to increase the informative value of the IE, by adding the location estimate as an extra source. Through this mechanism, it becomes possible to tell whether two probe requests are possibly originating from one or multiple devices and ultimately to count present devices. The necessary steps to make these decisions are described in Algorithm 1.

B. Positioning The positioning step uses RSSI information to localize devices using multilateration. To localize a device, RSSI values received at the different APs can be used to estimate a distance in meters using the log distance path-loss model.

$$PL = PL_0 + 10 * \gamma * \log_{10} \frac{d}{d_0} + X_g \tag{22}$$

The basic equation of the log distance path-loss model sets the path-loss(PL), represented by the RSSI value, in relation to several other factors [67]:

- PL_0: The path-loss at reference distance d_0
- γ: The path-loss exponent, a factor to incorporate different environmental factors
- d: The length of the path
- d_0: The reference length
- X_0: A normal (or Gaussian) random variable with zero mean, reflecting the attenuation caused by path fading mechanisms.

Applying the path loss equation 22 to the measured RSSI values at the different receivers results in an approximation of the distance the sender has to each AP. Often, however,

Algorithm 1: Decision-making process

```
Result: Circumvent MAC address randomization based bias
while new packets are captured do
    if packet is captured by multiple APs then
        extract MAC address of sender and IEs;
        if database contains combination of MAC address and IEs then
            The device is not doing MAC address randomization;
        else
            if database contains equal IEs then
                Create location estimate;
                Compare current location with last location;
                if locations close enough then
                    Devices are the same device doing MAC address randomization;
                    Add MAC address as alias to previous MAC address.
                else
                    Devices are not the same;
                    Add to the database as new device;
                end
            else
                Add data to database;
            end
        end
    else
        Discard packet;
    end
end
```

multiple measurements exist for the same position and more than the minimum of three APs are receiving a signal from the same sender. The ASIMOV system profits from the excess data by incorporating as much information as possible into the multilateration process. The emerging prediction error, resulting from an over-specified system of equations, is minimized by using a non-linear least-squares approximation. This solution allows to fit a set of m observations (more than three RSSI values) with a model that is non-linear in n unknown parameters [95]. In this process, all unknown parameters, including parameters from the path-loss formula, can be approximated.

C. Unique identification ASIMOV determines whether the original device is applying MAC address randomization. The algorithm relies on snapshots of measurements over time, combining knowledge from the current snapshot and the aggregation of past snapshots, to classify devices as "not seen before", "randomizing" if they are randomizing their MAC addresses, or "non-randomizing," if devices do not apply MAC randomization at all.

1. **Already seen**: Devices that can be identified as known, based on their IEs and their location.

2. **Not seen before**: Devices that have yet unknown IEs.
3. **Randomizing**: Devices that are randomizing their MAC addresses
4. **Non-Randomizing**: Devices that are not applying MAC address randomization

4.3.2 Bluetooth — BluePIL

BluePIL (Bluetooth Passive Indoor Localization) introduces a distributed streaming architecture that uses a node-sink topology to deliver near-real-time positioning estimates of BT devices. It defines a data processing pipeline accomplishing identification and localization tasks through passively captured BT Basic Rate/Enhanced Data Rate (BTBR/EDR) packets in several steps, *i.e.,* device identification, signal strength filtering, signal strength merging, the localization algorithm, and location filtering. BluePIL is based on a Python implementation running on low-cost hardware (*e.g.,* Ubertooth Devices and small computers, such Raspberry Pi's or ASUS Tinkerboard), and requires a minimal setup, since configurations are handled automatically.

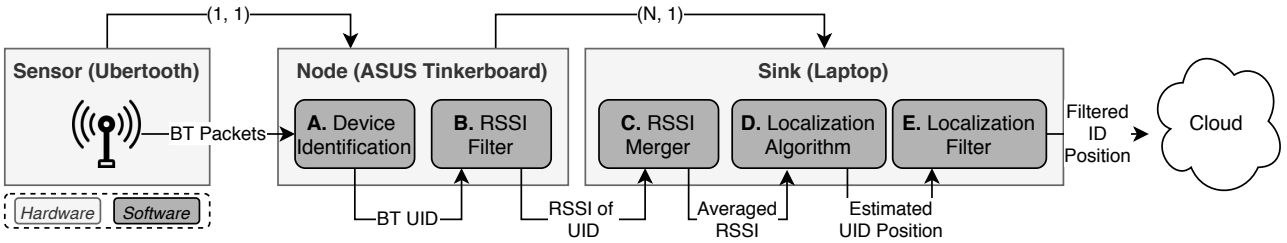

Figure 14: BluePIL's Data Stream Pipeline

Figure 14 describe the *BluePIL* data pipeline, designed toward a flexible deployment and operation of individual hardware components (*i.e.,* sensors, nodes, and sink). In general, *BluePIL* is set up as a streaming data processing pipeline. It allows for physical or virtual logical processing entities in a system deployed to be configured in different ways. *BluePIL* is based on a distributed node-sink setup (*i.e.,* , a system where many physical nodes send data to a single physical sink), which is responsible for forwarding data to an entity, where it can be stored or processed (*e.g.,* the cloud). Computations are performed as early as possible to avoid bottlenecks downstream and to reduce the amount of data forwarded by the sink.

A. Device Indentification The identification step allows the system to profit from the fact that BTBR/EDR lacks any sort of MAC randomization and avoids building complex systems for fingerprinting by using a unique identifier that is already available: the BT address. The BT address consists of the LAP, the NAP, and the UAP. These three address parts and their usefulness for the device identification problem are discussed separately in the following. The NAP is, as the name says, not significant, and can, therefore, not be derived from passively captured BT traffic. [82] describes an approach, where possible NAP values are selected heuristically from the list of all manufacturer OUIs. These values, however, have to be validated through a targeted inquiry request, which breaks with the passive nature of

BluePIL. The LAP is easily obtainable from passively captured BT traffic. It is contained in the CAC in any packet and can be read without the need for any further processing [13, 82].

While LAP is not globally unique, it is sufficient to identify devices under certain circumstances [13, 19, 82]. As *BluePIL*'s goal is to identify mobile devices, such as smartphones or tablets, it is important to account that the five biggest smartphone manufacturers share 72% of the smartphone market among them (as of the first quarter of 2020 [48]). Considering *a* the probability of encountering a LAP collision as $P(col)$, *b* the probability of encountering a different OUI as $P(dO)$, and *c* the probability of encountering the same CID $P(sC)$, it can be stated that encountering the same LAP twice means that their OUI is different since BT addresses are globally unique. Thus, the probability of a LAP collision is defined in terms of $P(col) = P(sC) * P(dO)$. Even without any further optimization, this gives a fairly small probability of around $P(col) \approx 5.96\mathrm{e}^{-8}$.

Assuming that the 20 largest smartphone manufacturers share (almost) the entire market, $P(dO) \approx \frac{19}{20}$ and $P(col) \approx 5.66\mathrm{e}^{-8}$ holds. Thus, if in a certain environment the system would register 10,000 different BT addresses, for example, the probability for a LAP collision would still only be at $1 - (1 - P(col))^{10{,}000} \approx 0.06\%$. This is sufficient for the potential use cases of *BluePIL* and, therefore, the LAP as computed in [82] is used as a quasi-unique identifier. This identifier is also suitable to identify individuals carrying a BT device, since all devices in a piconet use the master device's LAP for the construction of the access code, *i.e.,* two connected devices, such as a smartphone and a pair of BT headphones, do not produce two separate identifiers.

B. Device Indentification RSSI values obtained from the sensor are pre-processed as a first step. Figure 15 shows an example of RSSI measurements for a static device over a period of five minutes. This example illustrates the large amounts of high-frequency, high-variance noise that must be taken into account when working with this type of data. While part of the noise can be attributed to inaccuracies of the sensor, a significant amount of disturbance originates from the effects of multipath fading, *i.e.,* a signal may travel along multiple paths towards a sensor that diverges from the most direct path, the line of sight. This effect is caused, for example, by reflections of the signal on surfaces in the surroundings of the target device and the sensor. It makes the RSSI a difficult value to work with since the distribution of this noise is not Gaussian, an assumption that many filtering and smoothing approaches work under. The main goal of this step in the processing pipeline is, therefore, the elimination of the noise caused by multipath fading and the conversion of the noise distribution to a Gaussian one.

Existing research suggests that noisy parts of RSSI values correspond to the lower set of values in the RSSI distribution. [16], defines a unidirectional outlier filter to be effective. It eliminates values that deviate from the maximum value by a certain degree. [28] determines the maximum to be the most effective filter for a pre-processing of RSSI values for localization purposes.

The signal that travels along the line of sight, *i.e.,* that is not influenced by multi-path fading, covers the smallest distance and, thus, arrives at the sensor with the highest strength. A combination of a maximum filter followed by a mean filter is, thus, used in *BluePIL*. To account for the streaming paradigm, these filters work in a purely retrospective way, *i.e.,* work with a local subset of the data that only uses values from the past. To this end, a rolling

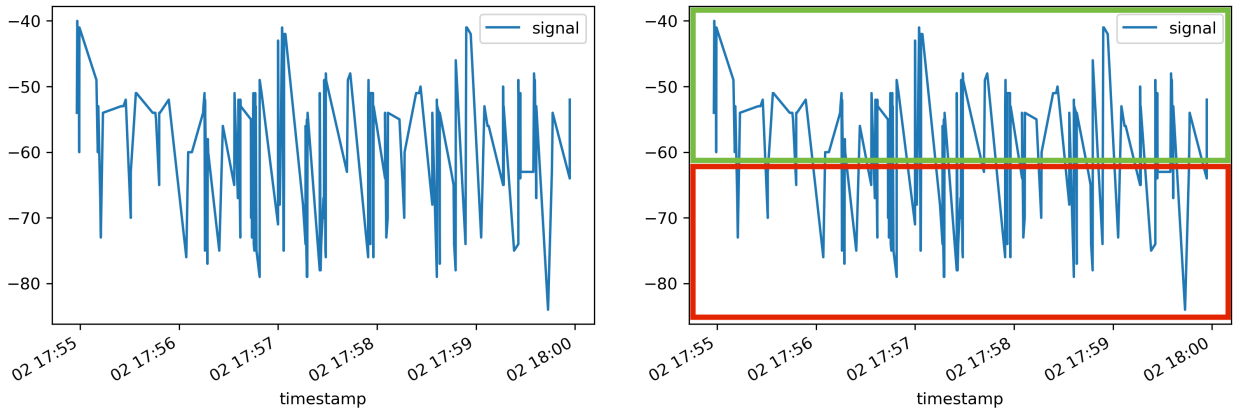

Figure 15: (Left) RSSI Measurements for a Static Device Over a Period of 5 mins Using an *Ubertooth* Sensor, and (Right) RSSI Values Potentially Useful (Top Box/Green) and Those Probably Caused by Multi-path Fading (Lower Box/Red)

time window is implemented only containing values from the interval $[t_c - \Delta t, t_c]$, with t_c being the current time and Δt the window size, which are determined by the update frequency of the sensor and the expected variance of the data.

C. RSSI Merger To compute a location from pre-processed RSSI values, a strategy has to be determined to merge data streams. This part of the processing pipeline deals with two problems: First, update cycles may differ between sensors, *i.e.,* it cannot be assumed that all sensors will have the same amount of data available at a specific point in time. Second, data delivered by sensors may be fairly sparse. This may be due to the quality and capabilities of sensors themselves, due to environmental factors or due to characteristics of the target device. The goal of this step is to handle these problems, taking into account the streaming paradigm implemented for *BluePIL*.

Interpolation is able to help with both the problem of differing update cycles and sparsity of data. In general, *BluePIL* builds upon the assumption that update cycles of individual sensors are short enough to legitimize the linear interpolation between two data points as a valid estimation of the true state of the system. To enable the inference of RSSI values at a certain point in time through interpolation, measured values must be available preceding and succeeding said point. The signal strength merger will, therefore, delay the emission of a value from a sensor until data is available from all other sensors before and after the point in time, where the value was received.

D. Localization Algorithm Since *BluePIL* is completely passive, information is limited to the signal strengths detected on an external sensor from any ongoing BT connection. This rules out any fingerprinting-based approaches since they require the creation of a radio map with the devices involved beforehand, leaving the path-loss-model-based approaches. A path loss model requires parameters n and RSS_C to be defined beforehand in order to calculate a distance from a signal strength value. Based on existing research [16, 26, 88], it is viable to set n to a fixed value based on the environment *BluePIL* is working in, as long as this does not change drastically. n is dependent on environmental factors and does not

vary between devices. The issue with RSS_C, however, is not so easy to solve. Transaction strengths may vary between BT devices. Due to adaptive power control, they may even change over time for the same device [13]. The choice of a fixed value for RSS_C is, therefore, not an option.

To approach this, a method was designed that dynamically estimates the location of a BT device and the necessary channel parameters of the path loss model. With k the number of sensors, (x_i, y_i) the location of the i-th sensor, (x, y) the location of the target device, and d_i the distance between the i-th sensor and the target device, the following multilateration problem is defined:

$$(x - x_i)^2 + (y - y_i)^2 = d_i^2, i \in 1..k \tag{23}$$

Since it is not possible to compute d_i from the path loss model directly due to the issues mentioned before, the path loss model equation for distance is solved and then combined with the multilateration problem above:

$$\begin{aligned} RSS(d) &= RSS_C - 10n\log(d) \\ d &= 10^{\frac{RSS_C - RSS(d)}{10n}} \end{aligned} \tag{24}$$

$$(x - x_i)^2 + (y - y_i)^2 = 10^{\frac{RSS_C - RSS(d_i)}{5n}}, i \in 1..k \tag{25}$$

A non-linear set of k minimizable equations is defined in terms of x, y and RSS_C, with RSS_C the calibration signal strength 1 meter away from the target device and RSS_i the RSS measurement for the i-th sensor. This corresponds to a problem that can be solved using a non-linear optimization algorithm. *BluePIL* uses *Levenberg-Marquardt (LM)*, an iterative minimizer that can be described as a combination of the *Steepest Descent* and the *Gauss-Newton* methods [64] [66]. With $\mathbf{p} = (x, y, RSS_C)$, the parameter vector, the vector $\mathbf{p}^+$ is determined where $f_i(\mathbf{p}^+)$ is minimal for all i. LM works through a local linearization of the non-linear set of equations at a certain area of interest according to the statement $f(\mathbf{p}+\delta_p) \approx f(\mathbf{p}) + \mathbf{J}\delta_p$, where $\mathbf{J}$ is the *Jacobian* matrix. For *BluePIL*'s set of equations, the Jacobian matrix is defined as:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial RSS_C} \\ \vdots & \vdots & \vdots \\ \frac{\partial f_i}{\partial x} & \frac{\partial f_i}{\partial y} & \frac{\partial f_i}{\partial RSS_C} \\ \vdots & \vdots & \vdots \\ \frac{\partial f_k}{\partial x} & \frac{\partial f_k}{\partial y} & \frac{\partial f_k}{\partial RSS_C} \end{bmatrix} = \begin{bmatrix} 2(x - x_1) & 2(y - y_1) & -\frac{log10}{5n} * 10^{\frac{RSS_C - RSS_1}{5n}} \\ \vdots & \vdots & \vdots \\ 2(x - x_i) & 2(y - y_i) & -\frac{log10}{5n} * 10^{\frac{RSS_C - RSS_i}{5n}} \\ \vdots & \vdots & \vdots \\ 2(x - x_k) & 2(y - y_k) & -\frac{log10}{5n} * 10^{\frac{RSS_C - RSS_k}{5n}} \end{bmatrix} \tag{26}$$

With the Jacobian defined, LM then iteratively adjusts **p** by δ_p in a descending direction until convergence is reached. To ensure that this convergence is to a global minimum, an appropriate starting point $\mathbf{p}_0$ has to be defined. For the problem posed, it is important that the minimum is found in the area of overlap of all sensors. To guarantee this, $\mathbf{p}_0$ is chosen at the center of the area spanned by the sensors and with a value for RSS_C that approximates the range of values that are expected from the relevant device class.

$$\mathbf{p}_0 = (\frac{\sum_i x_i}{k}, \frac{\sum_i y_i}{k}, -30) \tag{27}$$

Due to the limited resources available, the problem is generally limited to four sensors, *i.e.* $k = 4$. The localization algorithm may, therefore, also be referred to as a *quadlateration* algorithm in the following.

E. Localization Filter After having calculated a location in the previous step, the knowledge of the motion of a person carrying a BT device can be used to improve these results further. Kalman filters are a popular method for the improvement of positioning calculations and have been used in various path-loss-based localization approaches [16] [100] [59]. They combine models for the state of the system, the knowledge of previous observations, and models for the observation of states to estimate the most plausible state of a system captured through noisy observations.

To use a Kalman filter, it is necessary to define the following: the state transition model F_k, the observation model H_k, the process noise covariance Q_k, and the observation noise covariance R_k. *BluePIL* uses a simple kinematic model with (x_k, y_k) being current location's coordinates, and $(\dot{x_k}, \dot{y_k})$ the current velocity in x and y direction. The state vector is designed in a similar manner to [100]:

$$s_k = \begin{bmatrix} x_k \\ \dot{x_k} \\ y_k \\ \dot{y_k} \end{bmatrix} \tag{28}$$

With Δt_k being the time difference to the last state estimate s_{k-1}, the following state-transition matrix is defined:

$$F_k = \begin{bmatrix} 1 & \Delta t_k & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t_k \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{29}$$

This expresses a belief that the subject carrying a BT device will have moved in the direction gathered from the last measurement and that the velocity of said movement will not have changed abruptly. For *BluePIL*'s process noise covariance, it is used a discrete white noise as suggested in [56] and [7], under the assumption that the noise is a Wiener process, *i.e.,* is independent of previous time intervals and constant over a time interval. With the variance $\sigma_v^2 = 0.001$ [56], it is defined as follows:

$$Q_k = \begin{bmatrix} \frac{1}{4}\Delta t_k^4 & \frac{1}{2}\Delta t_k^3 & 0 & 0 \\ \frac{1}{2}\Delta t_k^3 & t_k^2 & 0 & 0 \\ 0 & 0 & \frac{1}{4}\Delta t_k^4 & \frac{1}{2}\Delta t_k^3 \\ 0 & 0 & \frac{1}{2}\Delta t_k^3 & t_k^2 \end{bmatrix} * \sigma_v^2 \tag{30}$$

Values obtained from the previous step in the pipeline are used as observations, *i.e.,* location estimates calculated through the modified multilateration method. Therefore, the following observation vector is used:

$$z_k = \begin{bmatrix} x_k \\ y_k \end{bmatrix} \tag{31}$$

Then, the following observation matrix is defined to express that the observation corresponds to the x and y coordinates of the state vector:

$$H_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{32}$$

Finally, the observation noise covariance matrix is defined. Those values used were determined experimentally and work well with sensors used for this setting, while for a different set of sensors, these values might have to be adjusted.

$$R_k = \begin{bmatrix} 0.3 & 0 \\ 0 & 0.3 \end{bmatrix} \tag{33}$$

Using the Kalman filter allows for the improvement of values calculated in the previous step using the information contained in previous values and the knowledge of the system dynamics. It eliminates outliers and smooths these results simultaneously, using plausibility as a determining factor.

4.3.3 LiDAR — LaFlector (2D) and LiCount (3D)

This section details the design of two different prototypes. One based on a 2D LiDAR initially used to range objects within its field of view, and a 3D LiDAR counting with more capabilities to distinguish between objects based on the view depth.

2D LiDAR LaFlector (a wordplay from *laser beam* emitted by the LiDAR and the laser's *reflection*) is designed to reach a flexible deployment and operation of individual hardware components (*i.e.,* sensors, nodes, and sink). It is based on a distributed architecture in which multiple nodes are connected to the respective LiDAR sensors sending data to a sink. The sink is responsible for collecting positioning data and determining the positioning of dynamic (people) and static (environment) objects. Thus, pre-processing steps are performed as early as possible to avoid bottlenecks downstream and reduce the amount of data forwarded by the sink [71]. The following sections show the assumptions defined for its design, detail LaFlector's design, and provide implementation details. LaFlector's code is open-source[1].

[1] LaFlector's source-code: https://gitlab.ifi.uzh.ch/rodrigues/laflector

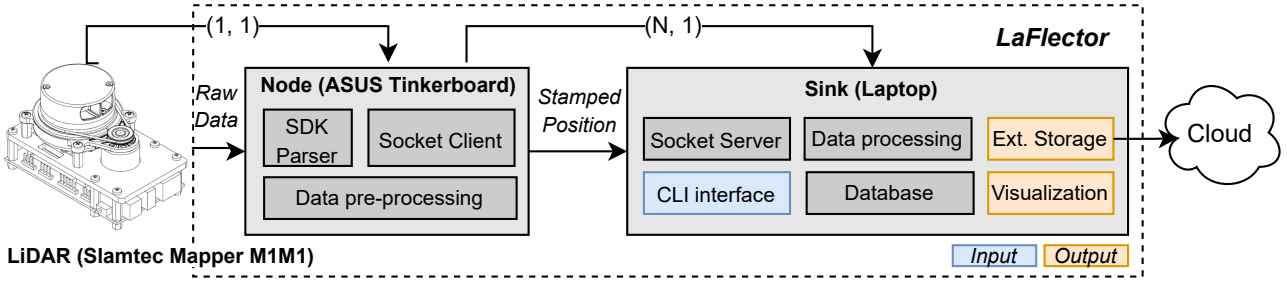

Figure 16: LaFlector's Node-Sink Architecture

Assumptions These are considered for an ideal operation of LaFlector:

- **LiDAR's Placement:** The scanner is to be placed to minimize dead spots, horizontally aligned with the floor.
- **LiDAR's Height:** Object classification expects a solid body with a pre-defined width concerning the measured distance. Thus, the laser must be operated at upper body height. Placing the scanner at foot level would lead to incorrect classifications.
- **Disturbances:** No other devices are continuously transmitting signals at the same wavelength to which the LiDAR scanner could respond to. This applies, for example, to laser pointers or infrared remote controls.

Architectural Components LaFlector's design (*cf.* Figure 16) is based on a distributed architecture consisting of a server acting as a sink and one or multiple nodes. A node is connected to a LiDAR device over WiFi or Ethernet, depending on the node's capabilities. This node-sink design was chosen to ensure extensibility and allows for multiple nodes (*i.e.,* LiDAR and data collection) to be run on one sink (*i.e.,* data processing).

- **LiDAR:** consists of no other visible sub-components and is basically handled as a black box. The Slamtec SDK provides the needed functions to work with the device.
- **Node:** The node's software runs either on traditional X86 or ARM architectures (*e.g.,* System-on-Chip devices, such as ASUS Tinkerboard or Rasperry PI devices) being directly connected via a USB port to the LiDAR sensor. Sub-components include the SDK Parser, Socket Client, and Data Pre-Processing as follows:
  - The *SDK Parser* collects the data from the LiDAR and passes it to the pre-processing.
  - The *Socket Client* connects to the Socket Server of the sink and receives commands.
  - The *Data Pre-processing* converts the distance (r) received and the angle (ϕ) for each data point into timestamped Cartesian coordinates (x, y).
- The **Sink** is the largest component consisting of six sub-components.

- The *Socket Server* receives data from one or more nodes via a TCP connection.
- The *Command Line Interface (CLI)* provides an interface capturing user inputs, forwarding them to the socket server, and controlling the behavior of the Data Processing.
- The *Data Processing* is responsible to segment, classifying and tracking dynamic objects in the line of sight of multiple nodes.
- TRhe *Database* supports the data processing component with a time-series database (*e.g.,* InfluxDB) to store data points delivered by the node.
- The *External (Ext.) Storage* maintains data being exported as a standardized timestamped coordinate to be contrasted or combined with different tracking sources, for instance, to support measurements from wireless tracking (*e.g.,* ASIMOV [75] or BluePIL [76]).
- The *Visualization* provides instant feedback on objects.

To account for the data streaming in the distributed node-sink architecture, data received by the sink work in a retrospective fashion, *i.e.,* work with a local subset of the data that only uses values from the past. Thus, a rolling time window is implemented only containing values from the interval $[t_c - \Delta t, t_c]$, with t_c being the current time and Δt the window size, determined by the update frequency of the nodes and the expected variance of the data (timestamped position).

Merging Upstream Data To compute a location from pre-processed positions, a strategy has to be determined to merge data streams in a rolling time window from different nodes. Bilinear interpolation is used to fit individual upstreams sent by nodes at different cycles with a weighted average of the nearest coordinates. In a rolling time window of Δt s, preconfigured by the sink, nodes accumulate timestamped (x, y) coordinates until t_c is reached and any missing (x, y) coordinate in the rolling time window is estimated by the interpolation at the sink. LaFlector builds upon the assumption that the update cycles of individual nodes are short enough to legitimize the interpolation between two data points as a valid estimation of the true state of the system. To enable the inference of position values at a certain point in time, measured values must be available preceding and succeeding a said point.

Location Algorithm If the same object is detected by multiple nodes, a measurement or a set of measurements should be associated with this object. This association considers the position *i.e.,* placement of LiDARs and timestamped coordinates of tracked objects reported by nodes. However, as LiDARs do not gather information from the tracked objects *i.e.,* persons, the challenge is to model the movement pattern so that the same person is not accounted for multiple times. Thus, it is considered that people follow an almost constant velocity pattern in an indoor environment, in which there are no sudden acceleration movements in contrast to their own average velocity. A nearly constant velocity model is given as follows [39]:

$$x(k) = Fx(k-1) + w(k-1) \tag{34}$$

where F is the state transition matrix of each person in a given time t_c, and $w(k-1)$ is a zero-mean white noise Gaussian process with covariance Q. The white noise is a statistical

model defined in [7] that represents the ”near-constant” velocity variation. F and Q are given as:

$$Fx = I \otimes \begin{bmatrix} 1 & \Delta t_k \\ 0 & 1 \end{bmatrix} \tag{35}$$

and:

$$Q = I \otimes \begin{matrix} \frac{T^3}{3} & \frac{T^3}{2} \\ \frac{T^3}{2} & T \end{matrix} \tag{36}$$

where Q is the power spectral density [39] of the process noise, T is the scan time, $\otimes$ is the Kronecker product and I is the identity matrix. This expresses a belief that the object will have moved in the direction gathered from the last measurement and that the velocity of said movement will not have changed abruptly. Values obtained from the previous step in the pipeline are used as current observations. Therefore, the following observation vector is used: $z_k(\begin{smallmatrix} x_k \\ y_k \end{smallmatrix})$ Then, the following observation matrix is defined to express that the observation corresponds to the x and y coordinates of the state vector: $r_k(\begin{smallmatrix} 0.3 & 0 \\ 0 & 0.3 \end{smallmatrix})$ Those values used were determined experimentally and work for the evaluation scenario in this paper. The following heuristics are applied after data collected by nodes are prepared.

Segmentation of Environment Data segmentation consists of two tasks. While the first task is to detect static objects *i.e.,* environment, the second task compares a different (x, y) in $T+1$ with the static objects to determine moving objects. An implementation detail specific to the Slamtec Mapper M1M1 LiDAR Device [77] is that matching has to be done with every laser rotation, and each rotation contains hundreds of points. Thus, the angle and distance list are written to a Python dictionary that has the advantage of a $\mathcal{O}(1)$ lookup time.

```
1 for angle in scan_dict.keys():
2 temp_angle = angle
3 while temp_angle not in static_dict.keys():
4     temp_angle = temp_angle + 0.001
5     if temp_angle - angle > 0.005:
6         break
7 if temp_angle in static_dict.keys():
8     # do not allow points that lay behind static objects, if this happens,
          something is wrong in the
9     # room and if the distance difference is bigger than a threshold it must
          be a moving object
10    if static_dict.get(temp_angle) > scan_dict.get(angle) and (
11            static_dict.get(temp_angle) - scan_dict.get(angle)) > self.
                  _static_moving_distance:
12        angle_list_moving.append(angle)
13        distance_list_moving.append(scan_dict.get(angle))
```

Listing 1: Comparing Dynamic to Static Objects

This snippet shows the acquisition and construction of the Static Dictionary. This dictionary remains unchanged for the rest of the current measurement. Once the static dictionary is created, detection of moving objects can start. Angle and distance are now retrieved again per rotation and written to the moving objects dictionary. The moving dictionary is compared with the static dictionary and as soon as a threshold (parameter: static-moving-distance) is exceeded, the value is stored in a separate angle and distance list. These values are then checked and classified in the next step.

Object Classification Starts with the two lists *angle_list_moving* and *distance_list_moving*, which contain the segmented measured values of the last three rotations. The first task of classification is to distinguish between multiple objects. To do this, one starts at the first point and compares it with the following one. If the distance between the points is smaller than a certain threshold (parameter: split-distance) it must still be the same object. The checkpoints are shifted and it is checked again. This process is repeated until the object is completely captured. After the objects from the lists are separated, they are identified in the next step.

Object Identification Decide whether a set of points that were previously classified as a human object can be assigned to an already existing object (*i.e.,* same person). The set of points is matched with all objects that are still active. There are three cases that need to be distinguished:

- **No object match**: The points do not correspond to any previously known object. The object must be new accordingly.
- **One object match**: The points can be clearly assigned to an object. In this case, the new position of the object is set.
- **Two or more matches**: In this case, it is not possible to decide which points belong to which object. Assuming that the objects continue to move in the same direction, the expected positioning is determined by means of a direction vector derived from the last way-points.

Output The output is available in two forms. **Plot**. The position of the detected objects and their direction vectors are displayed in a dynamic X/Y coordinate system. This output form serves for visualization. **Logger**. The system has a logger which creates a file with the start time at startup. The verbose level determines which data is written to the log file and which is not. This output is used for debugging. Furthermore, the logger function can be quickly exchanged with a database client, so that the corresponding data ends up in a database instead of the log file. This is in case the data should be further processed or combined with another data source.

3D LiDAR While web user behavior analytics is prevalent, in-store customer behavior analytics is less common due to difficulties in tracking users without disturbing them. With WiFi and LiDAR combined, we can take a passive approach to monitoring user movements within stores. LiCounter combines Yolo and Deep Sort algorithms for detection and tracking, which is evaluated in real locations to collect the data. Further, data is used for classification results by the Gaussian mixture clustering method based on business metrics.

The workflow of LiCounter contains three phases: The first phase is to design a method for localization; the second phase is synchronizing two LiDARs and preparing data for segmentation, and the last phase is data analysis.

1. In phase 1, only one LiDAR is used to collect images in different scenarios, these images are labeled manually in order to train the detector of YOLOV3 and the tracker

of Deep Sort. Virtual machines are set up with InfluxDB installed, which is able to receive data from both LiDARs.

2. Based on the result from phase 1, the detected distance of LiDAR with light is tested for a scenario setup. Two LiDARs are used in a designed mock scenario of shopping, where location, light, and setup would be measured to test the method. Data are collected from the scenario, which is used for segmentation in the next phase.

3. Phase 3 is implemented locally, containing data cleaning, defining metrics for classification, and user segmentation. Finally, a clustered result is returned containing whether the user is interested in an object or not.

Data preparation and collection The Intel RealSense SDK is an open-source and cross-platform library supported by multiple common programming languages [47]. The package *pyrealsense2* is used to access the official Intel RealSense SDK 2.0 in Python. It supports streaming data from the LiDAR sensor and converting the 640*480 objects to a NumPy array of depth data stored as a variable *depth image*. Further, *openCV* library (stands for Open Source Computer Vision Library) converted the depth data into a color map to use existing computer vision algorithms. To prepare a training dataset, data are saved as images of people passing by and different positions at several indoor locations. Example images are shown in Figure 17.

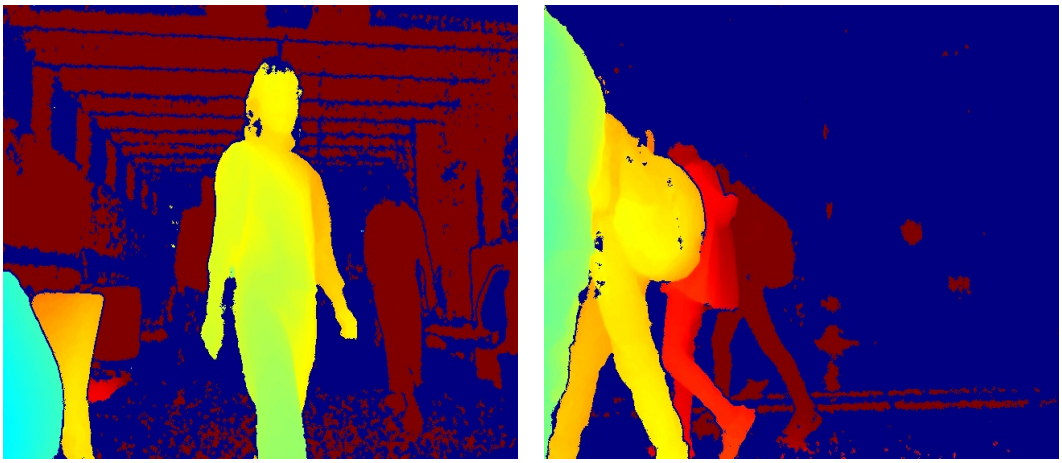

Figure 17: Example of Collected Images of 3D LiDAR (Intel RealSense L515). Lefthand-side Within a Laboratory at UZH. Righthand-side, At The Train Station.

Data is collected from four locations and divided into two categories regarding the light conditions. Different settings are applied to different locations. Figure 18, Figure 19 and Figure 20 show the real setup environment of data collection.

Daytime Condition with natural light.

- **UZH Laboratory**: It is the most convenient location considering the light, but the number of people is limited. Different angles are considered and tested *cf.* Figure 18.

- **Train Station**: An ideal location to have more people and more light influences. LiDAR is settled underground. The farthest distance detected is around 3.5 meters *cf.* Figure 19.
- **UZH/IFI entrance**: The LiDAR is set on a chair on the side of the entrance, facing people when they pass by. The distance is set as 2 meters *cf.* Figure 20.

Night Without natural light.

- **Mock condition at home**: There is no ambient light and a minimal number of people passing by. This scenario allows long-time data collection and more postures.

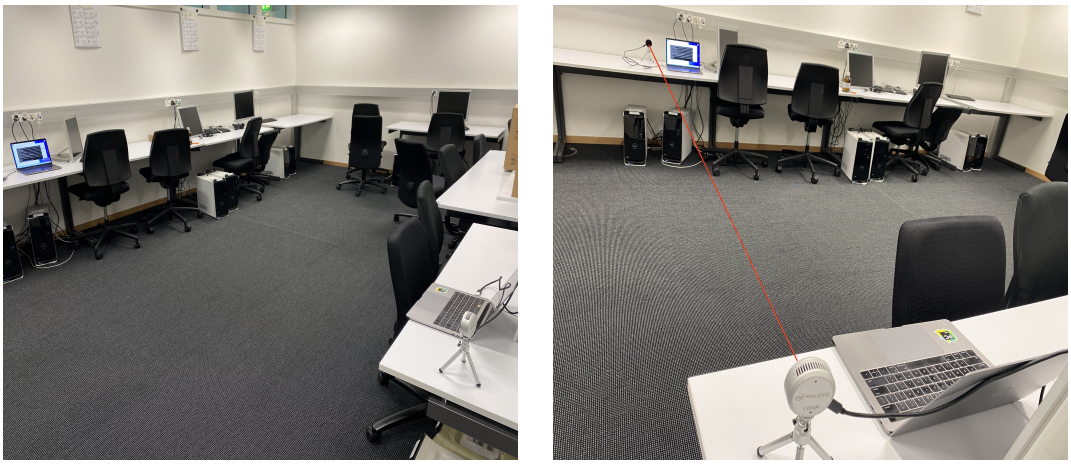

Figure 18: Data Collection Setup at the UZH Laboratory

Figure 19: Data Collection Setup at Train Station

Figure 20: Data Collection Setup at UZH/IFI Entrance

Image Labeling Labeling is an image annotation tool for generating text files with object location information that the YOLO algorithm can use [84]. To label images for customized object detection, in our case, a Person, we manually draw rectangles around the Person as closely as possible and set the class name as *person*, shown in Figure 21. We use the system timestamps as image names to avoid repeated names. The resulting data is a text file with the coordinate, x, and y of each object in the image and the index of the class name.

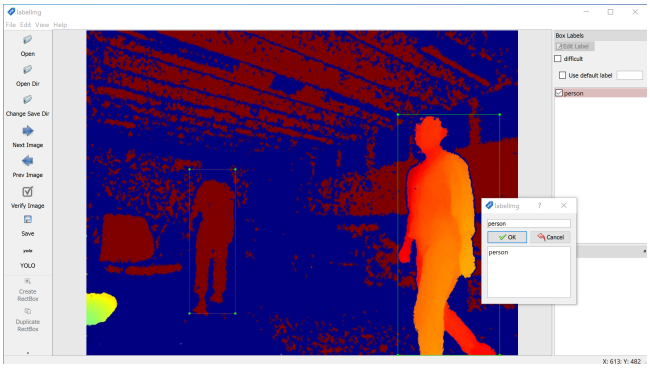

Figure 21: Labeling Images With The Tool *LabelImg* for Customized Object

The color images used in detection are converted from depth data and applied with a colormap, which is different from the normal RGB images. In other words, the images do not show details of the person and the pre-trained detector can not recognize the person in this type of image. Thus, a customized object detector needs to be trained. The image together with the text file generated by *LabelImg* is our dataset for training. Data is split into Train and Test as well. With YOLOv3, a pre-trained Darknet-53 model weight is used for the customized object detector in order to achieve a faster and more accurate training [83].

Data Pre-processing and Cleaning From the queried data of the VM, only one feature needs to be cleaned.Timestamps appear as strings in format *2022-01-19 14:51:28.156420* is convert into datetime object in the format of *%Y-%m-%d %H:%M:%S.%f*. Outliers exist in in-depth measurements. Since the range of detection is only 9 meters in an ideal environment, any depth data above 9 meters (9000 mm) is an outlier. Some measurements only

appear in one timestamp, having a duration of zero. Since a person cannot pass through the scenario setting within one frame, the zero duration data are also considered outliers.

The 3D LiDAR has a Depth field of view $70° \times 55°$ ($\pm 3°$. Camera front glass with -4.5mm and focal length is 1.88mm [47]. L515 camera outputs depth, which is the distance from the object to the image plane, as Figure 22.

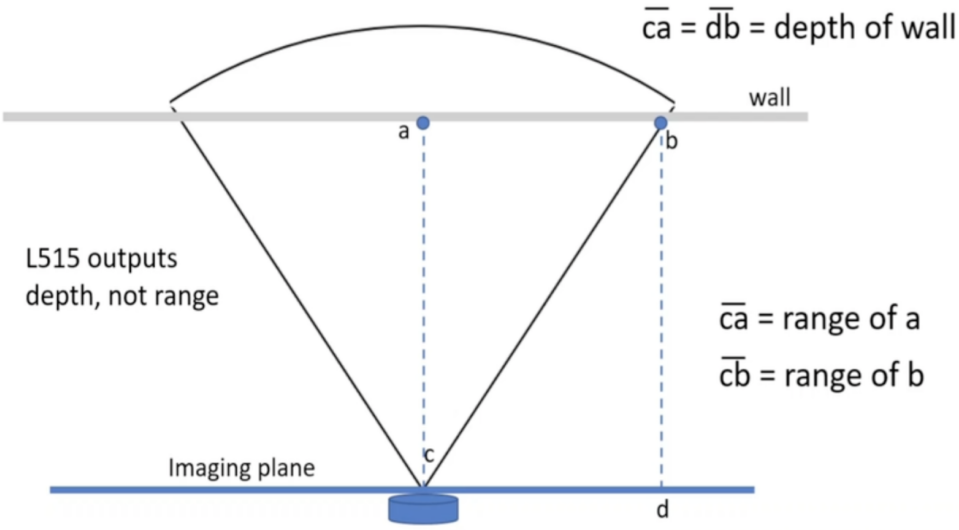

Figure 22: Converting Camera Frame to World Frame [47]

The center of the detected object is recorded as (u, v) coordinates in the Image plane. There is ambiguity in projecting the points from the image frame into the world frame but knowing the focal length and the depth help to disambiguate the points in the world frame. Here are two steps of how to convert from the camera frame to the world frame:

1. **Conversion factor**: Given the field of view of 70 degrees, the focal length of 1.88 mm, and the image plane width of 640 pixels, we can calculate the focal length in pixels. The conversion factor alpha is defined as mm per pixel.

2. **Distance from object center to point of interest**: We already know the depth d from LiDAR, the x from image plane coordinates, and the focal length f in pixels. We convert d to pixels using the conversion factor from step 1. Using similar triangles, we can derive the X which is the horizontal distance from the object center to the point of interest in the top view using $X/u = (d+f)/f$. The X is presented in pixels. We convert it to mm using the conversion factor. Since we place the cameras 2 meters away from each other, we can easily calculate the euclidean distance between the object center and the point of interest as $\sqrt{X^2 + (2-d)^2}$.

Metrics Provide an overview of the performance, and can lead to the generation of actionable insights. We can encourage users' behavioral change by intervening in the system, in order to optimize the metrics. Optimizations can be done manually, automatically, or by combining both. The manual optimization refers to the Hypothesis-Experiment-Evaluation Cycle, which is widely known in internet companies as A/B test, and in academia as controlled

experiments[57]. Below are metrics that we developed for the LiDAR counting application, focusing on measuring user number and user interest by measuring duration and distance.

- **Total user number**. A total number of distinct users appeared in the scene.
- **Accumulated interested user number**. A total number of distinct users that have spent more than 10 seconds within 0.5 meter range of the point of interest.
- **Percentage of interested users**. It is calculated by the Interested users / total users. Measuring how attractive the object is to the group of visited users. This is especially useful when we have multiple points of interest placed in different places that might have different user flow. This metric is not affected by the total user number. For example, if a product is visited by 10 users, 10 interested is considered to be more attractive than an object visited by 100 users but only 15 users are interested.
- **Total time spent on an interesting object**. When the distance between the user and the point of interest is below 0.5 meter and the time spent is more than 3 seconds, the time spent is counted as time spent on the interesting object.
- **Variation of interest**. Variation of user flow calculated on interested users. It is a product-centered metric, which gives insights about which product is more likely to be interested, even more, likely to be bought, and which does not.
- **Variation of user flow**. Divide the entire time into windows of 10 seconds. The number of people that have appeared in each 10-second window is one record. The variation of user flow is the variance of the record. A high total user number and high variation of user flow might be caused by visitor groups. A high total user number and a low variation indicate a stable user flow. This metric gives insights about the venue instead of products. For instance, it can be used to compare two retail stores at different locations, etc.

For example, the *total number of people* is defined as the number of distinct IDs in the frame within a defined time range. To count this value, data are grouped by ID. There are two cases to count.

1. **Number of people at each timestamp**: Since we cap the depth of each camera at 2 meters, there is no overlapping area. The number of people at each timestamp is the sum of counts in both cameras. There are no repetitive counts in this case.
2. **Total number of people**: Repetitive counts can occur in this case. People are assigned a different ID when going out of the frame and coming back. This also applies to people going from one camera field to the other one. One person would be assigned two IDs when crossing from one camera to the other camera field, and be counted as two people.

To solve the problem of counting twice a single person, we merge the IDs by replacing the two IDs of the same person who is detected by both cameras with one new ID. This can be done because whenever a person is crossing the border, it is within the detection range of both cameras, and is assigned two different IDs. A person can be assigned a third ID when

going outside the range of one camera and coming back, and a fourth ID when the same happens for the other camera, and so on and so forth. Therefore, we need to keep track of all the IDs that represent the same person and merge them as one ID to avoid repetitive count.

We do this by analyzing neighboring points according to timestamp. We identify the samples as should be merged when they satisfy all three criteria 1) happen at the same time: the time difference between the neighboring points below 1 second. 2) at the border: the sum of depth from both cameras between 16000 to 18000mm. 3)taken by two cameras: one ID with the '_1' flag and one without. We build a data frame called 'merge_ID' that collects all the IDs and their next IDs which should be merged. Then we build a graph that has all the IDs as nodes and the neighboring relation as edges. All the IDs that belong to a connected graph are the same person that has been detected multiple times. In our original data frame, we replace all the IDs that belong to the same graph with a new ID starting with 'm_' followed by an index number of that graph.

4.3.4 RFID and Camera — CCount

The application scenario considers 3D cameras and RFID readers mounted on the ceiling. Individuals within an exhibition wear or hold a low-cost UHF RFID tag in the form of a badge. As individuals walk within the view of the 3D camera and RFID reader, RFID tags are continuously read and the 3D Camera tracks the participant's position. In such a scenario, the CCount must correctly match data originating from 3D cameras with the data collected by RFID readers to uniquely count people in the scene (*cf.* Figure 23).

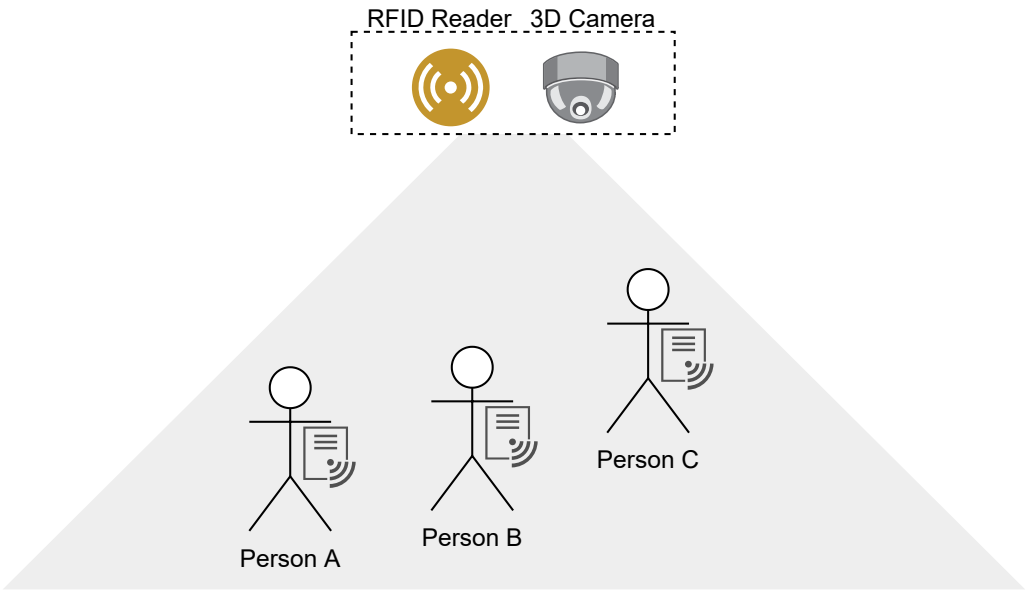

Figure 23: Illustration of the Application Scenario

The challenge is identifying RFID tags belonging to IDs assigned by the 3D camera and is, thus, associated with an individual. If a person who left the scene returns later, CCount has to recognize and prevent it from being counted twice. To accomplish this, a method for correlating the RFID ID of the 3D cameras with the Electronic Product Code (EPC) of the

RFID tags is required. For that, a frontend is developed to handle the interaction with the user, encompassing the selection of the desired camera, the associated event date, and a graphical representation of the dashboard associated with the selected event. The data is then requested from the backend, which is hosted on Amazon Web Services (AWS) and based on lambda functions (*cf.* Figure 24). It is built based on a serverless architecture built upon microservices to scale components while maintaining extensibility dynamically.

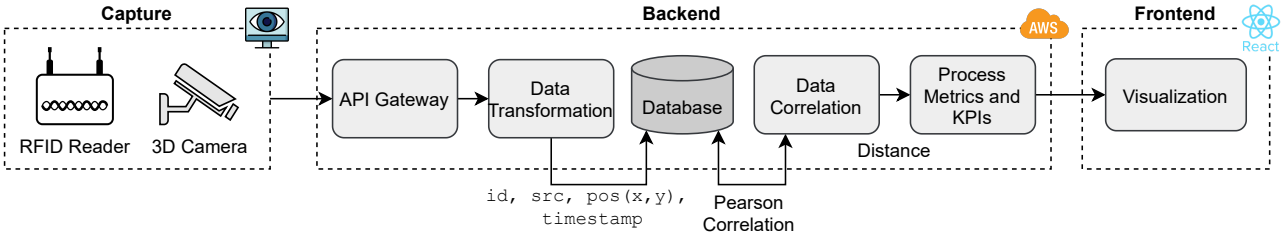

Figure 24: CCount Workflow

- The **API Gateway** collects and aggregates data coming from 3D cameras and RFID readers in time windows.
- **Data Transformation**: Data received at the API is typically encoded in different standards (*e.g.,* Cameras use Protobuf and RFID readers use JavaScript Object Notation, JSON), and in this sense, the data transformation standardizes them for database storage.
- The **Data Correlation** retrieves data in pre-defined time windows from the database and uses the Pearson correlation to determine, whether there exists statistical evidence for a linear relationship among variable pairs, such as coordinates.
- **Process Metrics and KPIs (Key Performance Indicators)** handle the localization and filtering of data and minimize errors by removing outliers. Once localization data is processed, metrics, such as the unique count of individuals, dwell times, or visit duration, are calculated.
- The **Visualization** provides a graphical frontend, where it is possible to visualize metrics and KPIs in near real-time, and a REST API, where data can be retrieved.

Cameras Implementation Details A camera sends data to the API Gateway, which forwards it to the Data transformation module. This module implements a lambda function that comprises a list of objects, each containing the following attributes: *timestamp*, *coordinates*, and *numberOfPeople*. The next step involves the preparation of the retrieved coordinates for the actual heatmap creation process. Figure 25 shows an example of a heatmap during the testing stage in the laboratory cafeteria (3D Camera deployed in the ceiling).

Heatmaps were dynamically generated based on the predefined time windows. The idea is to iterate over elements in the coordinates array and uniquely count the occurrences of every *(x,y)* pair with the timestamp laying within the slider values. This way, it is possible to ensure that every time the slider is adjusted, the data is filtered and the heatmap is updated accordingly.

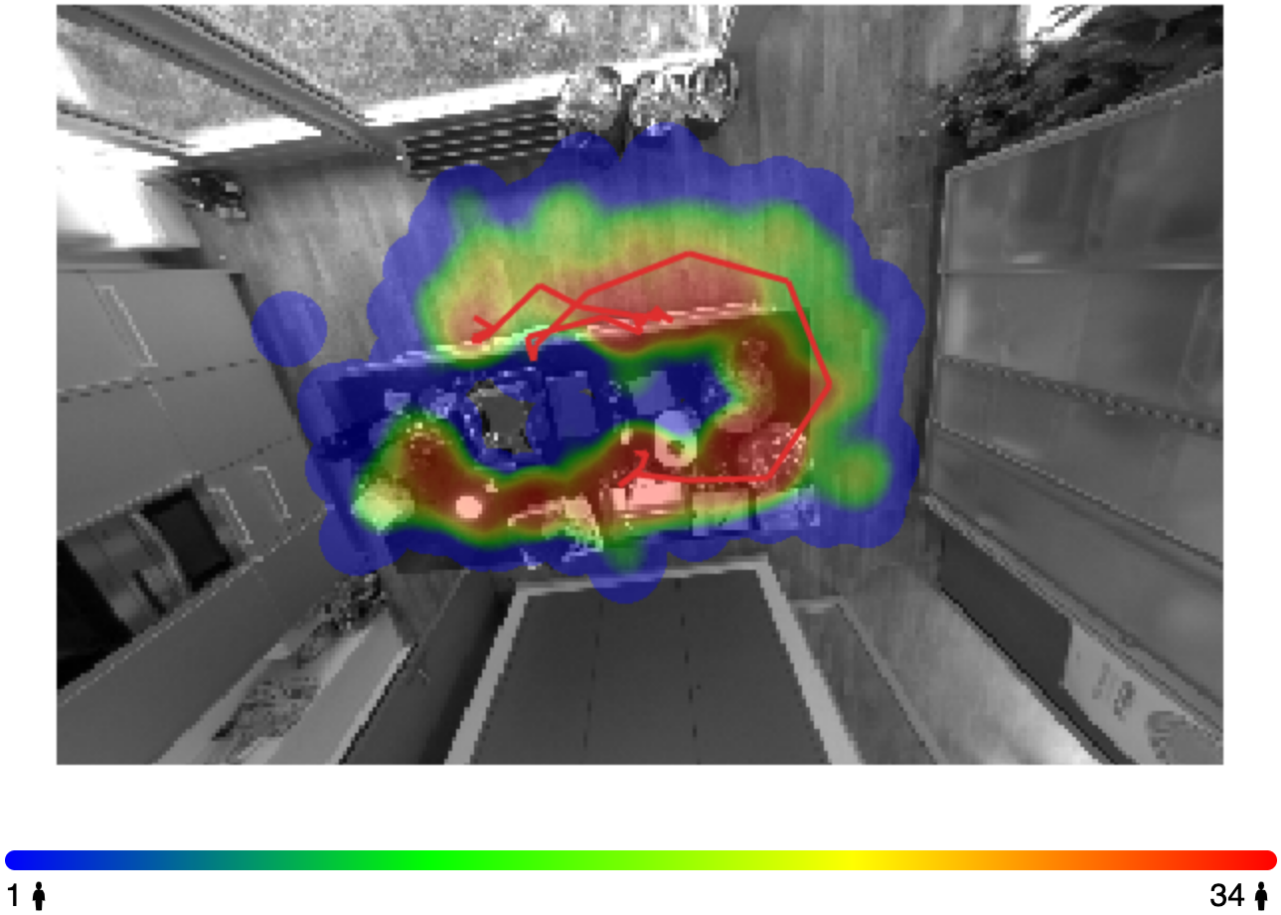

Figure 25: Adjusting Tracking and Heatmap Based on Camera Data

Data Transformation Before the correlation and metrics, as well as KPIs, are applied, measurement data needs to be transformed into the correct format. Three steps are necessary to correlate data: *(1)* transform relative into absolute positions and *(2)* ensure that the time difference (Δt) is within a pre-determined offset for a given time window. Lastly, *(3)* initial filtering of data and erroneous readings from RFID and cameras is performed.

1. Relative and Absolute Positioning For *(1)* different sensors operate on coordinate systems that differ (*i.e.,* are relative) from the absolute position in the monitoring environment. The necessary input is the internal mapping of the environment in which the position of each sensor (here cameras and RFID readers) and its field of view (*cf.* Figure 26) are determined. For example, cameras have a limited field of view relative to the RFID reader and are typically installed at critical points, such as entrances, exits, or checkout lines, where individuals tend to crowd. Conversely, RFID readers have a more extended range for tag reading, coupled with reduced accuracy, especially at crowded points.

2. Clock Synchronization An essential aspect of distributed systems is the need to ensure that different nodes are synchronized, and in the context of using different sensors, the

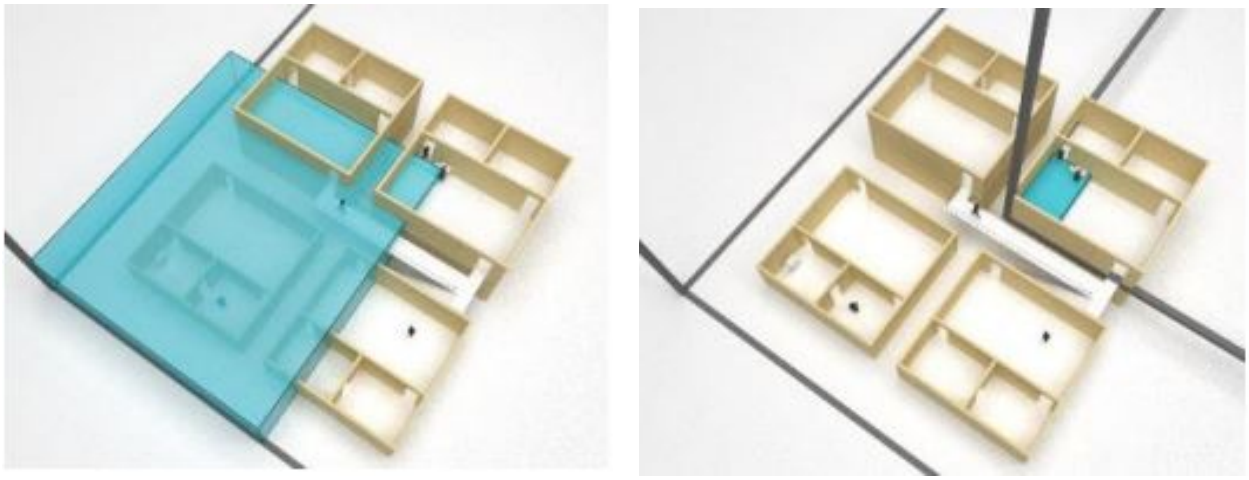

Figure 26: Absolute (Left) vs. Relative (Right) Positioning [79]

problem applies. At this step *(2)* RFID reader and cameras feature a built-in clock that keeps track of the date and time information (*i.e.,* received data is time stamped from its source). Since the clock is subject to drift, it may need to be synchronized periodically via the Network Time Protocol (NTP) or manually adjusted. Also, when receiving data in a time window, duplicates with the same ID and time stamp, but in different locations, are eliminated.

3. Data Filtering The Kalman filter in step *(3)* is used to smooth RFID readings and allows for the increased accuracy of values estimated in the previous time window. It requires two central equations that require specification and parameterization to fit a particular context. The first equation makes a forward projection about the state s based on the previous state, an (optional) control input and an error term [51, 93], $s_k = As_{k-1} + Bu_{k-1} + w_{k-1}$.

The matrix A relates the state s at the previous time window $(k-1)$ to the current time window (k). Matrix B relates the optional control input to the state s. w determines the process noise, assuming that it is Gaussian noise with a mean of 0 and variance of Q, $p(w) \sim N(0, Q)$. The second equation relates the state in time k to the measurement in time window k with a measurement error $z_k = Hs_k + v_k$. Therefore, the matrix H is called the transition matrix. To specify the matrices A and B correctly, it is important to recall the underlying process. The positions x and y in time window k received from RFID data rely on positions of x and y in the previous time window $k-1$ as well as on velocities $\dot{x}$ and $\dot{y}$ at time $k-1$. It is possible to take into account accelerations $\ddot{x}$ and $\ddot{y}$, too.

The model defined assumes a constant velocity for individuals, *i.e.,* the velocity is constant between time windows (Δt) and the accelerations $\ddot{x}$ and $\ddot{y}$ are both 0. While this reads as a simplifying assumption, it is justifiable in this situation, since the time windows are relatively small (1 s). Moreover, variations in speed (*i.e.,* deceleration and acceleration) are considered in the process' noise covariance matrix Q. Therefore, a control matrix B is not specified. The transition matrix is obtained as follows:

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The Kalman filter also requires parameterizing R and Q. While it is possible to optimize the transition or covariance matrix, instead of providing them, underlying algorithms for this are computationally expensive and not suitable in the context of near real-time applications. One of the parameters needed is the matrix of measurement noises R.

$$R = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}$$

This matrix contains measurement noises for x and y and their covariance. The covariance is assumed to be zero, so it is needed to obtain the variance of x and y. The RFID manufacturer states that in 66% of cases the real measurement will be within 1 m and in 85% of cases the real measurement will be within 1.5 m [44]. Assuming that the normal distribution holds for these errors, it is possible to fit a normal distribution to the values provided. By adjusting the distribution shape (*i.e.,* the parameter σ^2, since μ is assumed to be 0), the area under their curve is equal to 0.66 (with z values of -0.5 and 0.5). Thus, it is possible to state that 85% of these observations are within 1.5 m, yielding a standard deviation of around 0.52 m. Further, it is assumed that measurement errors for x and y are independent (*i.e.,* σ_{xy} = 0).

Figure 27 displays the values measured (blue) versus the values filtered (orange). It is visible that large swings are smoothed out (*e.g.,* for x in the interval [0:200]) unless there is a sustainable movement in position (*e.g.,* the one that occurs for x in the interval [200:400]). This is more apparent when looking at measures of y, since there are a few upward swings, all of which are smoothed out by the Kalman filter. While the unfiltered values are relatively more variable, the filtered values show a smaller variation at the beginning of each interval. The scenario used in this example is considered a static tag in the time interval [600:1000].

Data Correlation While cameras provide the actual position of individuals detected, they are not able to distinguish between individuals. The camera assigns a new ID to every individual detected in the frame, but different IDs may refer to the same individual leading to inaccurate measurements. The Pearson correlation [9] was used to determine statistical evidence for a linear relationship among variable pairs, such as time-stamped coordinates originating from the camera and the RFID reader. The Pearson coefficient $\mathbf{R}$ in Equation 37 should be at least 0.5 with a tolerance of 1 second between time stamps, which means that for each pair of coordinates $\forall(x, y) \in CAM$ of cameras, and each pair of coordinates $\forall(x', y') \in RFID$ of RFID tags, should have at least 50% of correlation (*i.e.,* moderate) with a max difference of one second between their time stamps. Considering that both RFID and Camera are synchronized, correlation R-values less than 50% and within the time window are discarded. A common result considering possible distortions in the RFID signal especially when multiple tags are grouped during a peak time, for instance.

$$\mathbf{R} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2(y_i - \overline{y})^2}} \tag{37}$$

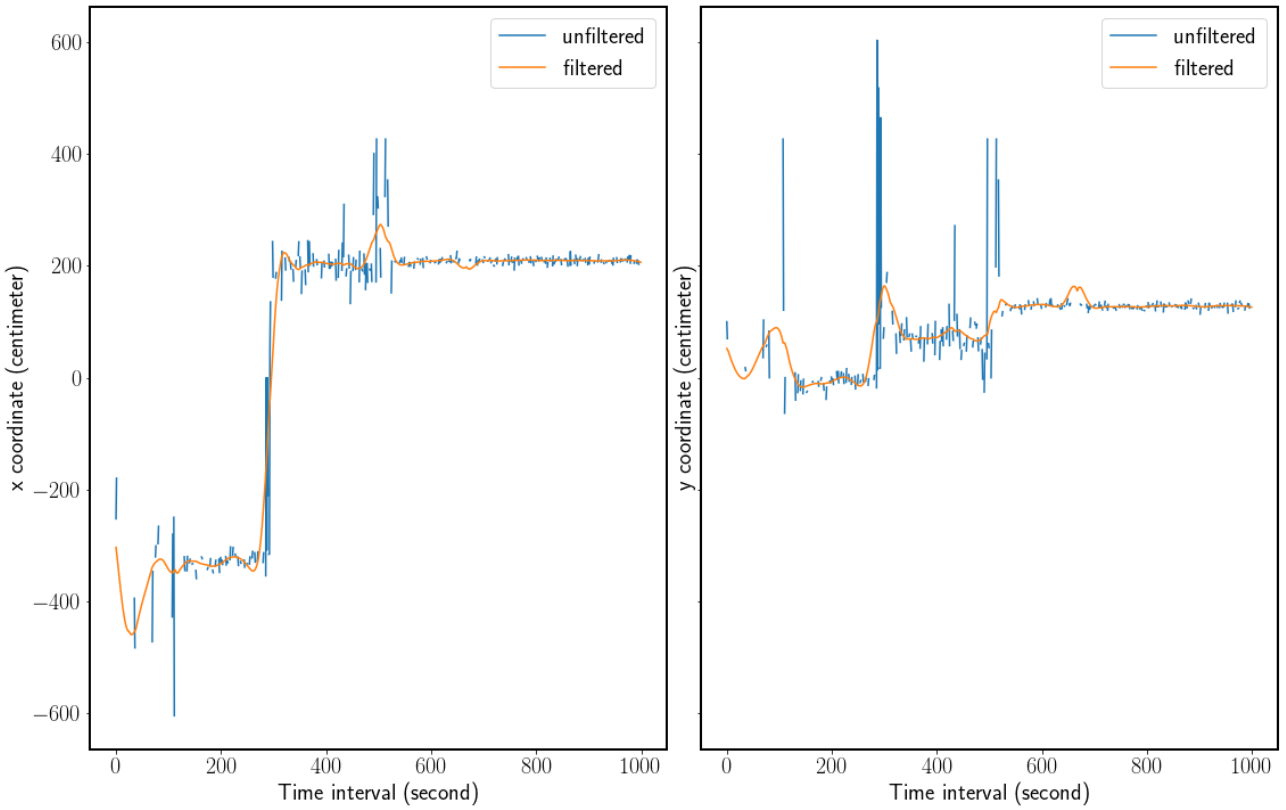

Figure 27: Comparison of Unfiltered and (Kalman) Filtered Values for x and y Coordinates with Static RFID Tags

In order to normalize the data for RFID readers, boundaries of the 3D camera (*i.e.,* line-of-sight) need to be manually mapped into the RFID reader's coordinate system. Whenever a change is made to the configuration of the RFID reader (*e.g.,* changing the antenna power or moving the reader), and these values must be reassessed. In addition, the matching algorithm has to read values for each RFID reader and apply the normalization. Furthermore, to smooth the location data and improve accuracy a filter is required. In this regard, a Kalman filter is one of the most suitable ones for near real-time applications: it does not require a large memory of past observations, is recursive, and is relatively fast [11]. Conversely, a moving average would require a certain number of rolling past observations to be kept in memory for smoothing.

Process Metrics and KPIs In the context of real-world applications (*e.g.,* shopping, sports events, and concerts), it is often helpful to create aggregated views of data to understand, for example, the overall trend and make it accessible for a more comprehensive (non-technical) audience. Thus, these data can be used for different purposes, such as to inform policy decisions or evaluate the effectiveness of marketing measures. The scenario of a trade fair (or other exhibitions), for example, is valid for a merchant or Point-of-Reference (PoR) to understand how visitors behave around a booth. In this context, three pieces of information seem of particular interest for practical RFID tracking applications: *(1)* How many of the visitor's spot a particular booth? *(2)* How many of the ones, who spot the booth, move closer

to see what it is about? *(3)* How many of the ones that inform themselves about the booth move on to interact with the PoR? In an ideal setting, KPIs are easy to understand and compare, which is often achieved by using a ratio:

$$\textit{Visibility} = \frac{\text{Number of Views}}{\text{Number of Opportunities}}$$

$$\textit{Engagement Rate} = \frac{\text{Number of Visits}}{\text{Number of Views}}$$

$$\textit{Interaction Rate} = \frac{\text{Number of Interactions}}{\text{Number of Visits}}$$

In order to understand the number of visitors in a place and its behaviors, it is possible to use the concept of *Views* (or impressions). This concept is defined as a count within the views zone of interest, describing the number of times the visit zone was viewed. In a similar direction, the *dwell time* consists of a minimum and maximum time interval (in seconds), which is different for each zone of interest. In this context, the dwell time is used to associate targets to specific areas of interest. Indeed, if a target is within the dwell time range and its location is inside a particular zone of interest, it is classified and assigned to that zone. There are three zones (one for interaction, one for visitors, and one for view) with others designated as opportunities. Given that *(i)* the reader can read 6 m in each direction, *(ii)* the location accuracy is approximately at 50 cm; *(iii)* based on simulating multiple PoRs, *(iv)* KPIs should be sufficiently different for later visualization. While the zone radius was defined in 100 cm steps, the interaction zone shows a radius of 100 cm, the visiting zone has a radius of 200 cm, and the view zone has a radius of 300 cm.

This definition enables a straightforward interpretation (and comparability) in practice since these values are percentages and, thus, range between 0 and 100 because any count in the interaction zone is also a count in other zones. Another consideration is comparability. While the above definition enables the reader to understand the number quickly, it does not tell how the number compares. One can base the assessment on how good a particular value is, which is necessarily between 0 and 100. This, however, is not practical, since it is not known if an interaction rate of 100% is practically achievable. Thus, two benchmarks are used. The first benchmark is an in-group benchmark, comparing the KPI of a particular period to the minimum and maximum of that KPI during a specified time range. The second benchmark does the same, but instead of using an in-group comparison, it uses an out-of-group comparison.

Lastly, to determine whether an individual is within any of the relevant zones, the distance between each observation and PoR is calculated by the Euclidean distance, where p denotes the observation of a tag consisting of x and y coordinates and M denotes the PoR position with x and y coordinates: $d(p, M) = \sqrt{(p_x - M_x)^2 + (p_y - M_y)^2}$. Afterward, a binary variable is created for each PoR's zone. These variables equal 1, if an observation falls within a defined zone, otherwise 0. This ensures that an observation can be simultaneously in multiple zones.

Visualization CCount extensively uses tables and data visualization techniques, such as heat maps, bar charts, and line charts, which rely on data that may be subject to change over time and needs to be shared among different components. The heat map generation

process is triggered as soon as the frontend receives the necessary data, in this case, person coordinates, from the backend. Heat maps and other KPIs (*e.g.,* number of unique

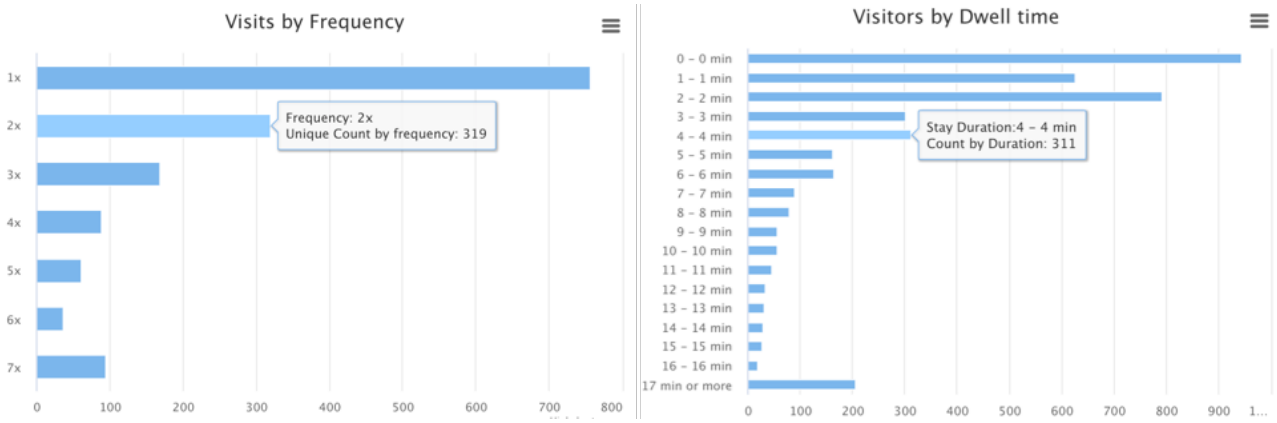

Figure 28: Visualization of Visitors and Dwell Time

visitors and dwell times as depicted in Figure 28) are dynamically generated based on the predefined time windows. This allows for the iteration over elements in coordinates array and uniquely counts occurrences of every *(x,y)* pair with the time stamp laying within the slider values. This way, it is possible to ensure that the slider is adjusted, data is filtered, and graphical elements are updated accordingly.

4.3.5 Synthetic Data Generation

Once the standard data structure is defined and implemented, as shown in the previous section, PasWITS can start data pre-processing to transform received information according to the defined structure. Thus, the use of APIs ensures that data can be received directly from the sensors or from the synthetic data generator for the purpose of rapid scenario experimentation and prototype evaluation.

In addition, since the fusion of different sensor data to generate coherent movement patterns of individuals is not commonly utilized in practice, datasets for such (indoor) sensor measurements are restricted (*i.e.,* no open datasets are available). However, [23] made their dataset of movement data in a real-world setting available. This data was generated by using three 3D stereo vision cameras in a public building over 13 days. The authors collected approximately 5.5 million measurement points and identified approximately 1,200 individuals taken in the summer of 2019.

Similarly to the authors' purpose to use this dataset for benchmarking and the analysis of occupant behavior and trajectory patterns, the dataset is used in this work here as the ground truth dataset from which synthetic measurements are generated and which is used to evaluate predictions of the PasWITS engine. Measurements can be emulated synthetically by applying transformations to readily available real-world movement datasets (*e.g.,* movement of people inside a trade fair over several days) and artificially placed virtual sensors in a virtual location.

Utilizing these data publicly available, the emulation of real-world sensor measurements can be performed by a synthetic generation of expected sensor measurements, according to the following transformations (equations 38 to 44 and Figure 12). The goal of this synthetic

measurement generation is to replicate real-world measurement data produced by a client utilizing the API. The process of synthetic measurement generation, described below, is repeated for every combination of every position in the real-world dataset ($p_{abs,i,t}$) with every sensor (s).

The left part of Figure 12 shows the true position of the sensor (blue) and an individual *i* at time *t* (gray). The center shows the true position of *i* in relative coordinates (p_{rel}) and the randomized, synthetically generated measured position ($p_{syn,s}$), based on the spatial precision of the sensor (gray dotted circle). The right indicates two randomly generated measurements of the sensor, where one ($p2_{syn,s}$) would be rejected due to insufficient measurement reach (blue dotted circle) of the sensor.

Relative Positioning By rotation and translation, with respect to the sensor's (s) position (p_{sensor}) in a globally defined frame of reference, the absolute position ($p_{abs,i,t}$) can be determined of a real-world position of an individual i at time t [25, 54]. Concerning the aforementioned frame of reference, it can be converted into a relative position ($p_{rel,i,t,s}$) regarding a new frame of reference using the sensor as the new coordinate system's origin. The sensor's position (p_{sensor}) is static (*i.e.,* regarding its placement) over time and described by the combination of three coordinates (x_s, y_s, z_s) with three rotational parameters ($\alpha_s, \beta_s, \gamma_s$) all defined in relation to the origin of a global frame of reference.

$$p_{rel,i,t,s} = T_s(x_s, y_s, z_s) + R_s(\alpha_s, \beta_s, \gamma_s)\, p_{abs,i,t} \tag{38}$$

where $R_s(\alpha_s, \beta_s, \gamma_s)$ and $T_s(x_s, y_s, z_s)$ are the sensor's translation (T_s) and rotation (R_s) matrices. The relative ($p_{rel,i,t,s}$) and absolute positions ($p_{abs,i,t}$) are 1×3 matrices comprised of the (x,y,z)-coordinates of i at time t.

Spatial, Randomized Measurement Error Additionally, the spatial measurement error of the sensor (μ_s), *i.e.,* the random introduction of spatial measurement errors in the real world due to imperfect sensors, is taken into account and modeled, resulting in the synthetic position without applying any spatial or temporal cutoffs ($\tilde{p}_{syn,i,t,s}$).

$$\tilde{p}_{syn,i,t,s} = p_{rel,i,t,s} + \mu_s \tag{39}$$

where μ_s is based on the spatial precision of the sensor (ρ_s), but randomized in the (x,y,z)-coordinates. As the precision only gives the upper-bound of the (absolute) spatial error, it needs to be ensured that the simulated error distribution follows a continuous uniform distribution, which would be expected in reality, *i.e.,* assuming errors without bias concerning distance. Therefore, μ_s takes the following form:

$$\mu_s = \begin{bmatrix} x_{randomized} \\ y_{randomized} \\ z_{randomized} \end{bmatrix} \text{ where} \tag{40}$$

$$|x_{randomized}|, |y_{randomized}|, |z_{randomized}| \leq \rho_s \tag{41}$$

A rejection sampling method ensures that the simulated spatial errors follow the previously described continuous uniform distribution over a sphere (3D) or circle (2D). That also provides significant performance benefits over other methods [49, 92]. Figure 29 shows this process by using the true position of the center of the sphere and then deriving simulated

measurements based on it, such that each of them is located inside the previously defined sphere.

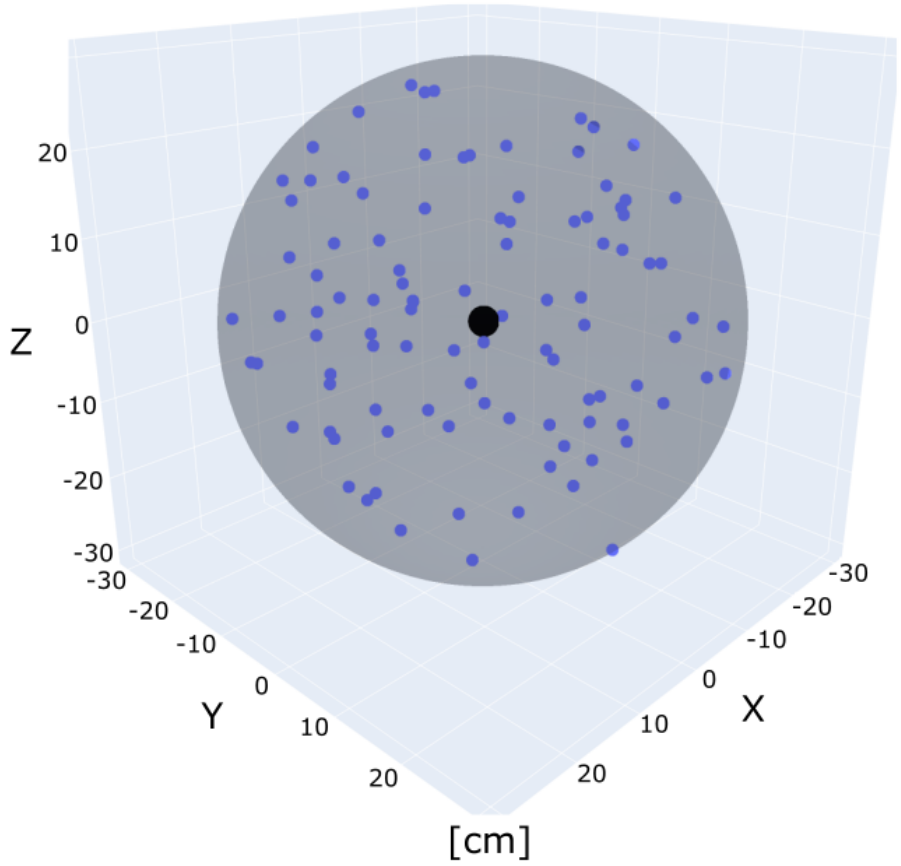

Figure 29: Exemplary Rejection Sampling Outcome for n = 100 with ρ_s = 30 cm. X, Y, Z Axis in [cm]. ● True Position, ● Simulated Measurement, ● Sphere Boundary

Figure (29) shows the result of an exemplary rejection sampling as implemented by [49, 92] using the true position as the center of the sphere with an applied spatial precision of 30 cm. Here, a simulation of 100 measurements of this exemplary sensor leads to no points being outside the sphere defined, which demonstrates that all randomized points outside the defined error range were rejected.

Spatial and Temporal Cut-off Functions Subsequently, to model the limited measurement reach of a real-world sensor (λ_s), the spatial cut-off function (Λ_s) is introduced. Applying cut-off functions to the intermediate synthetic position $\tilde{p}_{syn,i,t,s}$ ensures that no synthetic measurements are reported that lie outside the reach of the specific sensor. Similarly, a temporal cut-off function (Γ_s) is introduced in order to ensure that no two measurements time points ($t_{x,i,s}$ and $t_{x+1,i,s}$) are reported, where the time between them is smaller than the temporal resolution of the sensor (γ_s).

$$p_{syn,i,t,s} = \Gamma_s \, \Lambda_s \, \tilde{p}_{syn,i,t,s} \tag{42}$$

where $\mathbf{\Lambda_s}$ and $\mathbf{\Gamma_s}$ take the following form, respectively.

$$\Lambda_s = \begin{cases} \text{if: distance}(i,s) \leq \lambda_s \text{ then } \Lambda_s = 1 \\ \text{else: synthetic measurement is dropped} \end{cases} \tag{43}$$

$$\Gamma_s = \begin{cases} \text{if: } (t_{x,i,s} - t_{x+1,i,s}) \leq \gamma_s = \text{ then } \Lambda_s = 1 \\ \text{else: synthetic measurement dropped} \end{cases} \tag{44}$$

Object Identifiers and Randomization Lastly, each sensor measurement of each object has an attached object identifier. To ensure that identifiers are randomized and cannot be linked to the original individual's ID, the identifier for measurement can be randomized. Here, potentially different variants of randomization are considered to replicate different real-world behaviors.

- **Static, non-randomized identifier**: For each object, every measurement uses the individual's true ID. A real-world application where *e.g.,* the participants of a trade fair receive an RFID ticket and use a Wi-Fi-based app linked to their name.
- **Static, randomized identifier**: For each object, a constant identifier that is different from the true individuals' ID is used. For example, a real-world application where participants of a trade fair receive an entry ticket badge and use a Wi-Fi-based app, which is not linked to their name.
- **Object-based, randomized identifier**: An object reports an ID that is randomized but constant over measurements, regardless of the sensor. A real-world application of this would be a Wi-Fi device reporting a unique MAC (Media Access Control) address to the sensor.
- **Sensor-object-based, randomized identifier**: An object reports an ID that is randomized for each sensor trying to measure that object. A real-world application of this would be a Wi-Fi device reporting a unique MAC address to the sensor and utilizing MAC address randomization to obfuscate its path.

4.4 Fuslon Data Tracking System (FITS)

This PasWITS architecture is specified, its data processing pipeline discussed, and the respective API defined.

4.4.1 Architecture

Figure 30 summarizes the FITS architecture further describing the processing flow. At first, the pre-processing and analytics components are highlighted, which are supported by a time series database used to store the continuous streaming of data, and the core database holding configurations of locations, sensors, and measurements. Further, the data processing pipeline describes the exchange of data between these components. Lastly, the evaluator metrics describe the prediction model used to cluster identified objects observed from different sensors.

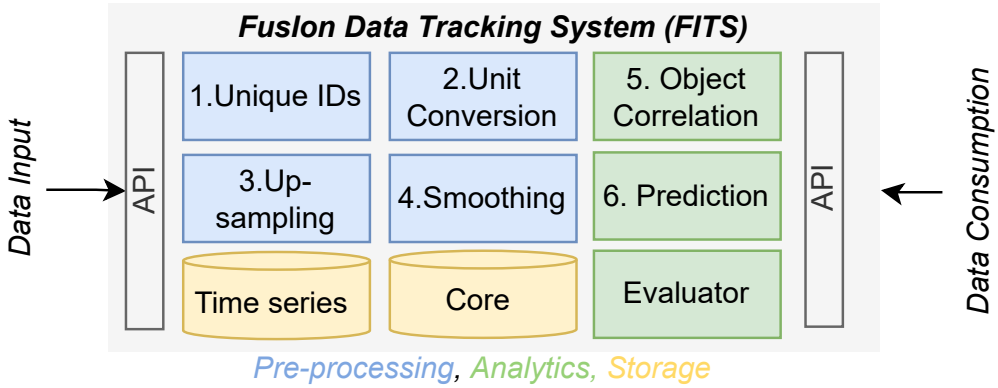

Figure 30: FITS' Pre-processing, Analytics, and Storage, to Data Consumption

1. Unique ID: ensure that each visitor captured is uniquely identifiable across the entire measurement dataset. For example, picture two camera sensors that assign numerical identifiers to objects in their reading range. Combining measurements of the two sensors makes it impossible to distinguish between the objects recorded. In the case of wireless sensors (*e.g.,* Wi-Fi or Bluetooth), this is not a concern, since objects are usually identified by their unique Medium Access Control (MAC) address. However, since FITS does not make any domain-specific assumptions, generic handling of object identifiers of any sensors is essential. Since every sensor that recorded the measurement has a unique identifier, concatenating the object ID and sensor ID makes the measurement identifiable across the entire dataset.
2. Unit conversion: converts measurements from sensors into a standard format. As specified in the location's sensor configuration, sensors have an origin position, and orientation, and provide their measurements in a specific unit (*e.g.,* "cm"). By calculating the position offset, rotation angle, and multiplication factor, data from different sensors is harmonized and translated into the location's coordinate system.
3. Up-sampling: ensures that the resulting position timeline is not sparse and contains a position prediction for each time step—even when no measurement was recorded for that point in time. Up-sampling has two purposes: *(i)* To compensate for missing data and *(ii)* to interpolate the position when the time interval is below one second. An interval of 1 s is sufficient to model the real-time movement of visitors. However, as this assumption might change, this parameter could be made a configurable variable in the location object. Eventually, up-sampling is done by first rounding the timestamped data to 1 s. Then, the data is grouped by an object identifier, and for each missing second a new row is inserted, as well as to impute the X, Y, and Z coordinates. A Kalman filter is used as a kinematic process that assumes visitors are moving at constant speeds [94]. Then, it fits on the sparse measurement data and is used to impute the X, Y, and Z coordinates of the up-sampled, newly created rows.
4. Smoothing: Since sensors, based on the Received Signal Strength (RSS), are prone to signal interference, the generated signal contains considerable noise. This step smooths measurement data to remove the noise captured within the data. Again, an object identifier groups the data, and the Kalman filter is used as a kinematic process that assumes visitors

are moving at constant speeds and is applied to the *X*, *Y*, and *Z* coordinates to reduce data noise.

5. Object Correlation: combines as the core idea measurements from different sensors for improved tracking accuracy. To leverage this concept and combine measurements, firstly measurements and sensors are identified, which originate from the same experiment by building clusters of unique object IDs received on each measurement. An ID identifies a cluster (*i.e.,* cluster ID) and contains the list of object IDs belonging together. Thus, each cluster maps back to a unique visitor in the location.

As previously mentioned, spatial and temporal dimensions are critical to group unique objects into a cluster. For example, grouping objects considering a one-hour difference does not lead to meaningful clusters, since the underlying visitor probably has moved in the meantime. To this end, time windows are identified in the dataset by sorting data frames by timestamps and filtering unique object IDs to build the respective clusters. Python™ dictionaries store which objects a particular sensor has captured and, similarly, by which sensor a specific object was recorded. Additionally, it stores possible object ID combinations across all sensors to build up a mapping structure that contains a cluster ID and a list of object IDs best fitting together. At this stage, these dictionaries are iterated to calculate the pairwise and cumulative Euclidean distance (*loss*) to determine the fit of the current cluster.

An important aspect is the calculation of the loss that is being used to select best-fitting clusters. The next code block illustrates the implementation of a *cumulative_norm* function that is used to calculate a loss based on the Euclidean distance. The function receives a list of vectors (*cf.* line 1), corresponding to the object cluster under evaluation. Each vector represents a measurement with *X, Y, Z* coordinates belonging to the object ID part of the cluster that is currently being evaluated. In the trivial case of only one vector, the loss of *0*, is returned.

```
1     def __cumulative_norm(self, vectors):
2         cumulative_norm = 0.
3         if len(vectors) == 1:
4             return cumulative_norm
5         index = [[i] for i in range(len(vectors))]
6         index_combinations = list(itertools.product(*index, repeat=2))[0]
7         for i in range(0, len(index_combinations) - 1, 2):
8             a = index_combinations[i]
9             b = index_combinations[i+1]
10            cumulative_norm += np.linalg.norm(vectors[a] - vectors[b])
11        return cumulative_norm
```

Listing 2: Cumulative Norm

Otherwise, all possible 2-tuples of vectors are calculated and iterated over (*cf.* line 7-9). For each 2-tuple, consisting of a vector *a* and a vector *b*, the Euclidean distance is calculated and summed up (*cf.* line 10). As the Euclidean distance is symmetric, the parameter order of the function is irrelevant. Eventually, the cumulative norm of the list of vectors is returned (*cf.* line 11). Subsequently, best fitting clusters (*i.e.,* lower loss) are selected, a unique cluster ID is assigned, and dictionary mappings are updated by inserting the new cluster ID as key and a list of clustered object IDs as value. Finally, these measurements are re-indexed with their appropriate cluster ID.

6. Prediction of Position: approximates the object's position with those clusters computed by combining all sensor measurements for a specific cluster. Measurements are grouped by

Table 3: Relevant CLEAR-MOT Metrics According to [10, 42]

Metric	Description
MOTA	Multiple object tracker accuracy
MOTP	Multiple object tracker precision
GT	Total number of unique object IDs encountered (relevant for the context of MT, PT and ML)
MT	Number of objects tracked for at least 80% of lifespan
PT	Number of objects tracked between 20% and 80% of lifespan
ML	Number of objects tracked less than 20% of lifespan

cluster ID and timestamp. A weighted average of these different sensor measurements is calculated to predict the *X*, *Y*, and *Z* coordinates at a specific point in time.

Data Storage: While the correlation and prediction of position steps are performed entirely in memory, data persistence is needed to support the pre-processing correlation process. Two databases are required, *cf.* Figure 30: a core database holding static configurations of measurements or settings and a time series database to handle the data upstreaming. The core database is used *(i)* to facilitate CRUD operations (Create, Read, Update, and Delete) on locations and sensor configurations, *(ii)* to store meta-information on the job status, *(iii)* to store the calculated object identifier clusters.

While the requirements for *(i)* to *(iii)* are typical CRUD operations performed by REST APIs, they can be handled easily by any relational database system. However, a time series database is used to store the predicted location timeline. Data can potentially become very large, depending on the number of sensors and periods of measurement. For example, considering a period of 10 h (*e.g.,* an event starting from 6 am and running to 6 pm), with an up-sampling rate of 1 s and 100 visitors, yields a total of 3.6 million rows. Thus, efficient read and write operations become critical, when storing or fetching this amount of data.

Evaluator Metrics: The prediction model is analyzed using those metrics, as specified in Table 3, to ensure complete coverage of the problem's sub-tasks and to provide a holistic view. To generate the mapping between true objects and predicted objects, mapping dictionaries of prediction object IDs and true IDs are created based on the "object_identifier_clusters" provided by the API response and the true dataset.

The evaluation pipeline starts with naïve metrics of clustering, where a contingency matrix is constructed. Afterward, the assignment of predicted and true objects is performed based on the highest overlap. Next, the *bcubed* measures recall, precision and *fscore* are estimated using those previously mentioned mapping dictionaries between true object IDs (also called occupant IDs in the dataset) and those prediction object IDs provided by the API response. These conclude the clustering evaluation methods.

Next, the prediction evaluation is performed using the Euclidean distances between previously matched predicted and true objects for each timestamp using the contingency matrix method. The total sum of error, the mean Euclidean distance, the minimum, and maximum distance, and the median distance, in cm, are calculated. The most meaningful measures evaluated later on include the mean and median. The total sum needs to be analyzed carefully since the number of measurement points directly impacts it. Finally, the naïve Euclidean distance metrics are also provided as visualizations, based on different splits.

CLEAR-MOT metrics not only provide classical analogon to accuracy (MOTA) and precision (MOTP), but also allow for the identification of how many objects were either mostly tracked (more than 80% correct), partially tracked (between 80% and 20% correctly tracked), and mostly lost (more than 20% correctly tracked), and the number of false positives and misses. To estimate the CLEAR-MOT metrics, individual frames are constructed for each timestamp consisting of these objects predicted, the true objects, and their positional coordinates for the specific timestamp. These frames are aggregated, and the evaluation algorithms provided by [42] are used to calculate relevant metrics. Since MOTA is sensitive to overfitting the model and the API provides up-sampled results, CLEAR-MOT metrics are calculated once without controlling for up-sampling and once with controlling for up-sampling (*i.e.,* removing objects predicted from frames where no expected true objects exist).

4.4.2 Data Processing Pipeline

FITS' workflow (data pipeline) consists of five stages as outlined below and follows the general design assumptions determined above:
1. Sensor Configuration Data input specifies the virtual measurement setup of the synthetic measurement generation. This includes the specification of sensor positions (x_s, y_s, z_s) & orientations $(\alpha_s, \beta_s, \gamma_s)$ as coordinate systems relative to the absolute frame of a reference as well as the definition of sensor parameters described in subsection 3, namely sensor spatial reach (λ_s), temporal resolution (γ_s), and the sensor's spatial precision (μ_s). Additionally, the specification of a sensor type is necessary to be able to analyze the performance using this parameter. Optionally, variables, such as sensor identifier format (default: *UUID4*) and stability functions (default: linear), can be defined.
2. Data Generation The synthetic data generation utilizes a call to the Synthetic Data Generator module to generate the emulation dataset based on sensors provided above. Additionally, it can take the preferred number of measurement points of the real-world dataset that should be analyzed for this sample. Furthermore, the identifier randomization method, including the identifier type, takes place. And lastly, it is possible to reduce the provided 3D dataset into a 2D dataset by setting the necessary flag.
3. Data Streaming All API calls, according to the details above, are performed, and the resulting output is stored temporarily. First, the API endpoint is called based on the test data input (either local or via a public Internet Protocol address, if deployed in the cloud). The pipeline starts with the construction of the API payload in the standard data structure format. This payload is pushed to the API to generate a new location. The response is temporarily stored, and the synthetic data received in the stage before is pushed to the API using the location parameters received from the API call beforehand. The data input batch is transferred to the API, and a response is queried every second to ensure that timeouts are caught. Lastly, the response, if received, from the API is stored for further processing, evaluation, and visualization as well as optionally persisted via an exportable file.
4. Data Pre-processing, Analytics, and Storage The core stage is responsible for preprocessing incoming data and storing them for further analysis. Those steps were detailed in Architectural Components and further steps on the evaluator metrics are detailed below.
5. Visualization The front-end module gathers all API responses as they would be sent to a real-world client, and enriches these, such that visualizations of predictions are made available persistently, if necessary. Therefore, the module first transforms the *JSON* response

of the API into a *.csv* file for the user, if requested. Afterward, the predicted movement of individuals is generated as an interactive graphic in 3D and 2D.

4.4.3 API Overview

The REST API is the public-facing interface for clients to interact with the system. It exposes the functionality over the HTTP protocol and uses JSON as its data format. In order to manage different locations and their respective setup of sensors, the `POST /locations` endpoint is provided to support CRUD operations.


```
{
    "id": 1,
    "name": "BIN-2-A.10",
    "external_identifier": "BIN-2-A.10",
    "sensors": [
        {
            "identifier": "d418a88b-bc41-476a-93e1-b0669b9295b9",
            "type": "RFID",
            "x_origin": 6,
            "y_origin": -2,
            "z_origin": 4,
            "yaw": 0,
            "pitch": -2,
            "roll": 4,
            "measurement_unit": "cm"
        },
        {
            "identifier": "08ea942b-9998-4604-a0de-7e516861a01f",
            "type": "Wi-Fi 2.4GHz",
            "x_origin": 0,
            "y_origin": -1,
            "z_origin": 1,
            "yaw": 2,
            "pitch": -1,
            "roll": 1,
            "measurement_unit": "m"
        },
        {
            "identifier": "9af28219-8191-4ad3-9f33-f356688c6cc5",
            "type": "camera",
            "x_origin": 6,
            "y_origin": -2,
            "z_origin": 4,
            "yaw": 0,
            "pitch": -2,
            "roll": 4,
            "measurement_unit": "cm"
        }
    ]
}
```


Once a location with a sensor configuration was setup, the client can post the collected measurement data to the `POST /locations/{location_id}/inputs` endpoint to create a new input batch job.


```
[
    {
        "object_identifier": "37957",
        "sensor_identifier": "d418a88b-bc41-476a-93e1-b0669b9295b9",
        "x": 264.0,
        "y": 255.0,
        "z": -12.0,
        "timestamp": 1559376000000
    },
    ...
]
```

Upon receipt, the system starts to process the measurement data. As this task is computationally expensive ($O(n^2)$) and depends on the input size, its computation is done asynchronously in a separate thread to keep the client from blocking. In response, the job and its processing status is returned.

```
{
    "job_id": 26,
    "location_id": 28,
    "status": "scheduled",
    "created_at": "2021-12-17 08:49:37"
}
```

By polling `GET /locations/{location_id}/inputs/{job_id}` the client will be updated on the job status and upon completion, the status is changed to `finished` and the client can fetch the output of the computation - the location predictions.

```
{
    "job_id": 26,
    "location_id": 28,
    "status": "finished",
    "created_at": "2021-12-17 08:49:37"
}
```

To this end, the client fetches the results at
`GET /locations/{location_id}/inputs/{job_id}/outputs`.
Results contain the `object_identifier_mappings` as well as the `positions` - a timeline with position estimates for the identified objects.


```
    "job_id": 26,
    "location_id": 28,
    "object_identifier_clusters": {
        "6c7b7f07-53a7-4259-8ef9-71573ad88be1": [
            "39202___1df21a15-6ca9-4d52-91d0-e389cd7742e3",
            "39202___caafac72-433d-47ef-a41f-5c39b2ea1495",
            "39202___bace279a-80b8-4ef7-80ec-13714e5c4e9d"
        ],
        "57402ef7-35d4-4d7c-82bd-bb31e8923368": [
            "39222___caafac72-433d-47ef-a41f-5c39b2ea1495",
            "39222___1df21a15-6ca9-4d52-91d0-e389cd7742e3"
        ],
        "d54ba674-3f35-4cb2-bf59-166453be7edf": [
            "37957___1df21a15-6ca9-4d52-91d0-e389cd7742e3"
        ]
    },
    "positions": [
        {
            "object_identifier": "57402ef7-35d4-4d7c-82bd-bb31e8923368",
            "timestamp": 1559376000000,
            "x": 145.01,
            "y": 270.8,
            "z": 12.36
        },
        ...
    ]
}
```


4.5 Data Consumption

Data consumption refers to the different ways to present information gathered by FITS or directly via sensors to users. In this regard, the design of a front-end followed a modular architecture that harmonically relates a group of five stakeholders with three different subsystems, such as a graphical UI, and a microservice-based backend.

4.5.1 Stakeholders

In order to have a better overview of all stakeholders involved in the system, Figure 31 shows an Entity Relationship (ER) Diagram, *i.e.,* a graphical representation of all entities and their relationships to each other in the platform. Within the system, there coexist five main entities, *i.e.,* users, which have different roles, agencies, clients, campaigns, and activities.

The platform supports different authentication and authorization layers, which implies users with different roles are able to perform different actions. Additionally, new users need to be invited into the platform in order to be able to interact with other stakeholders. At the top of the roles hierarchy, there are root accounts. These accounts are the most privileged in the system and have the strongest authorization and security level. Indeed, they are the only accounts that are consciously authorized to have a global overview of all other stakeholders and perform all possible facets of system administration. Since there are no security restrictions imposed upon these accounts, they are able to perform all actions which can be performed by lower roles. These permission-based layers are known as the cascade effects. Finally, root accounts are the only ones that are able to create new agencies and can actively manage them.

In turn, agencies are per definition entities that contain and relate to multiple clients (brands). Generally, agencies are external marketing companies that conduct large mar-

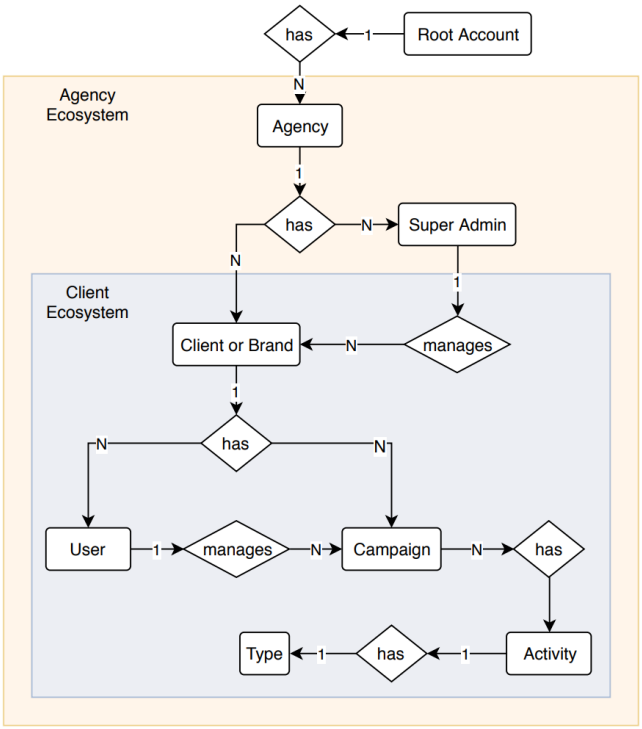

Figure 31: Stakeholders ER Diagram

keting projects and represent multiple brands simultaneously. In the proposed architecture, agencies are managed by super admins, which are responsible to assign new clients and invite new users to their specific agency's ecosystem. Similar to root accounts, super admins can create new entities in their environment. However, they can not open new agencies or assign clients to an agency they are not responsible for.

Clients (or brands) can have multiple marketing campaigns and cannot exist without being assigned to an agency. For those clients which are not assigned to a third-party agency, a default agency will be assigned to them. In case of the existence of the same client in multiple agencies, this will be treated as two different entities.

While a super admin is responsible for a whole agency, similarly admins are in charge of managing a subset of clients that have been assigned to them by a super admin. Indeed, the latter defines the client's ecosystem for a client admin. Following the authorization cascade principle, client admins can create new entities in their environment, however, they cannot open new clients or assign new users to a client they are not responsible for.

Thereafter, project managers cannot invite new users to join the platform but are responsible to create and manage client campaigns. In contrast, analysts can neither create campaigns nor invite new users and have almost everywhere within the platform read-only rights. Nevertheless, they are responsible for monitoring different marketing campaigns and ensuring a successful outcome of all promotional activities.

Table 4 summarizes the overall permissions-based authorization concept, which states that different users with different roles perform different actions. It should be noted, that both super admins and admins can create new users with a set of conditions. In fact, both are able to invite new users within the platform only with an equal or lower role equal than theirs.

For instance, a super admin is not allowed to create a new root account.

Table 4: User- and Role-based executable Actions within the Platform.

User role — *Can create*	*Agencies*	*Clients*	*Campaigns*	*Activities*	*new Users*
Root	✓	✓	✓	✓	✓
Super Admin	×	✓	✓	✓	(✓)
Admin	×	×	✓	✓	(✓)
Project Manager	×	×	✓	✓	×
Analyst	×	×	×	✓	×

4.5.2 Zones of Interest

In this regard, the common pattern of all possible activity types (by stakeholders) is a product that wants to be advertised in a certain location for a specific customer target. Therefore, an activity type is always associated to a physical place where the product gets demonstrated, distributed, or tested. This always implies a group of people in motion who consciously or involuntarily come into contact with the product from different angles within the product area. Because of this, in order to state whether the interaction with the product has been voluntary or unintentional for a set of walkers, the surface is divided into five different circular surfaces as shown by Figure 32.

Each circular surface can be viewed as a zone of interest, whose radius is depending on the available area during the product demonstration. The approach consists of geometrically distributing a predetermined amount of sensors over the entire area. The more sensors have been placed on the surface, the more precise will be the measurements.

Indeed, the algorithm with which a certain person is assigned to a certain area of interest is based on the signal strength of the perceived Wi-Fi signal and dwell time algorithm. The dwell time consists of a minimum and maximum time interval (in seconds) and can be configured independently of the zone. This interval, together with the signal strength, is then used to associate targets to specific areas of interest. Indeed, if the MAC address of a target is counted with a frequency that results to be within the dwell time interval, this target is assigned to the zone which corresponds to the perceived signal strength.

As shown in Figure 32, five different zones of interest have been categorized.

1. **Opportunities:** (or Walkers-by, Passers-by, Footfall) Count within the maximum range of the sensor(s). Describes the profile of a location in terms of total foot traffic. For the assignation, it takes into account the number of MAC addresses counted within the total pickup range of the sensors.

2. **Views:** (or Impressions) Count within the views zone of interest. Describes the number of times the visiting zone was viewed. The calculation relies on the number of MAC

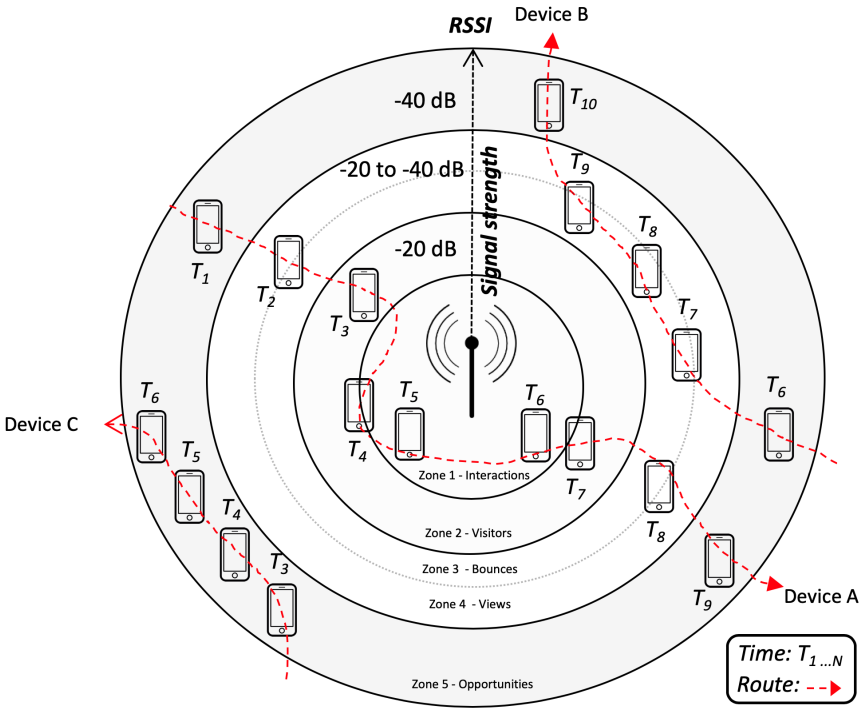

Figure 32: Dimensions: Zone of Interests

addresses counted within a defined signal strength and minimum/maximum dwell time range for the views zone of interest.

3. **Visitors:** (or Sessions) Metrics for the visitor's zone of interest. It describes the number of times the visiting zone was visited. The calculation takes into account the number of MAC addresses counted within a defined signal strength and minimum/maximum dwell time range for the visitor's zone of interest.

4. **Interactions:** (or Goals, Targets, Leads) Count of actions that indicates a positive lift for the brand or product, or an immediate impact on financial goals. The calculation takes into account the number of MAC addresses counted within a defined signal strength and minimum/maximum dwell time range for the interactions zone of interest.

5. **Bounces:** Count within the visitor's zone of interest that leaves the zone before converting to a visitor. The calculation takes into account the number of MAC addresses counted within the defined signal strength of the visiting zone with a dwell time *smaller* than the required minimum dwell time.

4.5.3 Parameters

Location A location is a dimension that defines a group of available data collection locations for metrics and, by extension, performance indicators. Currently, all metrics are available for individual activities and follow the hierarchy below:

$$Campaign \rightarrow Phase \rightarrow Place \rightarrow Location \rightarrow Activity\ (\rightarrow Zone\ of\ Interest)$$

Date-time Date-time is a dimension that is defined as a date and time for the collection of metrics and has the following hierarchy:

$$Year \rightarrow Quarter \rightarrow Month \rightarrow Week \rightarrow Day \rightarrow Hour \rightarrow Minute$$

Min / Max Dwell Time Minimum and Maximum dwell time are parameters that can be viewed as zone boundaries for collecting metrics in a zone by defining an upper and lower limit for visit duration (range in seconds). For instance, if the maximum dwell for the zone of interest ”interactions” would have been configured to 10 minutes, a person who would have stayed more than 10 minutes in that area would not be relevant for the analytics. In fact, the latter might be someone working for the promotional team. The Min/Max dwell time is a required parameter for the configuration of duration zones of interest.

Visit GAP Visit GAP is a parameter that expresses the length of time in seconds after which a visit ends if a signal is not reacquired. For example, if the visit gap is 5 minutes and a visitor returns within 5 minutes after leaving, that counts as one visit. This parameter is required for the configuration of zones. Defaults are 5 minutes.

Duration Period The duration period is a parameter that defines the minimum size in seconds for calculating the distribution of visitor duration. It is a required parameter for the configuration of duration and is correlated with the minimum dwell time for the visiting zone.

4.5.4 Metrics

Count Count (or counter) is a metric that expresses the number of persons counted within a given time period. It is computed with the total amount of filtered MAC addresses counted within a given time period. In the platform, the count is always equal to or larger than the unique count and is available for all zones and in quarter-hour, hourly, daily, and weekly time frames. Furthermore, it can be aggregated to any time period from hourly data.

Unique Count Per definition, the unique count is a metric that can be viewed as the number of unique persons, expressed with the total number of unique MAC addresses within a time frame counted within a given time period. At present, the unique count cannot be aggregated, *i.e.,* the sum of hourly unique counts per day is greater than the daily unique count. Furthermore, it is available for all zones and in quarter-hour, hourly, daily, and weekly time frames.

Average Dwell Time Average Dwell Time is a metric that expresses the average visit length distributed over a time period. This metric can be viewed as the average of last seen - first seen for the count. Per definition, a visit ends when a MAC address has not been seen

for a sufficiently long period of time, *i.e.,* a visit GAP. Average Dwell Time is available for all zones and in quarter-hour, hourly, daily, and weekly time frames.

Frequency Per definition, frequency is a metric that counts how often visitors return. For instance, a frequency of one is the number of visitors that visit once while a frequency of two is the number of visitors that visit twice, and so on. This metric is available on a weekly, monthly, and quarterly basis.

Duration This metric is a count of visits per duration, *i.e.,* the distribution of visitor durations. For instance, a count of 74 for a duration of 300 seconds indicates 74 visits for 5 minutes. It is based on dwell time per visit and duration setting and is available on a weekly, daily, and hourly basis.

Key Performance Indicators (KPIs) In the context of this thesis, KPIs are numerical values that demonstrate how effectively a marketing campaign is performing and achieving its key objectives. Clients (or brands) use KPIs at multiple levels to evaluate the outcome of their marketing investments in terms of reaching targets. High-level KPIs include overall factors, such as *Visibility-, Engagement-, Interaction-, Bounce-* and *Return-Rate*, *Loyality* and *Engagement Reliability*. Table 5 illustrates the equations expressing these KPIs.

Table 5: KPIs Formulas.

KPI	*Variable Name*	*Formula*
Visibility (or Capture Rate)	V	$\frac{Views\ Count}{Opportunities\ Count}$
Engagement Rate	ERa	$\frac{Visits\ Count}{Views\ Count}$
Interaction Rate	IR	$\frac{Interactions\ Count}{Visitors\ Count}$
Bounce Rate	BR	$\frac{Bounces\ Count}{Views\ Count}$
Loyalty	L	$\frac{Count}{Count\ Unique}$
Return Rate	RR	$\frac{Count - Count\ Unique}{Count}$
Engagement Reliability	ERe	$\sigma(daily\ ERa)$

- **Visibility** is a KPI that expresses the percentage of opportunities that saw the visit zone. This KPI describes the visibility of a an activity.

- **Engagement Rate** is a KPI that is expressed as the percentage of views that are converted to visits.
- **Interaction Rate** is a KPI expressed as the percentage of visitors that stayed enough in the ”interaction” zone to be considered interactors.
- **Bounce Rate** is a KPI that can be viewed as the percentage of views that entered the visits zone but left before converting to a visit.
- **Loyalty** is a KPÎ which is expressed as the average number of visits by a visitor. For instance, a loyalty of 1.9 indicates that, on average, people visited the activity about twice. This KPI will vary depending on the time period and is implemented for visits zone of interest. For example, weekly loyalty can usually be considered to be higher than hourly loyalty.
- **Return Rate** is a KPI that is used to calculate the percentage of visitors that visit the activity more than once. Typically, the return rate will vary depending on the time period, *i.e.,* weekly return rate is usually considered to be higher than the hourly return rate.
- **Engagement Reliability** is a KPI that indicates the level of variability in daily performance and provides a measure for different levels of performance (*e.g.,* excellent or poor performances). This KPI is defined as the standard deviation of the daily engagement rate.

4.5.5 Microservice Architecture

Traditional application models used to work with monolithic architectures, whose processes were tightly coupled, wrapped, and exposed into a single API. With a microservices architecture (also known as *serverless*), the system can be built with independent components that run each application process as a service. Indeed, these services are loosely coupled, well-defined, encapsulated, independent from each other, and exposed as a single container or API endpoint. Differently from a monolithic point of view, each service can be viewed as a reusable capability that needs to be as granular and abstract as possible.

Figure 33 illustrates the design for the serverless architecture adopted in PasWITS. The overall idea consists of having one microservice which is in charge of opening SQL connections, releasing database instances, and sending queries to the database itself. This means, that if other micro functions need to perform data manipulations, they can invoke the microservice that is in charge of managing the database directly. This has the advantage that a single application manages connection pools. Therefore, if there would be a change in the way the database is queried or accessed, only one microservice would need to be adjusted.

On one hand, this compromise does not introduce an anti-pattern nor contradicts the fundamentals of a microservice-based architecture. In fact, single services still have their own business logic and are well-encapsulated. On the other hand, the same database can be queried and navigated through different collections by using multiple instances. This allows easy communication across different services and reduces the complexity and negotiation effort to resolve a simple request.

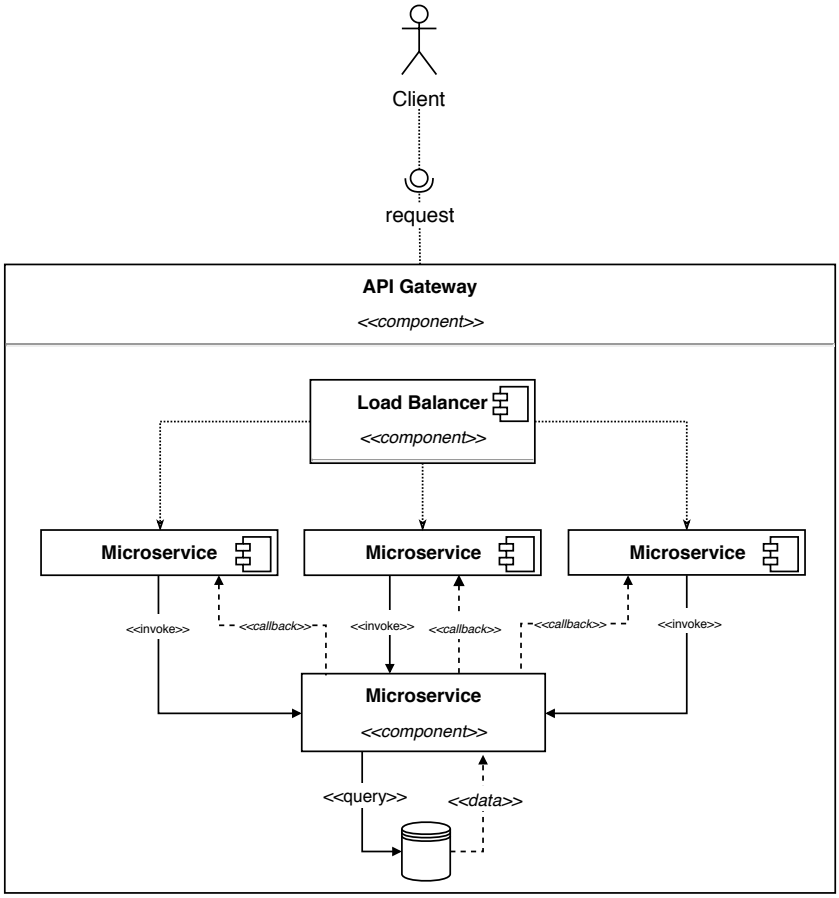

Figure 33: Serverless Architecture and Shared Database.

4.5.6 User Experience

In order to follow state-of-the-art UX patterns, a design evaluation based on a clickable prototype has been performed. The platform needs to have an intuitive navigation concept. This implies, that the user must be able to navigate through different states and stakeholders by following an unambiguous path. For this reason, a cascade structure has been adopted. This means that all stakeholders are connected through a navigable route. For instance, a client (or brand) might have a list of running campaigns, which might have in turn a series of promotional activities.

On one hand and as illustrated in Figure 34 clients, their campaigns and activities are connected through different cards. Indeed, in this overview, a client can immediately see its last running campaign and all other campaigns. Similarly, its last running activity is shown together with all other activities. The corresponding detail views are displayed by clicking on a specific campaign or activity. This high-level and presorted overview allows each client to see its last running items in their ecosystem. It is worth highlighting, that the last running activity must not necessarily be part of the last active campaign.

On other hand, as shown in Figure 35, the campaign level shows a list of running campaigns. Each card represents a campaign, it displays its overall score and contains two links with the next respectively last performed activity. Campaigns can be sorted by name, status, score, and last modified. Consistently to the navigation concept, campaigns and activities

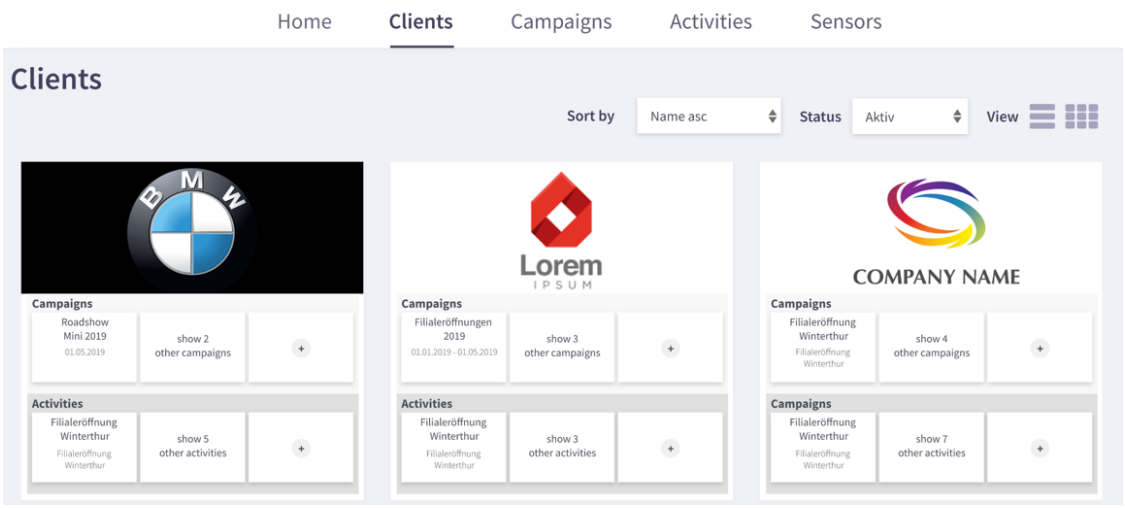

Figure 34: Clients Concept

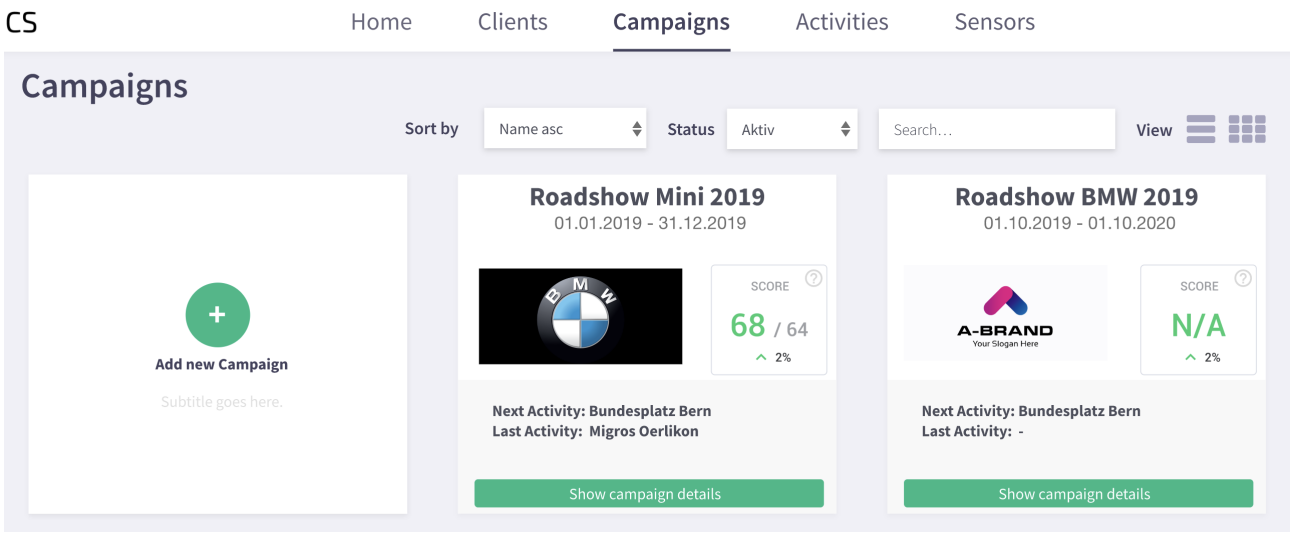

Figure 35: Campaigns Concept

are linked by an unambiguous path. Indeed, by clicking on a certain activity (either at the campaigns or clients level), the activity overview tab is displayed.

As illustrated in Figure 36, the activity tab contains the list of KPIs presented. These indicators are used to measure using numerical and tangible values the effectiveness of offline marketing investments and determine with a weighted average the overall score of the activity. The navigation concept is highlighted again in the top left corner of Figure 36. Indeed, it shows a nested navigation menu for navigating through the previous stakeholders by inverting the flow of the cascade effect.

Data visualization using multiple charts represents the key elements of the dashboard. Indeed, the presentation of the collected data in a pictorial or graphical format is at the base of every analytical decision. Charts are necessary to identify patterns, give interactive visualization and visualize temporal and spatial dimensions in a single and compact format. In this regard, a list of relevant charts has been identified considering a set of metrics, dimen-

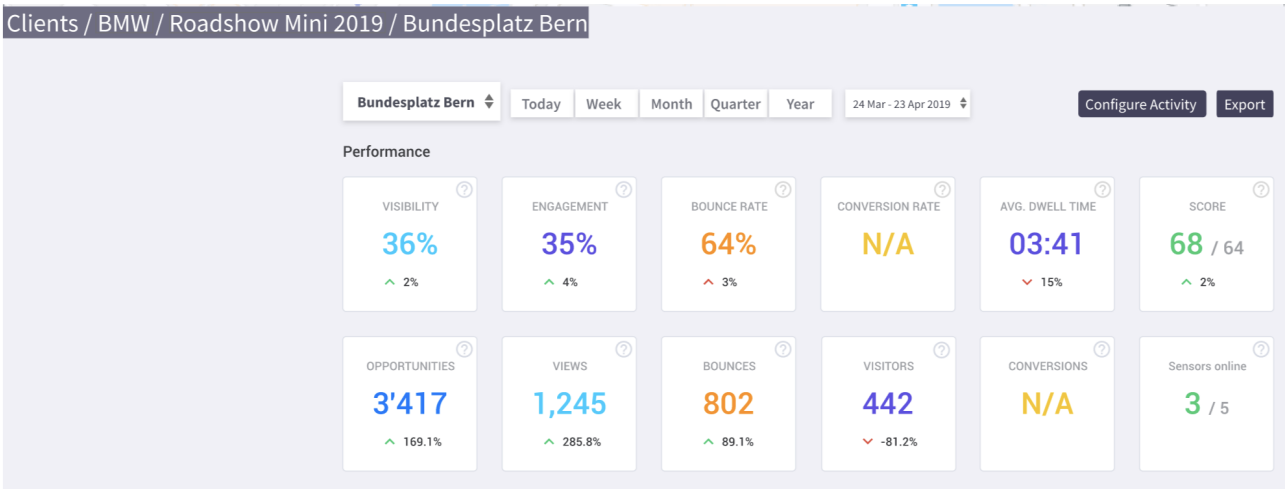

Figure 36: Activity Concept

sions, and activity-related factors. For example, heat maps are graphical representations of data with individual values contained in a matrix. These values are represented with different color shapes based on their intensity. They are used to relate metrics such as the average dwell time with different temporal dimensions (days and hours). Figure 37 illustrates a heat map with average dwell time sorted by hours and days.

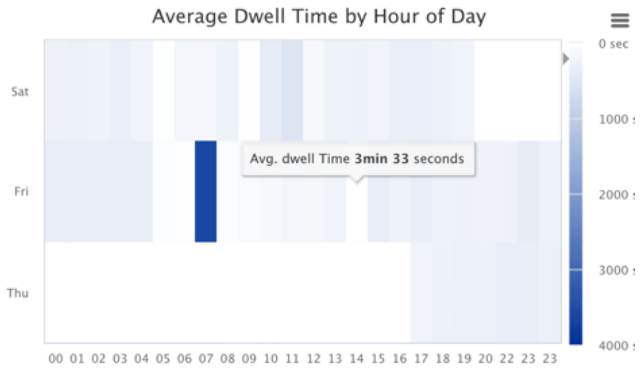

Figure 37: Heat Map

Figure 38: Histogram Series

A histogram creates intervals (bins) and counts how many values fall into each bin. Histograms are useful to visualize and relate zone of interest with temporal dimensions and numerical metrics. Figure 38 shows the relationship between the metrics' unique number of visitors, the temporal dimension days, and the zone of interest wrapped into a single histogram.

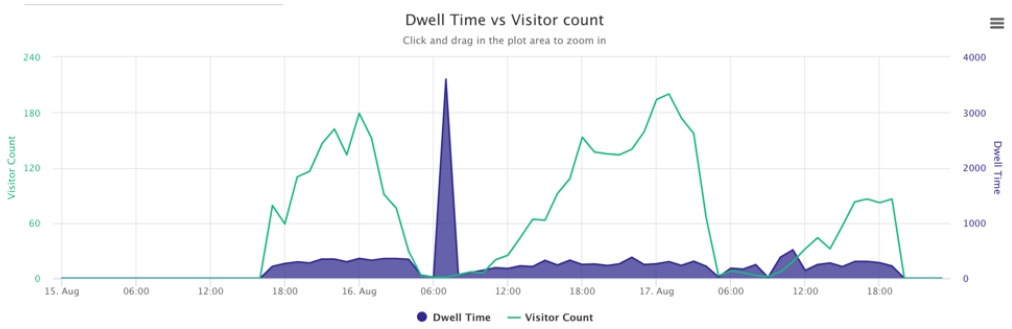

Figure 39: Line Plot

The line chart represents a series of data points connected with a straight line. Line charts are most often used to visualize data that changes over time. Figure 39 illustrates a line chart that relates metrics such as average dwell time and the visitor count, with a temporal dimension expressed in hours.

5 Experimental and In-field Evaluations

The experimental evaluations are concerned at first with the data input, secondly with the FusIon Data Tracking System FITS itself, and the data consumption lastly.

5.1 Data Input

In different settings the role of data input was evaluated. First for WiFi, second for Bluetooth, third with a LiDAR, and with a RFID and camera combination.

5.1.1 WiFi — ASIMOV

Three experiments had been conducted to test the localization-based approach of MAC address de-anonymization. While two experiments were focused on localization, the third one was run during a real-life event.

Localization Experiments Two experiments were conducted to localize devices based on the signals that it emits. Experiment 1 is a localization experiment based on the emitted RSSI values, measuring a designated sender's RSSI values on four different monitor nodes. Experiment 2 was a joint experiment: it included RSSI measurements from both WiFi probe requests and Bluetooth-based measurements. For both experiments, the packet dumping of the Probe Requests was conducted using *tshark* [96].

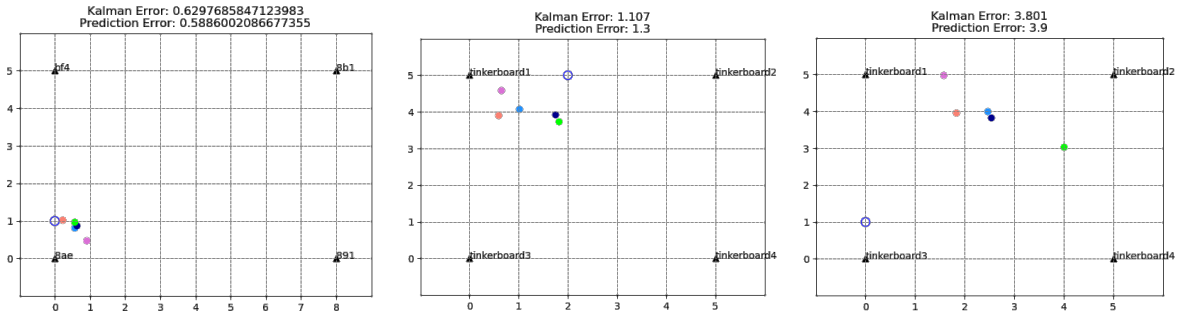

Figure 40: Sample Results of Localization Experiments. Left: Experiment 1 on Position (0,1). Middle: Experiment 2 at Position (2,5). Right: Sample of Step 1 of Experiment 2

Experiment 1. aims to localize a device at different points in space within the covered area. For this, four monitor nodes were placed at the corners of an 8 by 5-m grid to measure RSSI values while a sender moves one meter at a time, halting one minute per grid point. Starting with the first grid point, the sender started to send probe requests for one minute. After each minute, the sender was moved forward to the next grid point until it reaches all 48 grid points. As the sender was moving on a specified path, a Kalman Filter was used to smooth out the location estimates. The sender was a Raspberry Pi 3 with Alfa-AWUS036NHA Wifi-Adapter, and the monitors were four Spitz GL-Routers. Artificial probe requests were generated and transmitted at a high rate using packet injection and the python

package *Scapy* [12]. Each probe request contained a global sequence number and a burst number for later identification. This experiment was conducted in open field space.

Experiment 2 aims to compare and combine the location estimates of both types of frames and gain insights into whether a combined approach has better accuracy than the Wi-Fi-only method. It included RSSI measurements from both Wi-Fi probe requests and Bluetooth-based measurements. This experiment uses a configuration similar to experiment 1, but the area consists of a 5 by 5-m grid, with 32 measurement points, including an additional Bluetooth device in place. The hardware used for monitors is also different. For this experiment, monitors are four ASUS Tinkerboards equipped with an external network card Ralink RT5572N USB dongle for packet monitoring and an Ubertooth One, which is the additional hardware for Bluetooth packet capturing. This experiment was conducted in an urban environment on a concrete balcony.

Both experiments 1 and 2 were evaluated by measuring the deviation of the sender's estimated position from the sender's correct position. At each point on the grid, this deviation is calculated and, finally, averaged overall steps.

For experiment 1, the overall error in meters was 87 m without filtering and 58 meters applying Kalman Filter based smoothing. It results in a per step error of 1.7 m or 1.1 m with the Kalman Filter. Including the maximal possible distance of 10.8 m, this results in an error of 0.15 without filtering and 0.1 using Kalman Filter. The deviation of the unfiltered location estimates ranges from 0.03 to 5 m. A sample of Experiment 1 on position (0, 1) is shown in Figure 40. The true position is shown with the empty big blue circle. The different estimates of the unfiltered (light blue), Kalman Filter based (dark blue), and variance-based approaches (orange, green, violet) are shown as filled dots.

For experiment 2, the overall error in meters for the WiFi-based localization was 72 m without filtering and 68 m applying Kalman Filter based smoothing. It results in a per step error of 2.2 m or 2.1 m with the Kalman Filter. Including the maximal possible distance in 8.4 m, this results in an error of 0.26 without filtering and 0.25 with Kalman Filter. The deviation of the unfiltered location estimates ranges from 0.44 to 4.5 m. A sample of experiment 2 with the sender at position (2, 5) is shown in Figure 40.

Bluetooth-based localization in experiment 2 achieved an overall error of 68 m. It results in a per step error of 2.1 m. The deviation of the unfiltered location estimates ranges from 0.8 to 3.8 m. Including the maximal possible distance of 8.4 m, this results in an error of 0.25. Due to the limited time of 60 s used for the measurement, not enough data was collected to apply the Kalman smoothing.

Compared to experiment 1, experiment 2 produced slightly less accurate results, which can be seen in the higher per step error and the fact that the Kalman Filtering did only have a minor impact. The error does not source primarily from the variance of the different types of estimates. They all tend to be quite close together; instead, location estimates are completely off in their approximation (*cf.* Figure 40): the different location estimate variations are close together, but far apart from the sender's true location.

In-field Experiment Experiment 3 was conducted during a real-life event on June 2, 2020. This experiment counts devices instead of verifying location estimates. Therefore, the effectiveness of using the combined approach of IEs and location estimate was evaluated using the system proposed. For this experiment, a livealytics booth was placed at the event with a

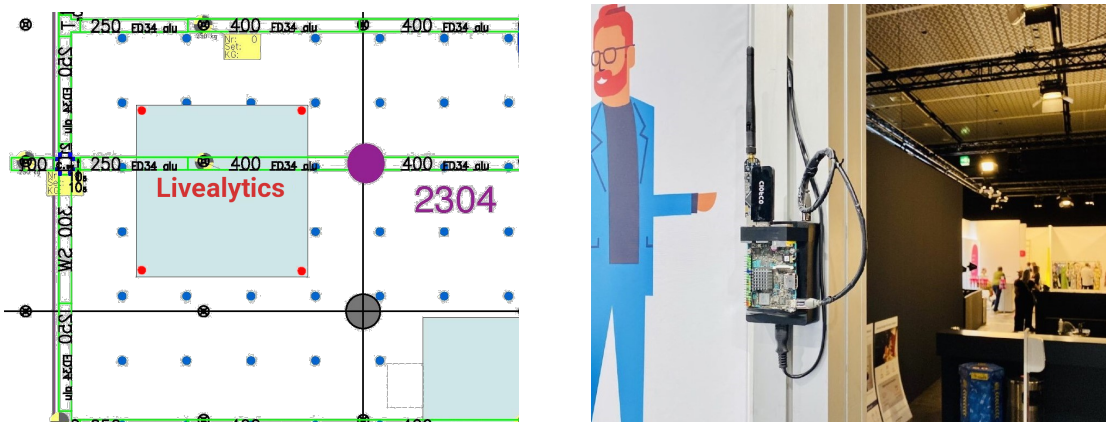

Figure 41: LiveAlytics in-field Experiment. Left: Map of the Livealytics Booth at the Event. Right: Monitoring Device in Use

4 by 4-m area (*cf.* Figure 41). The livealytics booth is represented by a cyan square and red dots represent monitoring devices. The ASIMOV monitor nodes were placed at two meters above the floor. The same devices were present in experiment 2, including all hard- and software components used as monitor nodes. Figure 41 shows the devices placed at this event. The monitor node code was slightly adapted to start and stop at specified times and store all data locally. The monitoring process took about three hours.

Since the ASIMOV can run analyses using numerous parameters and different user thresholds, it is possible to obtain different results. For the analysis of experiment 3, the following parameters were chosen: 10 s of interval size, 2 km/h for the assumed walking speed, 1s for in-burst threshold, and 3 for minimum AP detection rate. In this experiment, ASIMOV detected 425 different devices on-site; 370 devices were seen multiple times, while 55 appeared just once. Additionally, of all detected devices, ASIMOV was able to identify 301 devices applying MAC randomization and 69 devices that were not. Overall, 708 different MAC addresses were captured.

For this event, two external data sources were available to compare: a ticketing system and a ceiling camera. The ticketing system counts the number of people entering and the number of people leaving the site. The ceiling camera keeps track of the number of people entering and exiting the area of the livealytics booth. However, both external evaluation sources had a slightly different focus than the ASIMOV system: the ticketing system counted all people entering and leaving the building and, therefore, did keep track of the overall event. The ceiling camera was limited to the stand's exact borders and did not include any person outside of this area. An overall number of 566 people's first entrances were accounted for, which corresponds to the number of total visitors that entered the site for the full day. This count, however, includes not only visitors but also all staff and exhibitors. The official number of visitors during the entire day was 360. Additionally, the amount of people seen multiple times using MAC address randomization is 370.

Assumptions The proposed system's principles lead to assumptions of device density, covered area, moving patterns, and interferences, which were made accessible. For device density, it is assumed that one device represents one person, which is usually the case for

mobile phones (one person typically carries one mobile phone). Since the system counts devices, people carrying multiple devices can break with these assumptions. In such a case, it can reduce the accuracy to infer the number of people present in a place. It is further assumed that all devices are with WiFi turned on, producing probe requests.

The system needs to be evenly distributed in the area under surveillance. If the monitoring devices can not cover the area entirely or have too much overlapping space, the localization could fail. Further, it is assumed that people carrying devices are moving around in space at a relatively slow pace. Even though the ASIMOV system can be configured to different walking speeds, the localization approach fails if the devices' position changes quicker than the monitor nodes receive new probe requests. Lastly, it is assumed that no interference with other systems producing probe requests, modifying existing packets, or doing any radio wave disturbance.

Results Localization experiments have shown different results according to the hardware utilized as well as other factors involved. For experiment 1, the RSSI-based localization of a transmitting device using probe requests was successful, as the average deviation was low — about 1 m per position using the Kalman Filter. Different from experiment 1, experiment 2 produced slightly less accurate results. This fact can be seen in the higher per step error and the fact that the Kalman Filtering did only have a minor impact. The error does not source primarily in the variance of the different types of estimates. They all tend to be quite close together; instead, the location estimates are completely off in their approximation. All the obtained location estimates variations are close together but far from the sender's correct location.

Possible reasons for the lower overall accuracy in experiment 2 include the environment (it was tested on a concrete floor) and different hardware used (an ASUS Tinkerboard with Ralink RT5572N WiFi adapter). Especially, the environmental aspects are known factors for changing the measured RSSI values [65]. Additionally, it is possible and based on the visuals in Figure 40, that the monitor on the bottom left corner was producing questionable measurements. This behavior could have been provoked by the direct sunlight exposure of this monitor compared to all other monitors [65].

It is not possible to state, whether the different behavior of experiment 1 in comparison to experiment 2 originates from the location or the different hardware. However, the first experiment's success was only repeated partially in the second experiment: first, the Bluetooth-based approach had hardware-based failures, and secondly, the WiFi-based approach had a higher error rate than in the previous experiment. Therefore, based on results of experiment 2, a combined approach of WiFi- and Bluetooth-based localization is unlikely to improve.

For experiment 3, both retrieved validation numbers fall short to measure the number of people that passed by the Livealytics booth due to the following reasons: the number of devices counted by the ASIMOV system, 425, is between 566 and 360, and the number of people counted in ticketing system and the total visiting people, respectively. Therefore, it approximates the number of people present at the event. When the amount of detected MAC addresses are compared to the number of devices identified, ASIMOV's key contribution can be seen.

Limitations Apart from the advantages that the proposed approach has compared with other ways to handle MAC address randomization, some limitations exist regarding the used methods. The first concern is related to the instability of RSSI. One of the main reasons some systems do not rely on the RSSI values [87][99] [62] [38] for localization is the fact that it is error-prone and unstable. However, as the proposed system's prime goal is not localization but to distinguish devices from each other, the problem becomes less relevant, and only in the worst case, this become an issue. In this solution, the IE identifies the device in the first place [86]. Localization is only used if the IE identification fails due to having too similar devices and the same IEs.

The proposed solution relies mainly on probe requests. If no such frames are transmitted, the system is rendered impractical. However, compared to other frame types, probe requests do not require association or interaction with the device and have a high transmission rate compared other frame types. Additionally, probe requests are transmitted in bursts, which increases the chance of a single packet being monitored by multiple receivers. Compared to other proposed approaches and concerning the proposed solution, it can be stated that it is an accurate effort to handle the difficulties imposed by MAC randomization.

5.1.2 Bluetooth — BluePIL

Experiments were conducted in an indoor and outdoor space to evaluate the effectiveness of *BluePIL*'s device localization method. The respective scenes are shown in Figure 42. The indoor experiment was performed in a room at that time being empty. This was ideal since it allowed to keep the amount of signal interferences as low as possible.

The outdoor experiment was done on a private terrace in a residential area of Zürich. This allowed to decrease the amount of signal interference from multi-path fading, since more space was available. A $4.2\,\mathrm{m} \times 2.9\,\mathrm{m}$ area was designated to perform the experiments. An Ubertooth One sensor was placed at each of the four corners of this space and connected to a MacBook Pro via a 2.5 m USB cable. Nine points were defined, where static measurements would be carried out.

Figure 42: Evaluation Environments: (Left) Indoor and (Right) Outdoor

Experiment 1 — Indoor This experiment was then conducted via a *Nokia 7 Plus* smartphone being connected to a pair of *JBL Reflect Flow* Bluetooth headphones. Music was streamed over said connection throughout the experiment to generate traffic that could be captured passively. In order to keep the conditions as realistic as possible, the headphones were placed in a test subject's ears and the smartphone in their front right pant pocket. The test subject then stood for 5 min at each of the nine points indicated in Figure 43. The four Ubertooth One sensors were configured to record any packets that could be intercepted during that time interval.

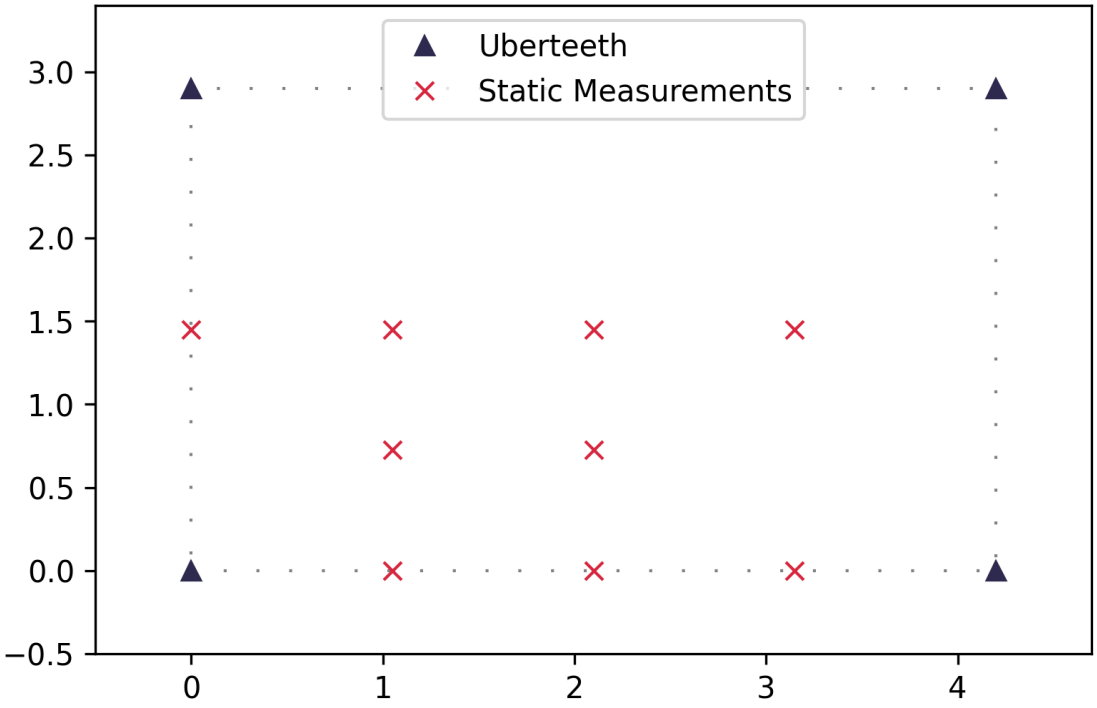

Figure 43: Indoor Setup (x and y Axes in Meter)

Experiment 1 used a static version of the localization algorithm, which allowed for simplifying the merging of data sets from individual sensors. The streaming interpolation method used in the final processing pipeline could be omitted and the data could be merged using a static interpolation and re-sampling process. While this initial approach differs slightly from the final system, it does not invalidate results of this experiment as an evaluation of the device localization method. This experiment also included an evaluation of the filtering methods used in the *BluePIL* processing pipeline, namely the method used for signal strength filtering and location filtering. The following variants were included for signal strength filtering: a simple rolling mean filter, a rolling maximum followed by a rolling mean filter, and a rolling maximum followed by a rolling median filter. With regards to location filtering, the improvement gained by the Kalman filter was analyzed.

Experiment 2 — Outdoor A second experiment was designed to evaluate the *BluePIL* device localization method under more challenging conditions and to evaluate the system's performance in its final streaming architecture. A space of $5\,\mathrm{m} \times 5\,\mathrm{m}$ was designated to

Table 6: Indoor Environment Experiment Results

True Point (m)	**Mean Estimated Point** (m, rounded)	**Mean Error** (m, rounded)	**Mean No. Meas/Sensor**
$(2.10, 0.00)$	$(1.798, 0.826)$	0.888	258.5
$(1.05, 0.00)$	$(1.695, 1.692)$	1.856	253.25
$(3.15, 0.00)$	$(3.193, 1.629)$	1.718	206.25
$(2.10, 1.45)$	–	–	–
$(1.05, 1.45)$	$(0.942, 1.522)$	0.612	231.0
$(3.15, 1.45)$	$(2.761, 1.965)$	0.671	299.5
$(0.00, 1.45)$	$(0.556, 1.238)$	0.682	249.75
$(1.05, 0.73)$	$(1.118, 1.412)$	0.822	190.75
$(2.10, 0.73)$	$(1.931, 1.809)$	1.239	228.0
Overall Mean Error: 1.061		**Overall Mean/Sensor**: 293.63	

perform the experiment. An Ubertooth One sensor was placed at each corner of the space, connected to an Asus Tinkerboard. A MacBook Pro was used to control these nodes. 32 equally spaced points were chosen in the $5\,\mathrm{m} \times 5\,\mathrm{m}$ space (*cf.* Figure 44). A test subject holding a Nokia smartphone, which was streaming audio to a pair of Bluetooth headphones, traversed these points, resting at each one for one minute. During this minute, data was captured by Ubertooth sensors.

A first approach repeated Experiment 1 under more challenging conditions. 32 equally spaced points were chosen in the $5\,\mathrm{m} \times 5\,\mathrm{m}$ space. They are shown in Figure 44. Experiment 2 follows the same testing procedure as Experiment 1 (*i.e.,* same test subject and moving pattern). This data was analyzed ex-post with the same static version of the pipeline used in Experiment 1.

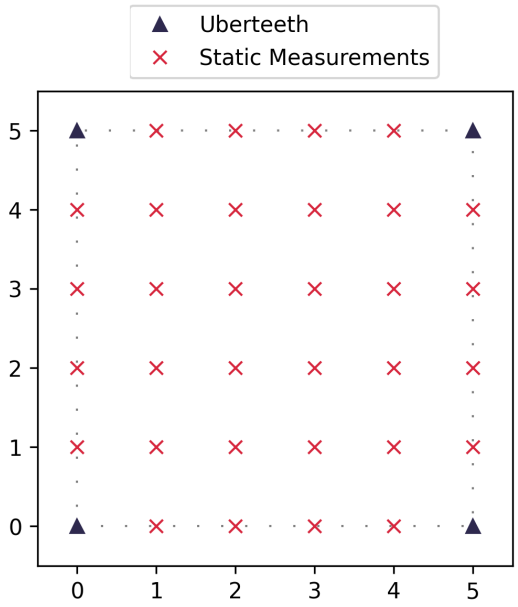

Figure 44: Outdoor Setup (x and y Axes in Meter)

Table 7: Outdoor Environment Experiment Results

True Point (m)	**Mean Estimated Point** (m, rounded)	**Mean Error** (m, rounded)	**No. Localizations**
$(1, 1)$	$(0.989, 1.942)$	1.263	36
$(1, 4)$	$(1.160, 3.108$	1.255	33
$(4, 1)$	$(3.465, 3.188)$	2.320	26
$(4, 4)$	$(3.227, 3.766)$	1.065	26
$(2.5, 2.5)$	$(3.257, 1.703)$	1.129	22
Overall Mean Error: 1.406		**Mean No. Localizations**: 28.6	

This second step tested the system in its full streaming implementation. To this end, five points were chosen in the $5\,\mathrm{m} \times 5\,\mathrm{m}$ space. Again, the Nokia smartphone and the JBL headphones were used to stream audio over BT. The near real-time positioning pipeline was then run for 15 mins. During this time, the test subject covered each of the five points, resting at each point for 2 mins and taking a maximum of one minute for the change between points. One minute of buffering time was included at the beginning. These points were traversed according to the following order: $(1, 1) \to (1, 4) \to (4, 1) \to (4, 4) \to (2.5, 2.5)$.

Tables 6 show the results of the indoor experiments. The average location estimation, the average localization errors, and the average number of measurements per sensor for the Nokia smartphone's LAP is shown. Data for the point $(2.1, 1.45)$ is missing in these results from the indoor experiments, due to the failure of one of the sensors that were only noticed after the completion of the experiment.

Experiment 2 revealed concerns with the Ubertooth sensors used. Most importantly, the performance regarding the number of packets captured deteriorated significantly from Experiment 1. During the first part of the experiment, the number of packets captured per second and per sensor was reduced to about 0.38 compared to 1.0 from before. This created problems in the location computation, especially in the first part of the experiment: Due to the decreased number and the fact that captured packets were not evenly spread throughout the time interval, it occurred that, for some of the points, there was no overlap between these points in time of the RSSI measurements. Consequently, it was impossible to merge the RSSI value streams between sensors. Only points that produced a sufficient overlap of a minimum of 10 s were, therefore, analyzed for the first part of the experiment.

Table 7 and Figure 45 show results for the second part of the experiment. It is noticeable that the streaming processing pipeline handles the sparsity of sensor measurements better than individual evaluations performed in part one of the experiment. This is mostly because the processing stream is able to use values for interpolation that lie outside time intervals defined as resting periods at each point. These results are more accurate than for the first part of the experiment, but worse than in Experiment 1 with an overall mean error of $1.406\,\mathrm{m}$. It should be noted, however, that the mean error value for point $(4, 1)$ forms an outlier, differing from the next lower value by $1.057\,\mathrm{m}$, more than five times the difference between any other two points ($0.198\,\mathrm{m}$). This corresponds to a pattern that was already observed in Experiment 1.

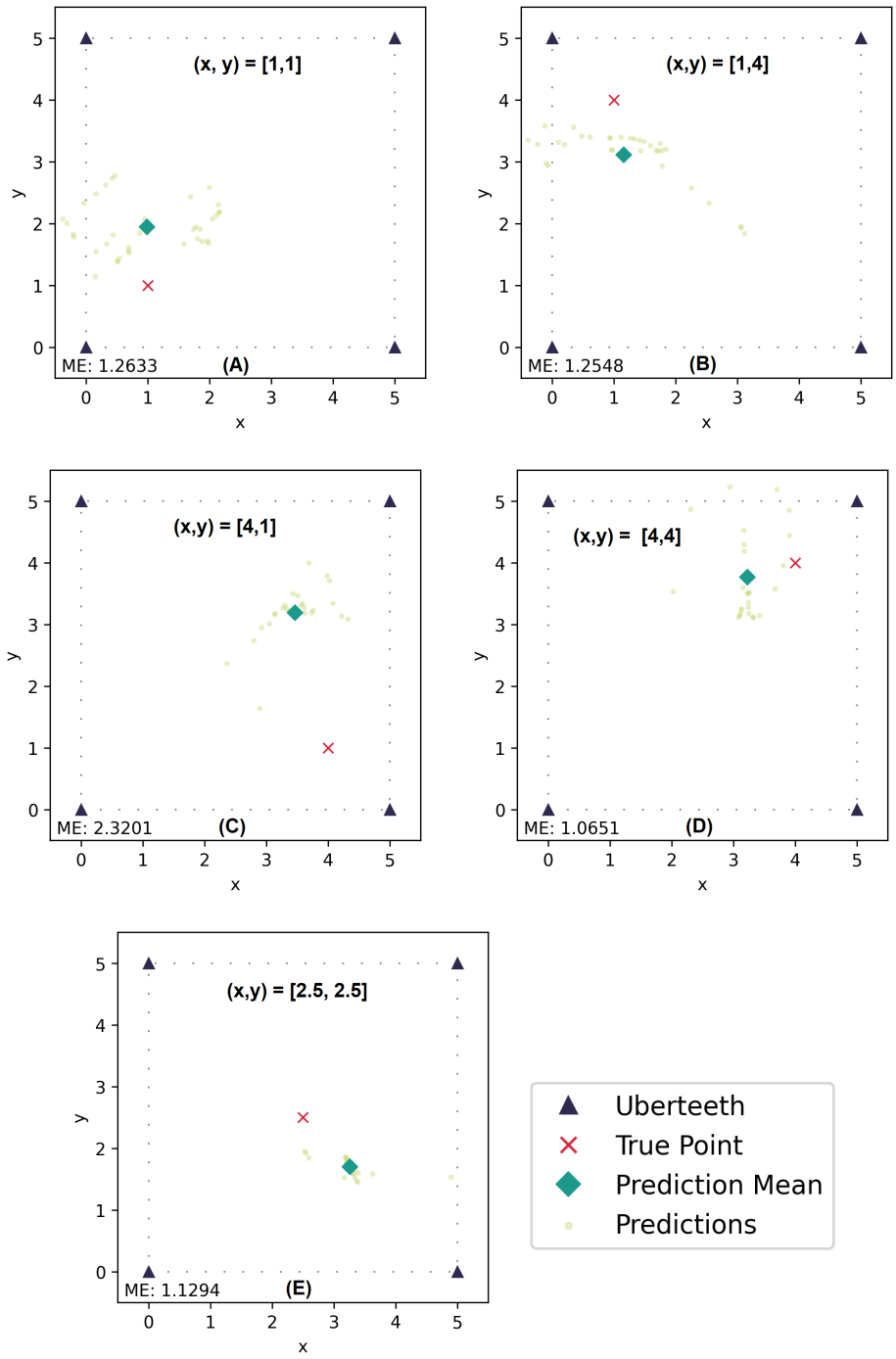

Figure 45: Outdoor Experiment. (a) Point [1,1], (b) Point [1,4], (c) Point [4,1], (d) Point [4,4], and (e) Point [2.5,2.5]

Discussion of Results and Findings All evaluations were conducted in real-world environment containing externalities such as multipath fading, interference with other objects in the room, and other wireless signals. Overall, the positioning accuracy in both experiments was fairly similar, with an average error of $1.04\,\mathrm{m}$ and $1.061\,\mathrm{m}$ for the outdoor and the indoor experiments, respectively. The error values ranged from $0.398\,\mathrm{m}$ to $1.703\,\mathrm{m}$ for the

outdoor and $0.612\,\mathrm{m}$ to $1.856\,\mathrm{m}$ for the indoor experiment. While the overall sensor performance was quite similar in the indoor and the outdoor experiment, collecting around one measurement per second, it was more stable in the indoor environment, where the mean number of measurements per sensor for the Nokia smartphone's LAP ranged from 190.75 to 299.5 compared to 132.75 to 439.5 in the outdoor case. Both sets of results show some outliers in the upper range of the error values, most notably point $(1.05, 0.00)$ for the outdoor experiment and points $(1.05, 0.00)$ and $(3.15, 0.00)$ for the indoor experiment.

Device localization in Experiment 1 produces results at around $1\,\mathrm{m}$ accuracy on average, both in indoor and outdoor spaces without a prior calibration of the system to estimate RSS_C and, in a completely passive manner, setting it apart from existing approaches. Results for the indoor space were slightly worse, which may be explained by higher amounts of noise from multi-path fading due to the more constrained dimensions. The first part of Experiment 2 did not show the same level of success. The decreased timespan allocated for each measurement combined with the deteriorated performance of the sensors led to large parts of the data being unusable. The performance for remaining data points was significantly worse than in the first experiment. The second part of the second experiment, however, showed the effectiveness of the streaming architecture, most notably the signal strength merger component, which dealt well with the additional sparsity of signal strength measurements encountered. The localization performance was good, apart from a single outlier, a pattern which was also observed during the first experiment. These results are comparable to existing research, which is advantageous given that the *BluePIL* system estimates locations and channel parameters simultaneously.

The existence of negative outliers in localization results in both experiments warrants discussion. An inspection of the raw data revealed that these outliers were not caused by a failure in the processing pipeline, but were already present in the data received from the sensors. A plausible explanation for this is that environmental factors, or possibly a combination thereof, will have lead to RSSI values not representing the location of the test subject accurately. The experiments took place in a residential area of Zürich, some of them outdoors. Consequently, influencing factors, such as background noise, the topology of the space, temperature, and humidity, could not be controlled and may have led to the disturbance of the signal. The test subject carrying the test devices may itself have had an effect on the signal strengths measured, as is had been suggested by the investigation of the influence of the human body on RSSI measurements [91]. The Ubertooth sensors used present another possible explanation since the accuracy and consistency of the RSSI values they deliver is unknown.

The deterioration of sensor performance in Experiment 2 presented a further challenge. Since the environment was identical to the outdoor space used in the first experiment and the frequency of measurements was lower regardless of the distance from the sensor, these two factors do not explain the degradation. One potential explanation could be the change in weather conditions. While the first experiment occurred in spring, the second one was performed in mid-summer, with temperatures exceeding $30°$C on an asphalt surface. The sensors may have overheated, leading to the decreased frequency and lower accuracy of measurements in the radio components. Existing research suggests a strong influence of operating temperature on RSSI measurements for chips very similar to the one used in the Ubertooth One [15]. The unpredictability of these results may have been exacerbated by shade reaching some of the sensors over the duration of the experiment, breaking the

assumption that all four sensors exhibit the same radio characteristics. This hypothesis could also explain the better performance in the second part of the experiment, which was executed later on the same day when temperatures were lower and the entire environment was covered by shade.

5.1.3 LiDAR — LaFlector and LiCounter

LaFlector (2D LiDAR) The evaluation of LaFlector was conducted in a 18 m^2 room. The windows were covered and there were static objects in the room (*e.g.,* sofa, table, TV, and chairs). Runs were performed with the parameter's default value. To simulate objects' appearance, both entering the measurement room and spontaneously joining the measurement height were attempted. To test an object's disappearance, the tracked person left the room or suddenly went below the measurement height. The measurement height was 144 cm above the ground for all test runs. Four scenarios were run five times each:

1. **Single Person Tracking**: One person moves through the room at a walking pace. There are no other moving objects. This scenario is the basis for further evaluations.
2. **Two Persons Tracking**: Two people move simultaneously at a walking pace in the room. They never stand behind each other and are always at least one meter apart. Their paths might intersect at different times.
3. **Two Persons Crossing**: Two people cross paths. For a short time, one person stands behind the other. They do not make any hard changes on their direction.
4. **Single Person Behind Static Object**: A static object stands in the room, which was captured at the beginning. A person moves behind the object for a short amount of time and continues in the walking direction.

These runs were measured using the following criteria. A criterion can be considered as passed or failed.

- **Positioning (Pos.)**: The tracking is accurate and continuous. The criterion is considered as passed if the object is tracked accordingly to the real position and there are no unexpected jumps in the way-points.
- **Classification (Class.)**: The person is recognized and correctly classified. No static objects or objects are classified as human objects. The criterion is considered as passed when the number of human objects is correctly detected.
- **Identification (Id.)**: The same object is always identified as the same. If a person reappearing from behind a static object is classified as a new object, the criterion is considered failed. If two people cross each other and the system can no longer assign the objects because the object identification was lost, the criterion is considered failed.

Table 8: Failures in Each Scenario

#	Scenario	Pos.	Class.	Id.
1	Single Person Tracking	0	0	0
2	Two Persons Tracking	0	0	0
3	Two Persons Crossing	0	0	2
4	Person behind Object	0	0	1

Results Table 8 presents how often a criterion was not met in 5 runs. In general, it can be stated that the location algorithm worked as expected for most scenarios. However, there were failures in the identification and classification in Scenarios #3 and #4, which are explained in the next sections. Results are represented with the created plots of the LaFlector. A series of snapshots visualize the first scenario. These snapshots were taken at a regular interval (every 5 seconds) during the run, in which the last snapshot represents the remaining three scenarios before the object disappears/dies. Way-points are recorded for the evaluation. In the usual tracking mode, the way-points are not displayed by default. If desired, the plotting of way-points can be enabled in the plotter class.

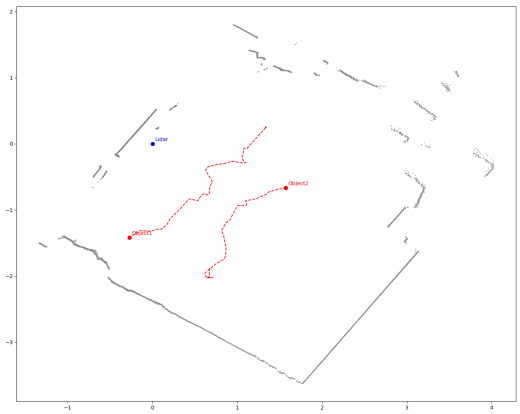

Figure 46: Scenario #3: Two Persons Crossing Plot

Scenario #1: Single Person In the first scenario, during all five runs, the person was correctly detected, tracked without interruption, and removed again after disappearing (*i.e.,* marked as “dead”). No static objects were classified as people. Figure 47 depicts the progress of the run with a running time of 40 s, in which the object alternately moved and remained stationary.

Scenario #2: Two or More Persons not Crossing Paths In the second scenario, two people were successfully tracked, as illustrated in Figure 46. Both persons were recognized at the same time, and their path was tracked correctly. Even when two people/objects were

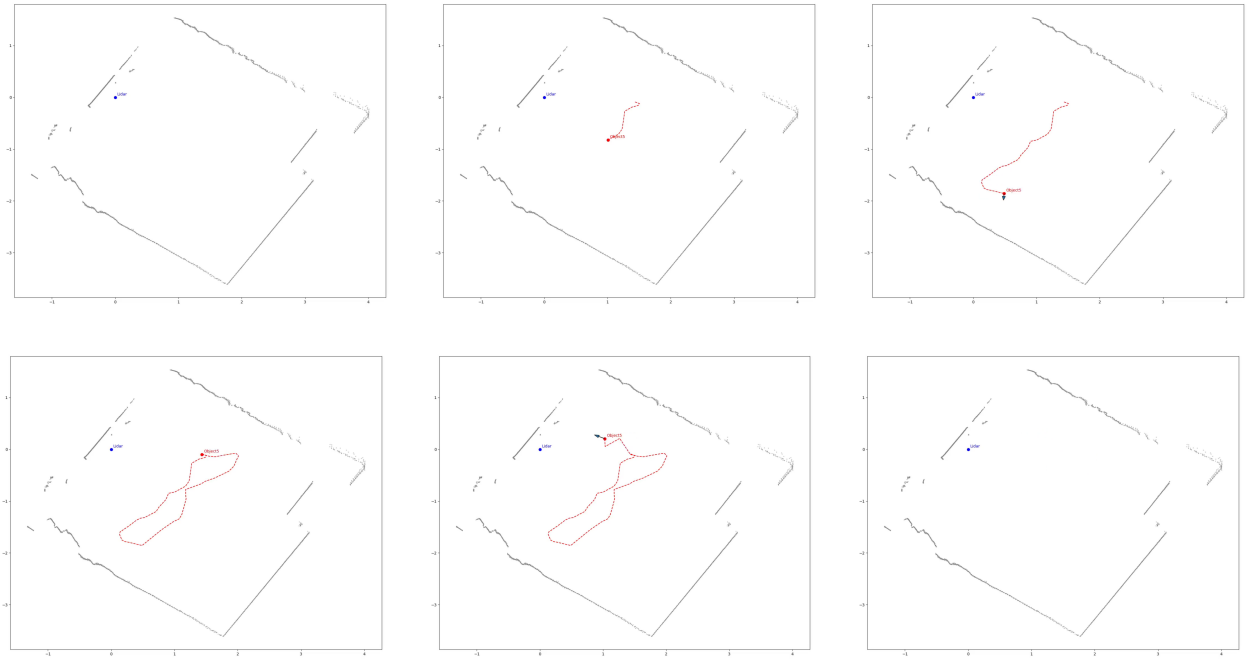

Figure 47: Snapshot Series of a Tracking Run, Where (a) Object Not Detected in T_0, (b), Object is Detected and Tracking Started in T_4, (c) In T_8, Active Tracking and Direction Vector is Visible, (d) Object Stopped and No Direction Vector is Calculated in T_{12} (e) Active Tracking and Direction Vector Visible in T_{16}, and (f) Object Died and All Way-points Cleared in T_{20} [T Measured in Seconds]. Video Captures of the Evaluation Can Be Found at [72].

close to each other (but not behind each other), the identification worked without any issues in five runs. The tracking was performed during 35 s.

Scenario #3: Two Persons Crossing Paths In the third scenario, two people cross paths, assuming that they do not change their speed significantly. If a person started walking while being covered by the other person, there would be no heuristic possibility to detect this. Since according to the last confirmed information, the person was at rest. The five crossings were performed with the following estimated crossing angles: twice 180 degrees, once 120 degrees, once 90 degrees, and once 60 degrees. In two out of five cases, the individuals could not be positively identified after crossing. The smaller the crossing angle, the higher the probability that the recognition fails because the persons are not distinguishable for the LiDAR for a longer time. Accordingly, the identification of the objects failed at 90 and 60-degree crossing angles. The best results can be achieved when the persons cross at a straight angle (180 degrees). The plot of a run with a duration of 8 seconds and a straight angle is depicted in Figure 46.

Scenario #4: Single Person Behind Static Object A flaw in the location algorithm occurred in the second run of Scenario #4, after reappearing behind the static object. The object was identified as a new object instead of the original due to a sudden change in speed or change of walking directly behind the static object. Once the model defined in [7] represents

a ”near-constant” velocity variation, the heuristic prediction model failed considering the covariance Q defined in the white noise Gaussian process. This highlights the importance of choosing a suitable location model for the characteristics of the environment. Considering that abrupt accelerations are exceptions for indoor tracking reproducing the behavior of an exhibition, it is possible to tolerate such outliers.

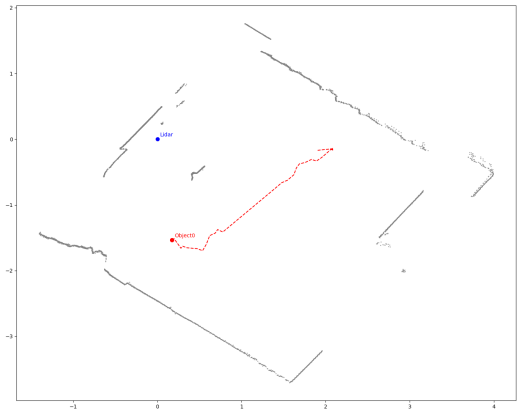

Figure 48: Scenario #4: Single Person behind Object Plot

Discussion The results reflect the strengths and weaknesses of a LiDAR sensor. As expected, the positioning was accurate. However, as a natural disadvantage of tracking devices based on light reflection, it is impossible to track when objects are obscured by other objects, regardless of whether they are static or in motion. Herein, it can only be evaluated with a certain probability whether the object still exists and where it is located when reappearing. How well this case works also depends significantly on the parameterization. For example, if people are expected to move close to each other, the split distance and existing-new-threshold parameters should be decreased. This increases the probability that the person will be detected. At the same time, it increases the risk that single persons will be detected as several.

The LiDAR interface was responsible for making the data from the LiDAR scanner available for data processing. The data processing always received the latest laser data via the Influx database and the data acquisition worked without interruptions via the socket.

From the first scenario, it can be deduced that the segmentation of static and moving objects works reliably. The moving object was recognized as such and tracked. Further, scenario 2 shows that the system can detect and track two or more people simultaneously. Errors occurred in scenario 3, in which LaFlector could no longer determine which of the intersecting persons was in two out of five cases. Scenario 4 further demonstrates that the heuristic to predict movement worked as expected being able to associate the path with the object.

Objects were recognized after disappearance except for one case. It follows that when two crossing objects are in motion, the system appears to be less accurate than when one of

the objects is static. It is possible that even better results could be obtained with more complex, finer-grained heuristic predictions. However, this would also require a higher-resolution LiDAR scanner. In particular, far-away objects from the scanner would otherwise take too long to be reliably detected since the density of measurement points decreases with distance.

The graphical illustrations from the results are from the plotter of the system. The log file created with each run contains the desired information according to the selected log level. It is, therefore possible to read (x, y) points that an object had during a run. With a simple code extension, the logger's data could be made available for further processing or combination with another data source. Therefore, the requirement of expandability can also be confirmed.

LiCounter (3D LiDAR) This section presents the result of clustering from data that the LiCounter produced during its experiments. Due to the limitations of measurements in a real case, a scenario is made for testing the developed method with a mock shopping scenario. A product is chosen as the point of interest, in this case, it is the scooter. Testers need to pass through this scenario by performing as they are shopping, some of them are interested in the scooter by standing close to the scooter and staying longer; and some are not interested, thus passing by the scooter quicker. The data are trying to collect as many different use cases as possible. A home basement location is chosen with complete darkness. Model 5 weights are used. Scenarios are designed as the Figure 49 shows. Two LiDARs are facing each other from 4.6 meters away, the object is located in the middle(scooter), 2.3 meters from 2 cameras.

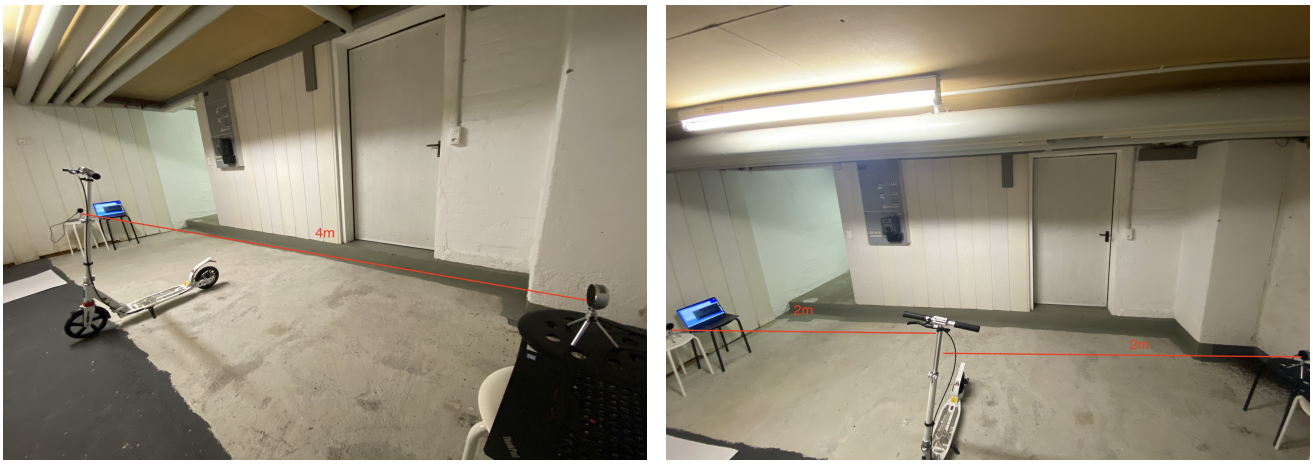

Figure 49: Testing Scenario Setup

In the designed scenario, the goal is to collect data from different people who behave differently. Specifically, people staying within 0.5 meters of the scooter for more than 10 seconds mean they are interested. People staying more than 1 meter away from the scooter for an arbitrary amount of time are not interested. People staying within 0.5 meters of the scooter but for less than 10 seconds are defined as random actions. Doing any random stay to enrich the variety in the dataset.

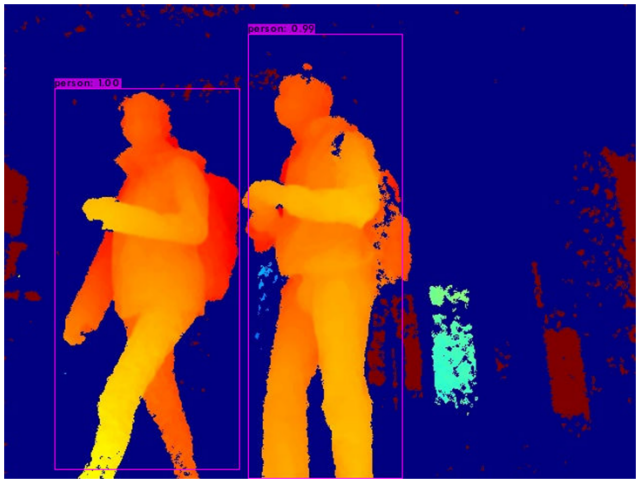

Figure 50: Testing Image from Model

The ground is flat enough to ensure that two LiDAR cameras are at the same height, facing each other to detect the people. Since the space is limited in the evaluation scenario, only one point of interest – the scooter is placed in the test, which cannot represent a shopping case. There is also a limitation with the number of testers. Only two testers are in the testing scenario and try to simulate a scenario of 40+ people by walking out of the frame and returning. However, the Deep Sort algorithm can sometimes identify the tester as the same person tracked and assigned the same ID. If more than 40 different people are testing, is it not likely for Deep Sort to assign the same ID to different people. If more objects could be placed and more testers could join, the testing scenario will better represent a real shopping scenario

Aggregated Data Duration and distance to point of interest can be aggregated according to ID. By seeing the duration and distance data together, we can tell whether a person is interested or not. User 10_1, 26196_1, and 26340_1 have a duration of 0 seconds. Since we assume a person cannot pass through the frame within 1 second, we consider them outliers. Examples are given in the following tables and *cf.* Table 9.
Interested:

- User 12 stays for 17 seconds, is as close as 0.1 meter to the point of interest, and is always within 1 meter.
- User 1_1 stays for 10 seconds, with a mean distance below 1 meter, and a minimum distance of 0.5 meters.
- User m_4 stays for the longest duration, almost 5 minutes, with a minimum distance below 0.2 meters, and on average below 1 meter.

Not Interested:

- User 26318 stays for 124 seconds, which is a relatively long time. The distance is however mostly above 1 meter.

Table 9: Table of Aggregated Data

ID	duration_s	min_distance	max_distance	mean_distance	median_distance
10_1	0	333.548	333.548	333.548	333.548
12	17	135.754	986.894	746.168	862.349
17_1	12	1043.73	1696.26	1242.42	1221.86
1_1	10	503.734	1195.82	878.085	939.682
21_1	55	535.317	2000	944.518	642.119
26148	5	935.945	1137.72	1009.91	982.993
26196_1	0	2000	2000	2000	2000
26318	124	969.066	2000	1426.55	1420.23
26333	9	932.863	1303.73	1046.97	948.048
26340_1	0	2000	2000	2000	2000
26349_1	36	919.834	2000	1158.82	1076.66
27_1	17	1045.88	1406.73	1155.09	1086.91
29_1	1	1318.52	1323.64	1321.08	1321.08
62_1	197	1202.48	2149.75	1361.29	1224.29
71_1	6	986.309	1127.39	1065.69	1072.92
m_0	192	186.17	2000	1198.11	1274.3
m_1	290	282.726	2040.44	1269.23	1267.13
m_2	295	424.273	2000	886.104	799.889
m_3	124	145.922	1029.47	608.459	714.834
m_4	298	187.018	2000	907.82	869.639
m_5	94	595.206	1011.2	886.911	939.012
m_6	94	452.63	2000	1011.86	990.608

- User 17_1 is also always more than 1 meter away, staying for 13 seconds.

Unclear:

- User 26349_1 stayed for 36 seconds, but mostly more than 1 meter to the point of interest.

Clustering We compare the duration to the different aggregated measures of distance. 16 distinct persons are detected in this testing scenario. Furthermore, to avoid repetitive counts, IDs of people that are simultaneously detected by both cameras are merged.

1. **Duration and mean distance:** Mean distance is distance weighted by duration since the longer duration the more apparent, thus more weight. Only two clusters are identified. This clustering result ignores the distance to the point of interest as a horizontal line at around 125 seconds can well separate the two clusters *cf.* Figure 51.

2. **Duration and median distance:** Three clusters are identified in Figure 51. Cluster 1 (yellow) with a duration below 60 seconds, regardless of the distance, Cluster 2 (purple) with a duration above 150 seconds and a distance above 1.2 meters, and

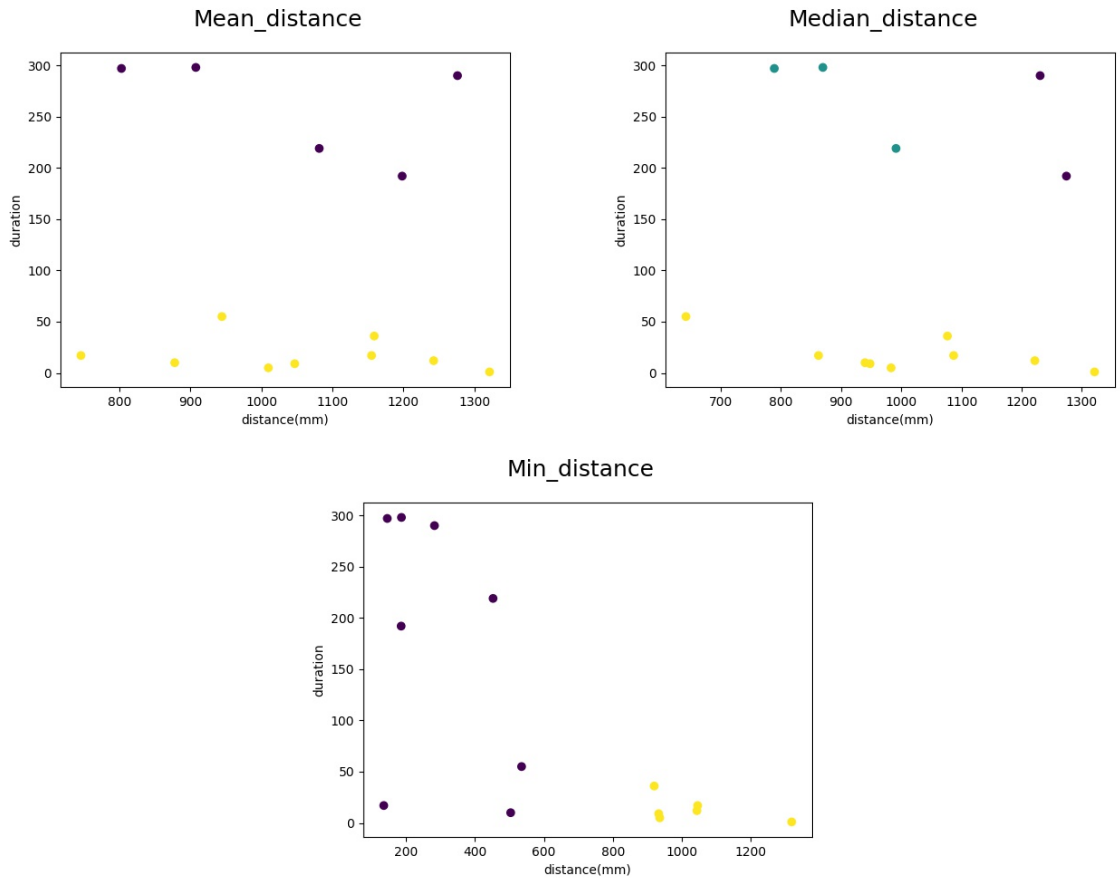

Figure 51: Clustering of IDs.

cluster 3 (blue) with a duration above 200 seconds, distance below 1 meter. Cluster 1 is a mix of interested and uninterested people. Those who stay below 10 seconds, no matter how long the distance, are not likely to be interested. Those who stay a longer time with a distance below 1 meter are likely to be interested. This also fits our description of an unclear group. Cluster 2 represents people who stay long but far away from the point of interest. We consider them as not interested since 1.2 meters is too far to see the point of interest clearly. People might be just staying somewhere and chatting. Cluster 3 is more likely to be an interested group, since they stay very long and relatively near (below 1 meter).

3. **Duration and min distance:** Min distance shows more clearly the group of people who are not interested. Cluster 2, the yellow cluster in Figure 51, shows people who have never been within 0.9 meter of the point of interest.

LiDAR Limitations The results can reflect the limitation of the solution for this project, which can be used for further development.

- Inaccurate LiDAR. The detection of depth from LiDAR is inaccurate, and this type of camera does not offer any tool for calibration so far. The inaccuracy of in-depth measurement is the limitation of the device that cannot be overcome by data analysis methods currently.

- Light Measurement. The detected distance based on light is measured from an app on a mobile phone, which only measures indoor lighting conditions. It is not professional to measure the influence of daylight. If it can be solved, the measurements and setup of scenarios can be more accurate.
- Limited Testing Space. Due to the coronavirus, it is not easy for many people to gather together to test. Thus, we lack data to represent the real case. Also, the lack of adequate physical space to implement the camera and evaluate the method.
- Limited Data. Due to the lack of data, we cannot calculate the metrics mentioned. For example, we do not have the data from different venues and points of interest, which is unable to get results about the percentage of interested users, variation of user flow, and variation of interest.

5.1.4 RFID and Camera — CCount

The evaluation of CCount combines and matches data collected by RFID readers with those of 3D cameras.

Real-world test scenario A 3D camera was installed and configured at the Axelra office to quickly test the efficiency of the data processing pipeline. Despite being able to quickly identify and correct possible bugs, the local testing environment did not evaluate important aspects, such as scalability and system reliability. Due to the collaboration with Livealytics, we had the opportunity to collect and process data from cameras installed at a local shop selling sports and outdoor activities clothes. It included data from four cameras in total, each recording different zones of the shop.

Figure 52 shows the scene from the camera placed at the outdoor department of the store. The camera collected and processed the data for generating the heatmap and the back-end. Xovis claims that 3D cameras can recognize visitors individually, even if they are standing nearby. Thus, to avoid nearby objects being recognized as visitors, the camera had to be configured accordingly, ensuring that it would only detect individuals having heights beyond a specified threshold. After testing sessions, an adequate configuration was found to provide accurate results for the environment found at the shop.

After the CCount front-end page is successfully loaded, users are presented with two text fields, the former indicating the selected day of interest and the latter the camera scene, and important information regarding the total amount of people recorded. In Figure 52 the camera scene with the corresponding heatmap is presented, providing a hot-spot analysis of specific zones. Further, a time slider allows the user to specify a start-end range and in turn filter the coordinates and update the other measurements.

From the composed chart presented in Figure 53 it is possible to observe that the total amount of visitors to the outdoor department remains constant during weekdays. On weekends, more specifically on Saturdays, the camera was able to record up to double the number of customers that usually visit the shop on weekdays. The overall average dwell time shows a slight increase. A similar picture is also given from the daily dwell time summary, illustrated in Figure 54.

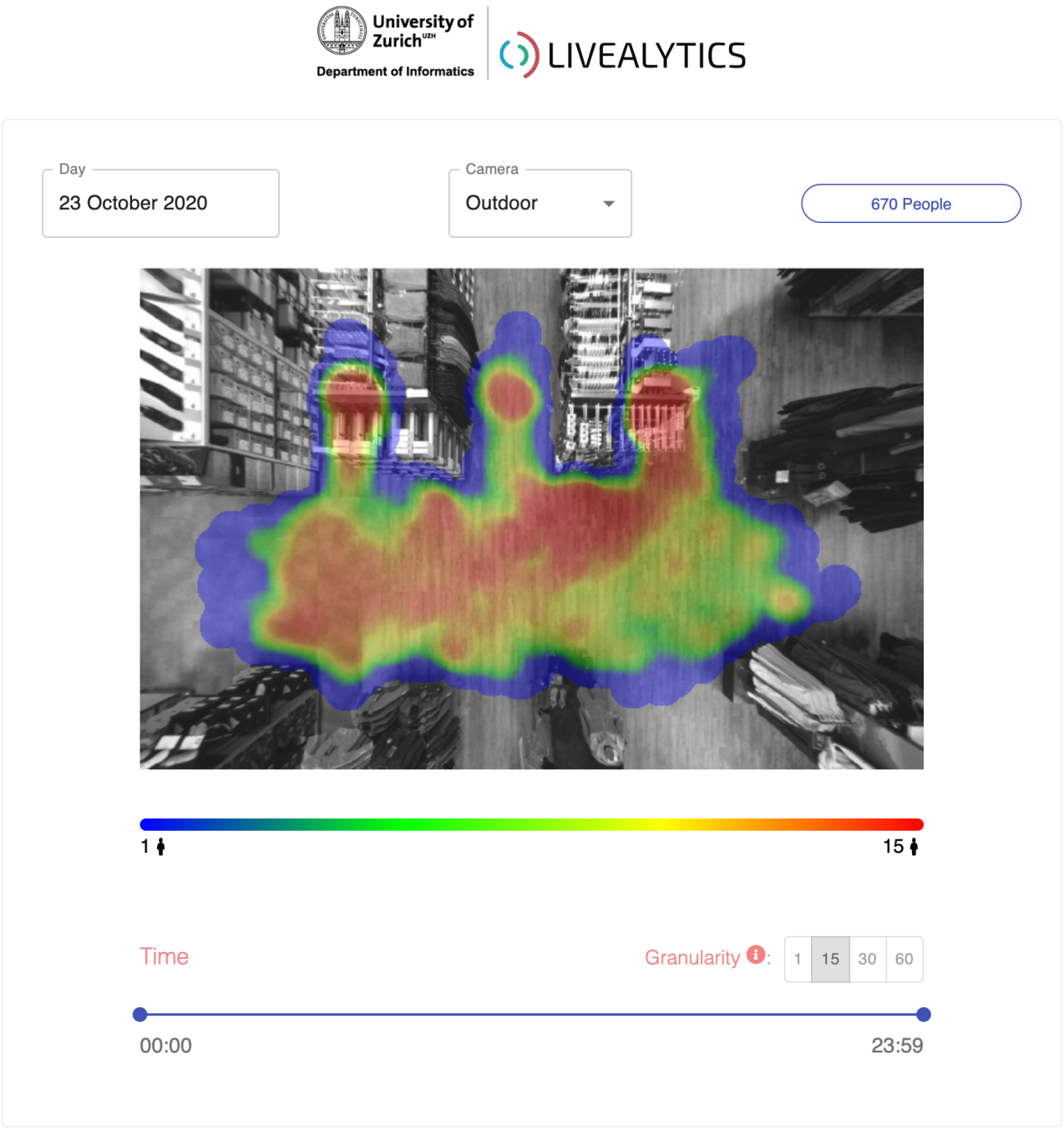

Figure 52: Heat Map/Path Visualizations Based on Camera Data

Another important functionality that CCount provides is interaction analysis and distance calculation. For this specific real-world test scenario, we had the opportunity to put the behavior and scalability of our distance calculation algorithm to the test. Table 55 shows a list of five interactions that took place on October 23rd, 2020 having a minimum duration filter of five seconds. Lastly, the list is sorted by the distance in descending order.

It can be seen from Table 55 that the highest distance is 0.56m, approximately a third of the necessary distance (1.5m) required for ensuring social distancing. For a detailed look

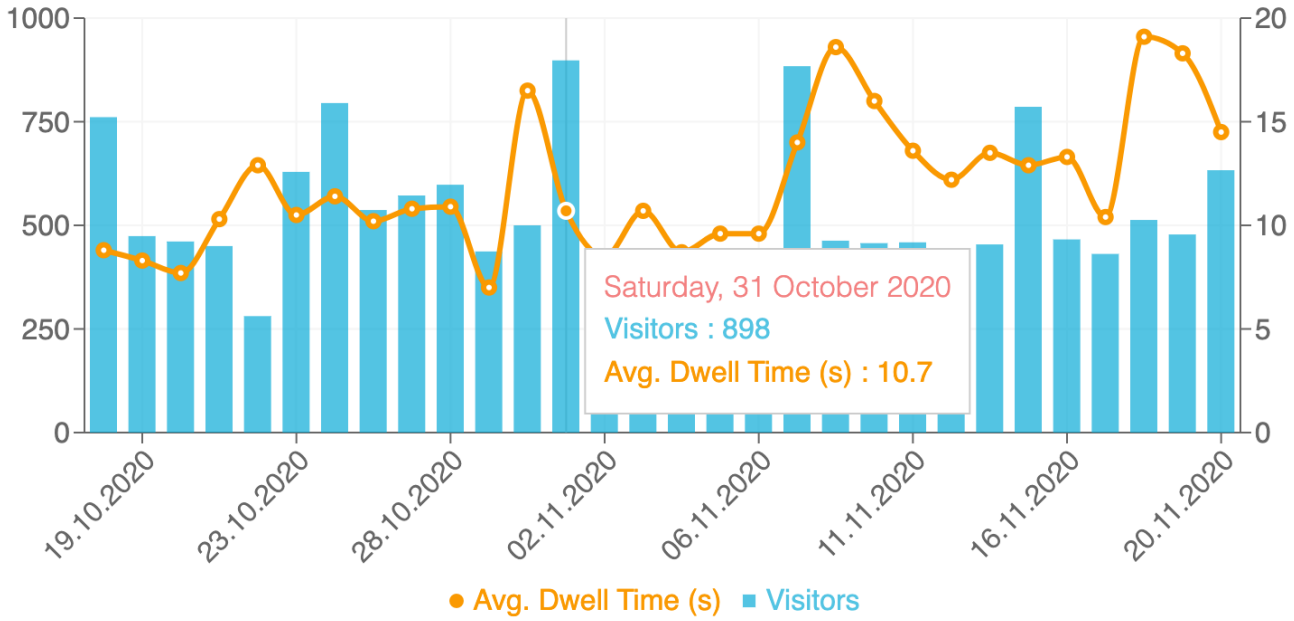

Figure 53: Outdoor Department Dwell Time History

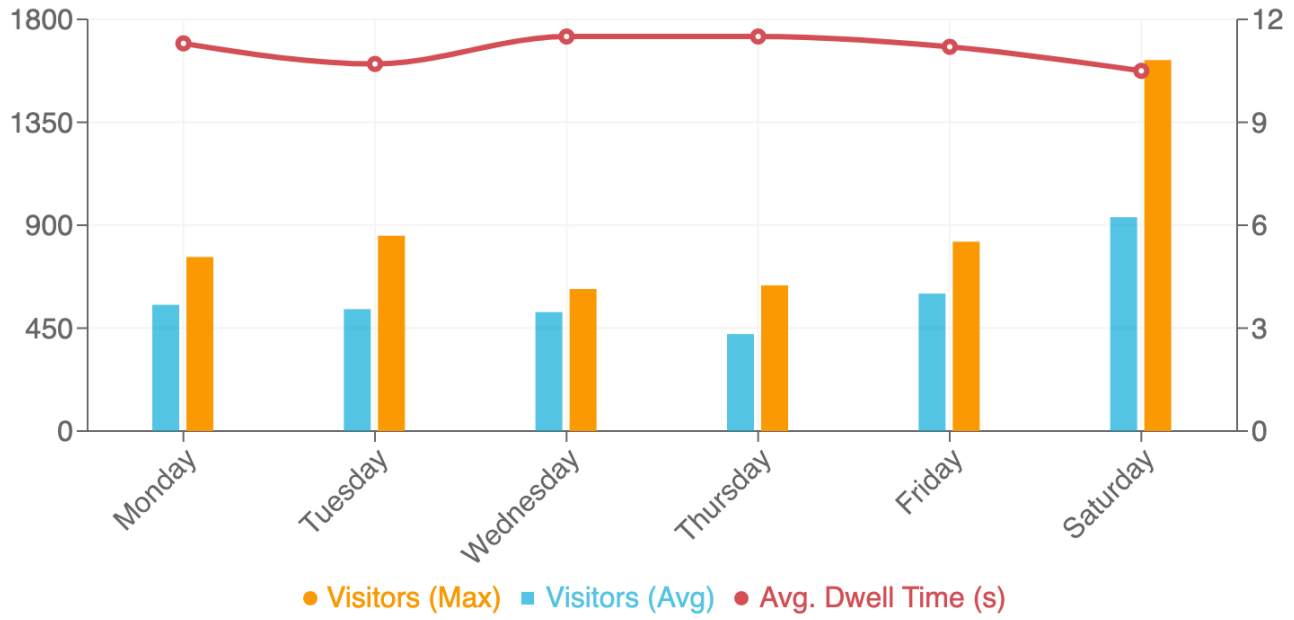

Figure 54: Outdoor Department Dwell Time Summary (Daily)

at the selected day, CCount front-end also provides a daily summary of the interactions, including the total number of interactions along with the average distance, as shown in Figure 56. From this composed chart we can observe that the average distance reaches its peak at around 12:00 pm whereas the interactions greatly increase in the afternoon. CCount offers the possibility to conduct this analysis for each day, given that data was previously recorded and processed by the back-end.

Min. Duration Filter: 5 15 30 60

Day	Start	End	Duration (s)	Distance (m)
23-10-2020	14:24:05	14:24:10	5.0	0.56
23-10-2020	14:22:03	14:22:08	5.0	0.54
23-10-2020	14:24:05	14:24:10	5.0	0.52
23-10-2020	17:58:47	17:58:52	5.0	0.29
23-10-2020	17:01:47	17:01:52	5.0	0.19

5 rows 1-5 of 10

Figure 55: Outdoor Department Interactions

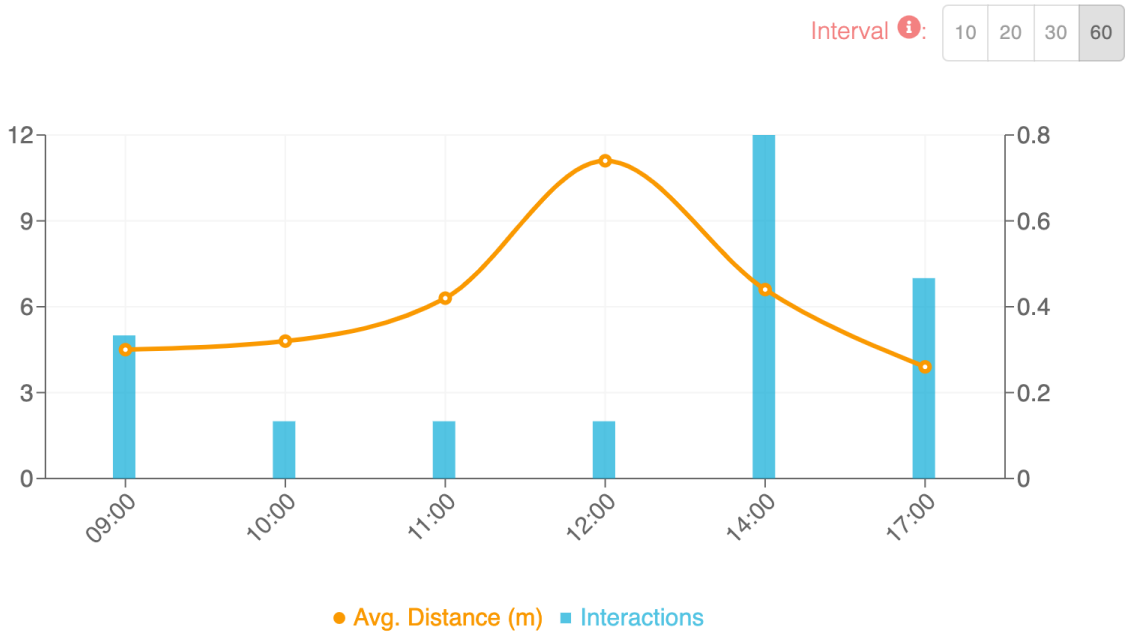

Figure 56: Outdoor Department Interactions Summary (Daily)

RFID and Camera A comparison of different RFID reader (Impinj xArray R680 reader) configurations (*cf.* Table 10) demonstrates the results from a practical example. Selecting the adequate RFID reader mode based on the environment is key to accurately calculating the tags' location. The test consisted of fourteen passive UHF RFID tags arranged in dif-

ferent locations within a warehouse and an RFID reader, and a 3D camera mounted on the ceiling. After running each mode for a minute, the collected data was analyzed.

Table 10: Average Count of Entries and Number of Electronic Product Codes (EPC) Found for Each Reader Mode

Algorithm	Average Count	Number of EPCs found
Auto Set Custom	59.538462	13
Auto Set Dense Reader Deepscan	62.153846	13
Auto Static Dense Reader	62.384615	13
Auto Static Fast	60.692308	13
Dense reader M4	60.769231	13
Dense reader M8	61.384615	13
Hybrid M2	61.307692	13
Max Miller	60.076923	13
Max Throughput	61.461538	13

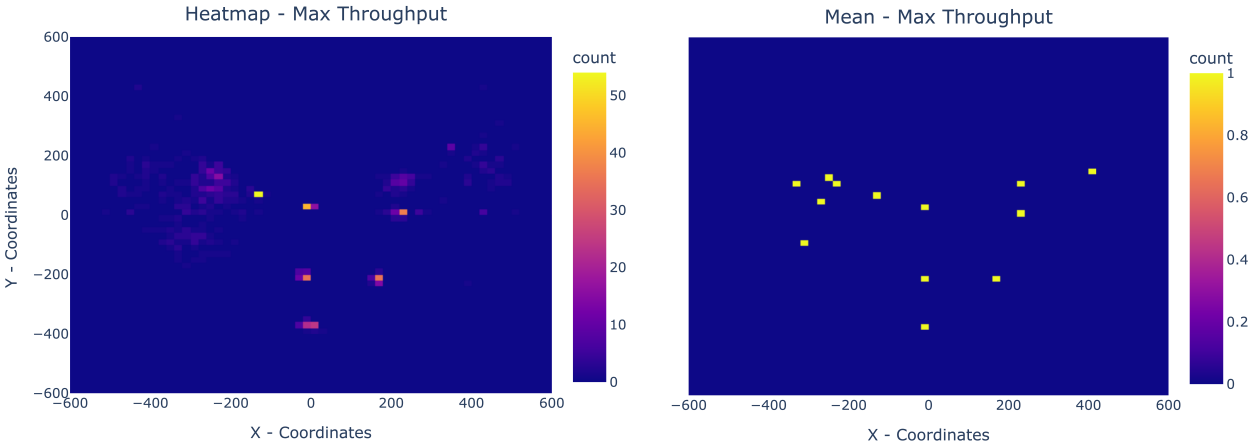

Figure 57: Heat Map of Max (Left) and Mean (Right) Throughput

Despite the use of multiple reader modes, 13 tags out of 14 were captured, *i.e.,* with a success rate of *92.9%*. The “missing” tag was not isolated from the others; instead, it was placed in a dispersed location where the signal could not be successfully captured. As part of the matching algorithm between 3D cameras and RFID tags, a critical metric is the number of times a particular RFID tag can be captured, *i.e.,* how many data points a reader can deliver within a certain period. As a result of changing several parameters in these configurations, the different RFID reader modes delivered approximately 60 entries within 1 min, which corresponded to the granularity of a 3D camera of one data point per second. Thus, the accuracy of the various algorithms can be compared.

Figure 57 (left) represents the heat map for RFID tags captured within a minute, whereas the right graph shows the approximate position of data points captured within this minute for each RFID tag. In regions where tags are more distant and the density is low, approximated positions are similar across the maps. However, the RFID reader performs poorly when

tags are clustered together closely, resulting in higher density. As a result, the heat map is unclear, and it is impossible to distinguish between the various tags. This behavior results in a large amount of noise in the data, making it more difficult for the matching algorithm to correctly identify individuals wearing an RFID tag as they appear to be moving.

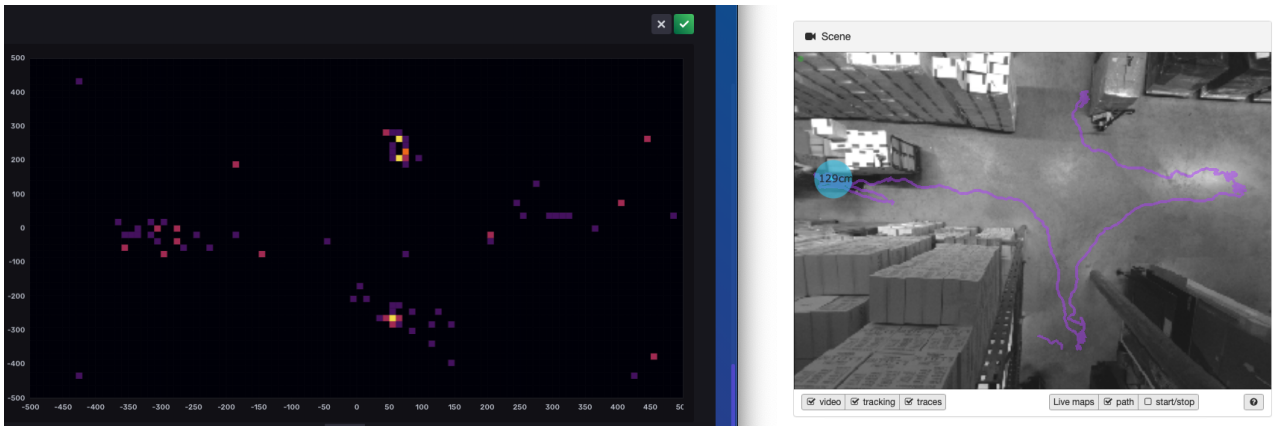

Figure 58: Recording Frame Used to Correlate RFID Readings. Warehouse in Schlieren, Zürich.

Figure 58 shows a snapshot of a recorded video, which was used as the focal point on the path where participants walked carrying RFID tags. Thus, it was possible to correlate RFID readings outside the camera's focal point with objects detected by the camera to increase the accuracy of the track. Moreover, this setup successfully assessed the system's functionality in real time.

Single RFID Tag This scenario evaluates the accuracy and precision of CCount, when tracking a single individual wearing an RFID tag under the scene; *cf.* Figure 60 (top) for the correct path of a single individual wearing this tag. The algorithm identifies the individual with 100% accuracy when the RFID reader regularly captures the RFID tag's position. The success rate of the correlation was higher than 70-80%. Figure 60 (top) shows the correct path of the single individual wearing the RFID tag.

Since the CCount system assumes that each individual is assigned a personal and unique RFID tag, the scenario considering people randomly walking but only a single individual carrying a tag should never occur. However, it was conducted to test the algorithm with arbitrary noise in the data. Several randomly walked under the camera, but only one carried an RFID tag. Since these individuals were moving in different directions, thus, making it easier to find a suitable correlation for the algorithm, the tag was successfully assigned to the correct individual. A false-positive rate of 0% was observed.

Multiple RFID Tags A group of individuals wearing RFID tags in a crowded environment was tracked to assess the system's accuracy and precision. The results indicated that the algorithm barely managed to assign tags correctly. While every individual in this variant wore an RFID tag, only one individual was walking. It has been previously demonstrated that, with multiple tags, the inaccuracy of the RFID reader increases substantially, making it more challenging to match the ID of the 3D camera with the RFID tags.

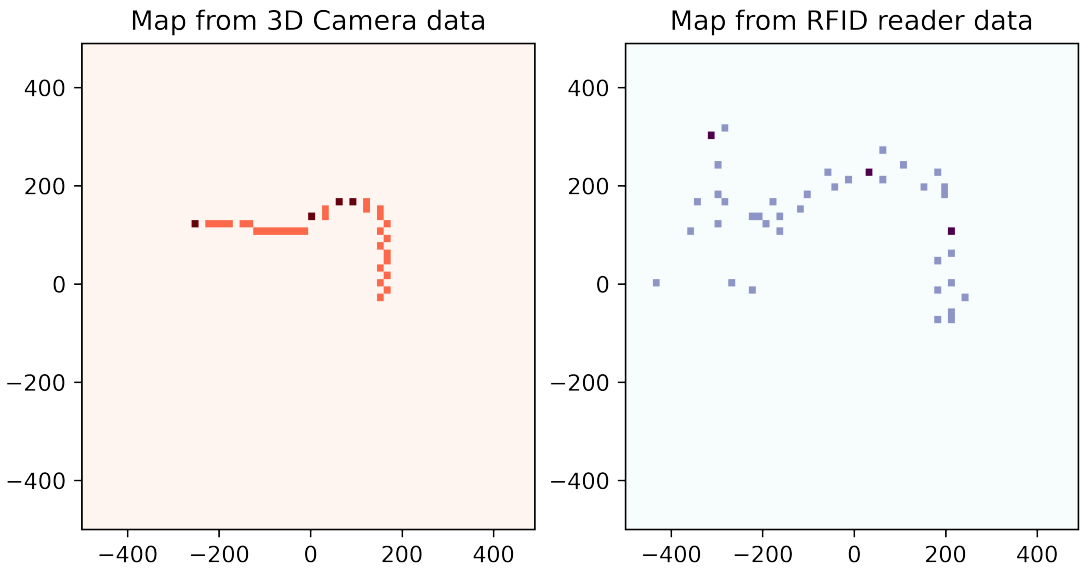

Figure 59: Maps From a 3D Camera and an RFID Reader of a Single Person Wearing an RFID Tag Walking Between a Group of People.

However, on selected occasions, the algorithm successfully found a match despite the noise. Figure 60 shows a path taken that was correctly assigned. As seen, noise is present in the data coming from the RFID reader. As for the previous scenario, a case considering individuals wearing an RFID tag and standing still did not yield precise results. The algorithm is designed to take only the correlation into account, which resulted in erroneous results as shown in Figure 60 (bottom). A different metric, *e.g.,* the *Euclidian distance*, can prevent this wrong assignment.

Visitor Counts: 3D Camera vs. RFID Reader Data collected during these experiments can be analyzed independently from each other and compared. In order to conduct this comparison, visitor counts are grouped according to 1-min intervals (*cf.* Figure 62).

In contrast to the 3D camera, which assigns different and random IDs to people entering and leaving the scene, RFID tags are always uniquely identifiable by the RFID reader, due to their EPC. Thus, data collected from RFID readers can determine the unique number of visitors present at any given time. In this scenario, two participants were physically present. Still, the 3D camera assigned up to 13 different identities within five minutes (when using a grouping based on 5 min intervals, different IDs were found to be 35). Note that the number of RFID tags did not exceed the actual count of RFID tags present, *i.e.,* 14.

In the period of *10:15-11:05 hours*, these experiments were performed using two different RFID tags. Participants entered and left the camera's field of view constantly, resulting in a large discrepancy between the two measurements. In the period of *11:10-11:20 hours*, different reader modes of the RFID reader were evaluated. In this scenario, no participants were present at the scene. Instead, RFID tags were statically placed under the camera. Therefore, the camera's count equals 0, whereas the RFID count is almost constant at 13. Consequently, the number of visitor counts taken from the RFID reader's data can be inter-

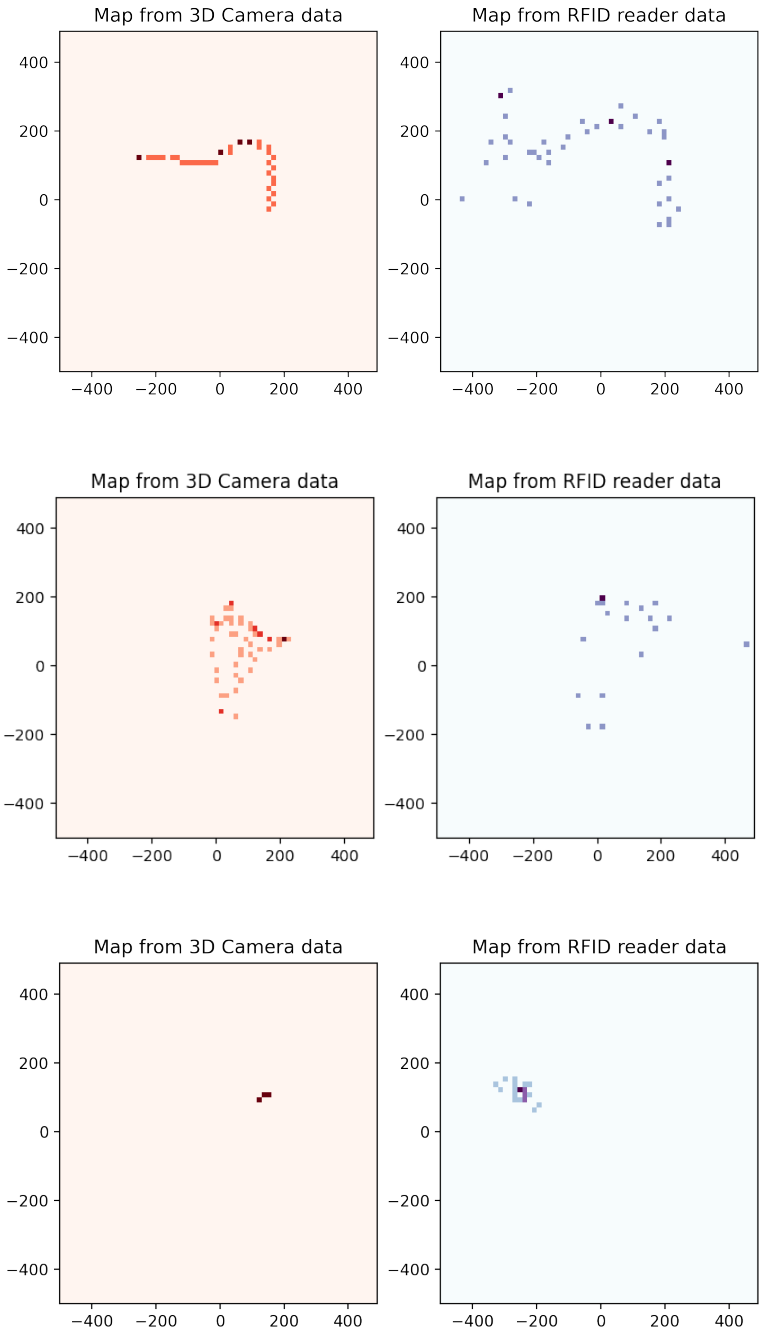

Figure 60: Top: Single Person Walking; Center: Single Person Walking Between a Group (Multiple Tags); Bottom: Multiple Persons (Tags) Standing Still in a Group

preted as a valid number.

Data Visualization The KPIs defined in the section above are incorporated into suitable visualization and displayed in the dashboard. A range of filters is embedded to facilitate the analytical use of the board. Figure 61 shows how the dashboard changes with different

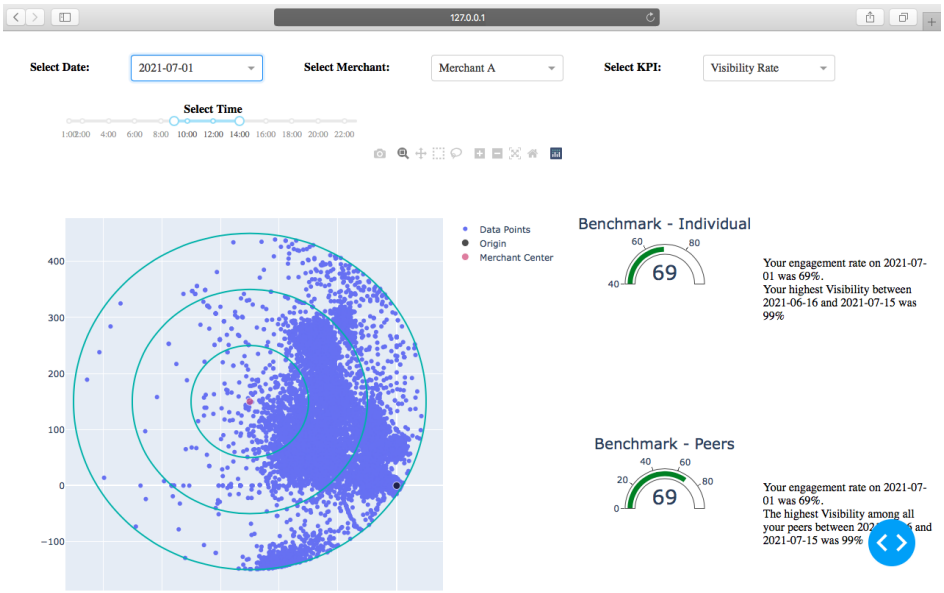

Figure 61: Dashboard Centered as Merchant A on July 1 Between 09.00 - 14.00 (A) Visibility Rate (B) Engagement Rate (C) Interaction Rate

filters.

Marketing campaign statistics for merchant A on July 1, 2021, are shown in Figure 61. It is clear that merchant A has a visibility rate of 69%, an engagement rate of 45%, and an interaction rate of 15%. Among the 69% of opportunities that appeared also in the visit zone, less than half of them converts to a visit, and less than a fifth of those visitors interacted with merchant representatives. Moreover, one can see that the dials used for benchmarking allow a quick assessment of the KPI in the broader context. For merchant A, the interaction rate is clearly below its own and peer group averages, while the visibility and engagement rates are both at or slightly above average.

As the web prototype presented can be seen as an information-oriented website, the Web Usability Index can be used for evaluating the usability of this web application [41]. This evaluation will go through the Index's five categories checklist because the Web Usability Index consists of a checklist with as many as 150 questions [22].

Starting from the first category: "Navigation and Orientation: consistency of navigation, the color of links, etc". The author believes the web prototype has consistent navigation. It is clear that the dashboard has all the filter options located on the top panel of the website. Three filter options are presented as a drop-down menu and a one-time frame filter choice is done with a timeline plus two adjustable handles.

In terms of "Interaction and Information-Exchange: availability of a homepage, skip functionality for intros, etc", the web prototype does encourage users to engage on the dashboard by providing a responsive display depending on the choice of filters a user has made despite the dashboard having only one page.

For the categories "Being up-to-date and Quality: marking of texts with author and date, absence of spelling mistakes, etc" and " Ease of Access and Accessibility: the connection between URL and website, availability of high- and low-tech variants of the website, etc", the prototype has been reviewed by the friend of the author. He has confirmed that the website

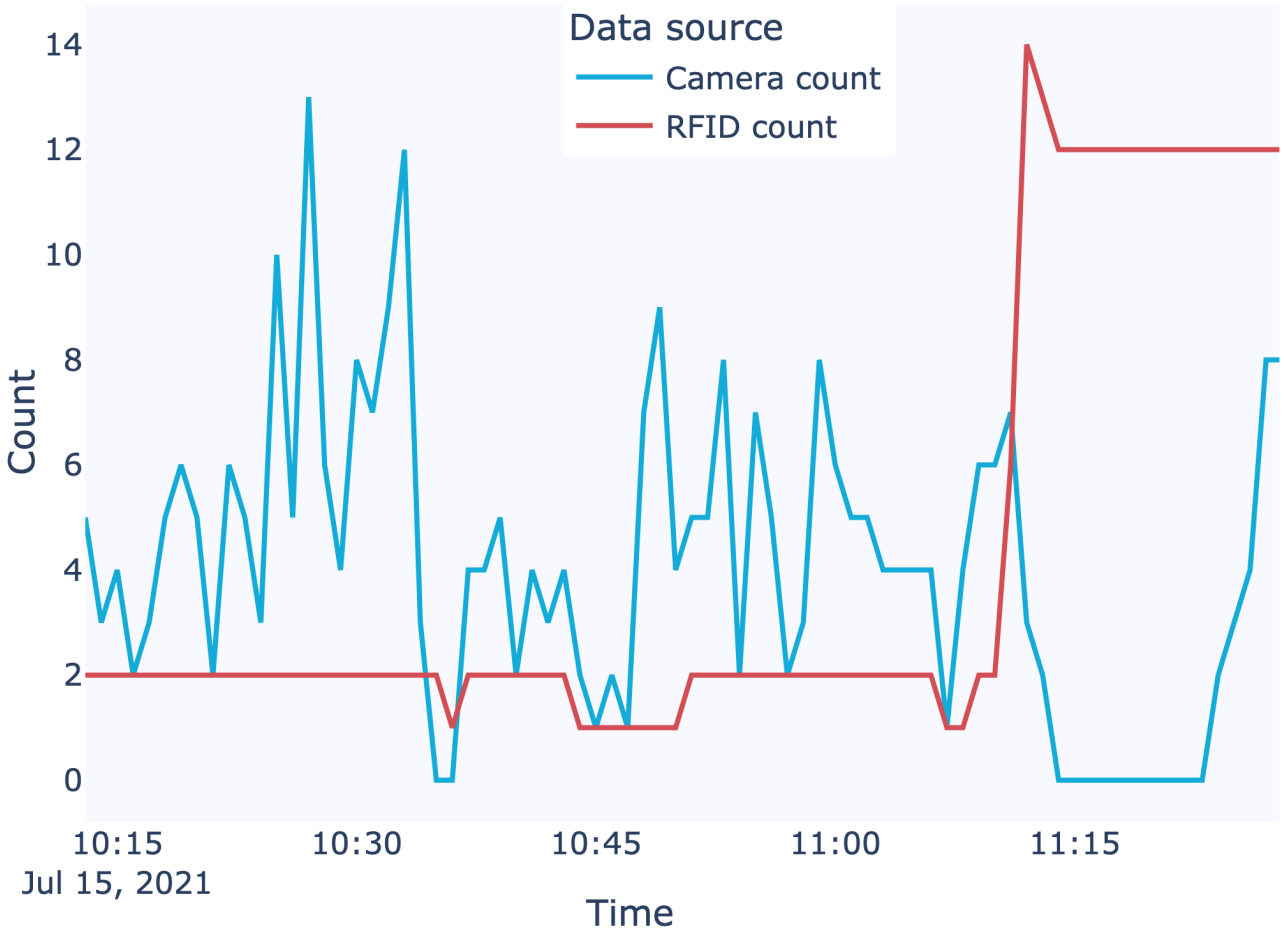

Figure 62: Visitor Count From 3D Cameras and RFID Readers

is available on the *localhost* and free from context errors.

Coming down to the last category on the list ”Information- and Text-Design: the size of the font, expressiveness of icons, etc”, the three visual charts are intentional choices of the author. The zone of interest scatter plot on the left side aims to present the overview of the crowd during the selected time range. The two dial plots provide prompt comparisons across merchants and/or over time. The text next to the dial is an aid for the users and provides a written explanation and the exact analytical numbers.

5.2 Fuslon Data Tracking System (FITS)

FITS was evaluated based on four different scenarios. As specified in Table 11, different sensor types, sensor amounts, and the number of data points were defined to evaluate the overall FITS performance. Accuracy was measured by clustering correctness, position prediction error, accuracy, precision, and the overall percentage of correctly tracked objects.

The test data input specifies the virtual measurement setup of the synthetic measurement generation. This includes the specification of sensor positions $(\mathbf{x_s}, \mathbf{y_s}, \mathbf{z_s})$ & orientations $(\alpha_\mathbf{s}, \beta_\mathbf{s}, \gamma_\mathbf{s})$ as coordinate systems relative to the absolute frame of reference as well as the

Table 11: Evaluation Test Cases

ID/Description	# Sensors	Sensor ID(s)	M. Quality	M. Points	Quality Aspect
1 - Base case	1	1	High	20,000	1
2 - Dense sensors	5	1,2,3	High	4 x 5,000	2
3 - Low M. quality	1	4	Low	20,000	3
4 - Sparse data	1	1	High	500	4

definition of sensor parameters, namely sensor spatial reach (λ_s), temporal resolution (γ_s), and the sensor's spatial precision (μ_s). Additionally, the specification of a sensor type is necessary to be able to analyze performance using this parameter. Optionally, variables, such as sensor identifier format (default: *UUID4*) and stability functions (default: linear), can be defined.

As depicted in Table 12, the "ID/Description" gives an identifier for the test case, "# Sensors" defines the number of sensors used in this test case, "Sensor ID(s)" refers to sensor IDs and parameters, "M. quality" determines the quality of measurements (*e.g.,* sensors with low error provide high quality), "M. points" specifies the number of true data points taken into consideration, and "Quality Aspect" refers to the respective aspect of the four quality aspects as defined above.

The characteristics of the virtual sensor used are listed in Table 12 and chosen based on real-world expectations for these sensor types. The sensor positioning can be seen in Figure 63 and was selected to mirror a real-world setup, *i.e.,* sensors spread among the location, but generally placed, where most people can be expected.

Table 12: Configuration of Virtual Sensors

ID	Type	Spatial Precision	Spatial Reach	Temporal Resolution
1	WiFi 2.4 GHz	60 cm [17]	20,000 cm [17]	150 ms [81]
2	RFID	20 cm [27]	100 cm [80]	20 ms [27]
3	Camera	40 cm	300 cm [85]	17 ms [85]
4	(low quality) BLE	100 cm [58]	1,700 cm [14]	2'56 ms [58]

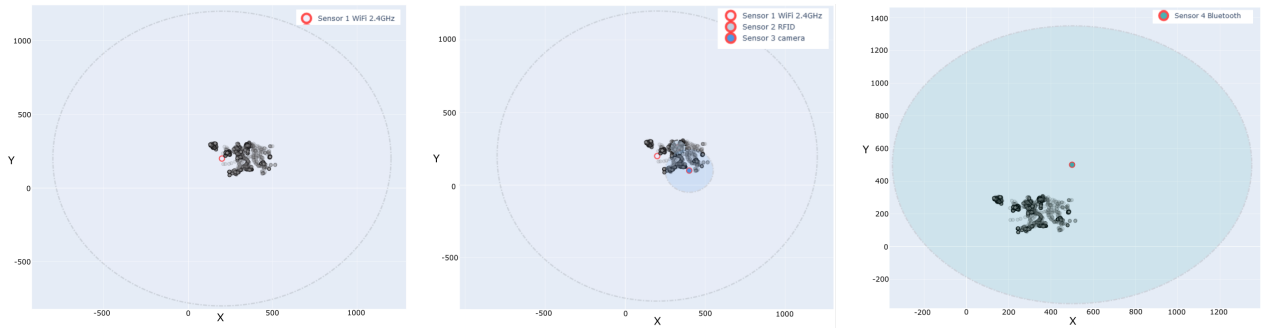

Figure 63: Evaluation Test Cases' Virtual Sensor Placement. X and Y Axis in [cm]. Left: Test Cases 1 and 4. Center: Test Case 2. Right: Test Case 3.

Figure 63 depicts the sensor placement representing test cases: *(a)* Left represents test

cases 1 and 4, *(b)* center shows the setup for test case 2, and *(c)* right highlights the sensor for test case 3. These colored points indicate the individual sensors according to their IDs; dotted circles around them define their measurement reach. Gray points determine the aggregated, simulated measurement points and indicate, where people can be expected. As the FITS performs both object fusion/clustering and position prediction, these metrics are analyzed individually and in a combined manner. To ensure that all effects observed are representative and not only a statistical anomaly, each test case is also executed three times, and the respective values are reported as a mean, including their standard deviation.

Object Fusion Evaluation Results A single real-life visitor may be captured by multiple sensors, each assigning a different unique ID to the object. Object ID clustering detects similarities between objects recorded by different sensors and tries to link them together in object clusters to recover the real-life visitor structure. The fusion algorithm minimizes the pairwise Euclidean distance within these objects of a cluster.

Object fusion quality was evaluated using both *naïve* and *bcubed* measures. The results achieved show that for all metrics (Naïve, bcubed recall, bcubed precision, and bcubed fscore) and test cases (1 to 4) these metrics achieved a result of 1.0 with a standard deviation of 0.0, indicating that individuals that were part of the true dataset were correctly identified for every sensor measurement point. These metrics calculated indicate how prominent the overlap between the clusters found (*i.e.,* the identified individuals) and the true individuals. Metrics can take values between 0.0 and 1.0, whereas 0.0 would indicate a total mismatch between found and actual clusters, and 1.0 would indicate a perfect replication of the provided initial clusters.

These results show that the model is able to correctly identify the clusters (*i.e.,* individuals) that were present in the original dataset. This is achieved with full recall and precision. For test cases with one sensor only (1, 3, and 4), this is intuitive since here the individuals can directly be identified by their ID assuming, as here in this case, that the ID is constant for any one-to-one interaction between a device and a sensor (*i.e.,* no ID randomization takes place when a device is already connected to a specific sensor). For test case 2, the model is also able to correctly identify all clusters. This could be potentially not only related to the good model performance, but also due to the use of real-world movement data, where there might not have been many overlaps for this specific dataset slice and, therefore, no ambiguous decisions for the model were made.

To verify that this behavior is not just an artifact of a model or evaluation error, the authors verified that the clustering performance indeed drops, when a different ID randomization method is used (*e.g.,* randomization of the ID, when a device is already connected to a specific sensor). An additional factor impacting these results is the up-sampling approach: increasing the number of measurements that can be mapped to certain individuals increases the overall accuracy.

Position Prediction Accuracy Results The position prediction quality was evaluated using different naïve metrics. Those results are shown in Table 13. The metrics, all reported in centimeters (cm), can be interpreted as follows: “Naïve Total” defines the sum of errors over individuals predictions; the lower, the better. However, values are comparable, when comparing test cases with the same amount of sensors and measurement points (test cases 1

Table 13: Results for Position Prediction using Naïve Metrics [cm]

Test ID	Naïve Total	Naïve Mean	Naïve Min	Naïve Max	Naïve Median
1	375,431 (1,415)	25.7 (0.1)	1.2 (0.2)	481.8 (15.2)	21.3 (0.2)
2	124,544 (1,023)	19.4 (0.2)	0.9 (0.3)	447.7 (18.0)	12.6 (0.5)
3	539,997 (6,162)	38.0 (0.3)	1.2 (0.6)	466.0 (76.2)	33.2 (0.3)
4	8,466 (393)	24.9 (1.2)	2.4 (1.0)	77.5 (20.0)	23.0 (0.7)

Table 14: Combined Quality Evaluation (*cf.* Table 3, Stripped of Up-scaling Data)

Test ID	MOTA [%]	MOTP	GT	MT [%]	PT[%]	ML[%]
1	91.8 (0.1)	773.1 (9.1)	237.0 (0.0)	68.9 (0.2)	12.8 (0.2)	18.3 (0.2)
2	96.7 (0.2)	619.4 (15.3)	56.0 (0.0)	66.7 (1.0)	24.4 (1.0)	8.9 (0.0)
3	94.9 (0.1)	1,549.9 (28.8)	246.7 (3.5)	63.4 (0.6)	17.4 (0.4)	19.2 (1.4)
4	99.1 (0.0)	790.1 (65.9)	4.0 (0.0)	100.0 (0.0)	0.0 (0.0)	0.0 (0.0)

and 3). “Naïve Mean”, “Naïve min”, “Naïve max”, and “Naïve Median” are standard statistical parameters applied to the prediction error; for each of them, the model is more accurate, if the value is lower.

As the goal was to propose a model that is able to estimate the indoor positions precisely in all cases, the mean metric is evaluated with the highest priority. The mean metric decreases when the sensor count increases (test case 2 vs. 1), which indicates that it improves its prediction accuracy by using multi-object tracking. Also, using lower quality sensors (*i.e.,* have a higher inherent measurement error) produces higher prediction errors (test case 3 vs. 1). This is seen in the total error reported, higher for case 3 than the base case. Lastly, the prediction error roughly stays the same, 24.9 (test case 4) compared to 25.7 (base case), regardless of whether 20,000 or 5,000 measurement points are used. However, the case with sparse data has a higher standard deviation, indicating that these predictions are not as accurate (although acceptable).

Combined quality was evaluated using different CLEAR-MOT and related metrics. The respective results are shown in Table 14. The CLEAR-MOT threshold was set at 100 cm, considering an individual correctly identified, if they identified a point within one meter of the individual’s actual location. The one-meter diameter was defined since the authors assumed that this would be the maximum number of devices placed around the individual’s center. Predictions with errors above this threshold are not considered “correctly tracked”.

MOTA is provided in %, in which the higher the percentage, the more accurate the model’s predictions are. MOTP is a precision indicator given in cm. A lower metric shows that the model’s predictions were precise. Lastly, MT (“Mostly Tracked”) indicates the percentage amount of “Ground Truth objects” (GT) that were correctly tracked in more than 80% of the measurement timestamps. A higher value for MT indicates that more objects are tracked correctly, and therefore, the model’s predictions are better. Similarly, PT (“partially tracked objects”) in percent (*i.e.,* object identified correctly between 60% and 20% of the time) and ML defines the percentage of “mostly lost objects” (*i.e.,* identification in less than 20% of the cases correct).

Performance Evaluation FITS' performance was evaluated to assess whether near real-time processing is feasible (*i.e.,* considering a run time of 15 minutes for 100,000 measurements). The performance analysis consists of increasing load tests to estimate the system's run time concerning the input size (*i.e.,* big O notation). The evaluation was performed on an Intel Core i7-9750H processor and 32 GB of RAM. Both, the FITS API and the Python script, were run in a single docker container to avoid the effects of the network. Table 15 shows the mean run time in seconds for each measurement with size *N* (number of data points) and the number of iterations (*i.e.,* number of times the experiment was repeated).

Table 15: Run Time Analysis

N	**Mean Run Time [s]**	**No. Iterations**
100	0.52	10
200	1.08	10
400	1.60	10
800	2.64	10
1,600	5.76	10
3,200	10.06	10
6,400	22.04	10
12,800	39.75	10
25,600	84.17	10
51,200	172.17	10
55,000	187.41	10
60,000	202.92	10
65,000	224.13	10
70,000	236.83	10
75,000	254.83	10
80,000	773.74	1
100,000	967.187	1

A dataset with 100,000 measurements and 1,000 visitors was created using the synthetic data generator for evaluation. The Python script posts these measurement data to the REST API of FITS and measures the run time until the input batch job computation has finished. Benchmarks with increasing input size (100, 20,000, ..., 80,000, 100,000) were performed subsequently. Each benchmark doubled in input size and was run 10 times to compute the average run time.

Figure 64 shows that for an input size up to approximately 80,000 measurements, the processing time is in the order of *O(n)*. However, there is a tipping point at 80,000 measurements, where the run time exceeds the linear bounds and increases drastically. It is important to note that the system was tested as a whole and not only the developed tracking algorithm. Thus, the performance of individual components (*e.g.,* read/write speed of InfluxDB) do have a significant effect on the overall performance, which was not yet optimized.

Especially, experiments that draw the attention are those at 80,000 and 100,000 data points, in which the increase of the average run-time time grows significantly, but not exponentially. This turning point happens due to excessive memory usage by InfluxDB, in which the operating system starts swapping data to the hard drive, significantly increasing the ex-

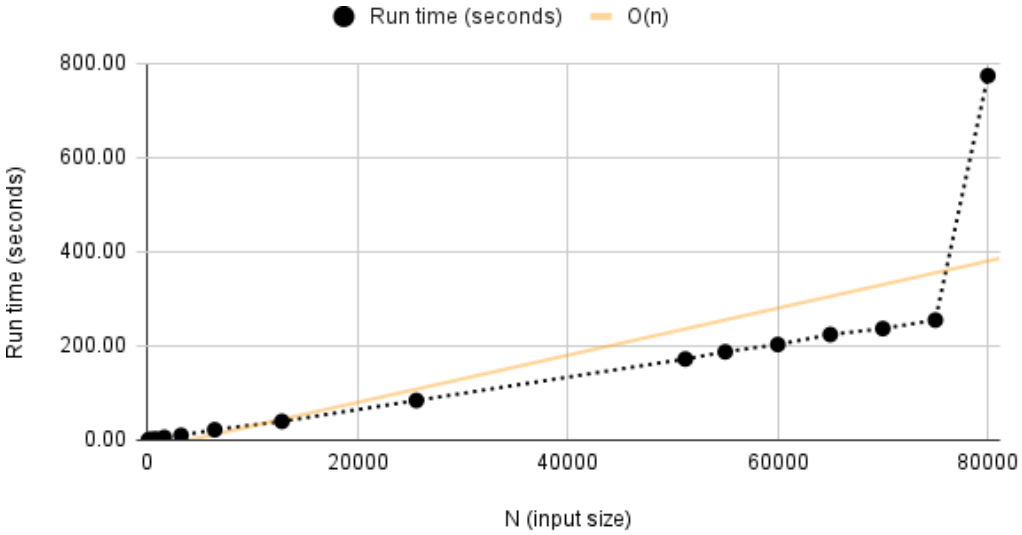

Figure 64: Run Time Analysis

ecution time. Such bottlenecks can be mitigated by dividing the measurements produced by the Synthetic Generator into two smaller datasets (*e.g.,* two datasets of 50,000 data points). This indicates that improvements in the execution pipeline can be made to distribute the storage and processing of data (*e.g.,* by using a distributed streaming platform such as Apache Kafka).

Discussion of Results Taking *test case 1* as the baseline, this shows that even one sensor suffices to achieve a good prediction, *i.e.,* achieving more than 50% "mostly tracked" and a mean prediction error of less than the sensors' spatial precision. The correlation of objects was able to correctly identify individuals for every measurement point in test case 1. Once only one type of sensor (Wi-Fi 2.4 GHz) is used in this test case, a perfect mapping between an ID and the true object can be generated. Thus, the up-sampling allows for the easier artificial fusion of true and predicted objects. This result is confirmed by analyzing the contingency matrix generated of relevant object IDs predicted with the true occupant IDs, which is of square diagonal format (assuming an ordering of columns). Although the sensor itself has a spatial precision of 60 cm, the results achieved, based on the naïve mean, resulted in an average deviation of 25.7 cm. Lastly, those combined evaluation shows an MOT accuracy of approximately 92%, with more than two-thirds (68.9%) of these objects being tracked correctly in more than 80% of the cases. This can also be considered a highly viable outcome.

Considering the measurement of additional sensors (*test case 2*) those results show a clear indication that the FITS engine achieves the goal set, *i.e.,* improving predictions' accuracy by taking into account multi-object tracking. This is proven by not only the decreased average naïve position error (6.3 percentage points lower than base case), but also through the higher MOT accuracy achieved. While the "mostly tracked" objects roughly stay the same, the multi-object tracking also achieves significantly lower "mostly lost" objects (9.4 percentage points lower than the base case), indicating that not only for easy to track objects these predictions can be improved but also for previously harder to track objects. Summarizing those results of test case 2: They show that the second quality aspect, namely increasing

the prediction quality, when considering more sensors, is reached.

The results of *test case 3* show that there is a measurable impact, as expected, on the prediction quality, if lower quality measurements are provided. This can be seen in both the naïve position prediction mean metrics (12.3 percentage points higher than base case) and in the "mostly tracked" objects (5.5 percentage points higher than base case). Especially dominant is the effect of the higher spatial sensor measurement error, when analyzing MOT precision, where the value almost doubled and indicated, therefore, a low precision. Lastly, *test case 4*, where only 500 data points instead of 20'000 were taken into consideration, shows that the statistical values reached are in general similar to the base case.

General Implications and Limitations Fusing data from different sensors and manufacturers implies the need to harmonize the data before its fusion. In this sense, FITS defines a minimal data structure so that data received in different temporal and spatial resolutions can be transformed into a uniform and comparable view. While this pre-processing stage implies a higher overhead in data preparation, it also ensures that different correlation and machine-learning models can be used. Furthermore, it contributes to the explainability of the approach since they operate on uniform data. Thus, a practical limitation of the approach proposed here is the need to adapt the pre-processing stage to the sensors used in order to extract the minimal data defined in the framework.

The use of APIs for both receiving sensor data and communicating with the front-end makes the approach used in FITS independent of the synthetic data generator. It is possible for data to be sent directly through the API once the pre-processing component is adapted to transform the data accordingly. However, regarding the synthetic load generator, a limitation is related to the creation of possible anomalies typically observed in real environments (*e.g.,* signal irregularities due to real-world barriers or walls), which might have a significant impact and lessen the overall data quality in a real-world application. Therefore, it is probable, that the synthetic data generation approach might not be able to entirely replicate the real-world aspects.

FITS complements the state-of-the-art by presenting a structured approach to data fusion, complementary to recent approaches based on ML to predict data patterns based on data from distinct sensors [1, 4, 70]. More specifically, different ML approaches can be incorporated into step 6 and within the evaluation presented in the FITS architecture (*cf.* Figure 30).

While there are engineering aspects that could be improved on within FITS, the main areas of improvement are the individual components of the processing pipeline. The *Up-sampling* component does not support a configurable time resolution for the up-sampling rate as opposed to the currently hard-coded value of 1 second. Further, the Kalman Filter configuration in the *Smoothing* component could be obtained empirically to model the real-world dynamics more precisely.

Another important implication is related to privacy since FITS can be used for both active and passive tracking. While active approaches typically rely on the user's consent, passive approaches do not require consent and may characterize a violation of privacy. In this sense, the use of FITS in public environments *e.g.,* public transportation, should be carefully planned to avoid users identified by correlating wireless signals and cameras.

5.3 Data Consumption

The Web interface was evaluated considering the System Usability Scale (SUS). System Usability Scale [6] is a user-oriented testing strategy for measuring the usability of a certain system. With this testing approach, participants are asked to score different questions with one of five responses that range from Strongly Agree to Strongly disagree. The questionnaire and the corresponding scoring system are outlined in the System Usability Scale Template [6].

Due to the fact that the platform at the current stage can still be considered a minimum viable product for early customers, the SUS approach has been chosen, since it can be used on small sample sizes and obtain reliable results. It is worth pointing out, that the usability of any software has to be considered in terms of the context in which it is used and therefore its appropriateness has to be viewed in that context [6].

In order to obtain sensible results for the SUS test, ten people have been interviewed with different backgrounds and skill sets. Participants have been divided into three different groups based on the following criteria:

1. *Developers*
   People who are technically knowledgeable and have a computer science background. Developers have been selected from Axelra [5], a tech venture builder based in Zurich. During the development of the platform, they have been involved in discussions regarding potential solutions and were aware of the progress of the system.

2. *Promoters*
   People working for different brands and agencies who are part of their promotional teams. These people have a good understanding of how to setup and run a promotional activity or a marketing campaign. They have never worked before with the platform implemented and have no technical background.

3. *Brands and Agencies Managers*
   People at the head of different brands and agencies who are responsible for the outcome of their marketing campaigns or promotional activities. These people are usually not part of the promotional team and are mainly involved in high-level marketing strategies.

The SUS experiment has been executed after the first official release in November 2019. The results are described in the section below and can be found in Figure 65.

SUS Results and Personal Feedback In general, all groups considered the system well integrated and free of inconsistency. Furthermore, almost all members would use the system frequently, independently of the group they belong to. One issue which differentiates developers and non-technical people regards the complexity of the system in terms of technical understanding. Developers found the system easy to use while promoters and managers were struggling with it at the beginning. After explaining to them the core functionalities and features they started to feel more confident to use it.

For instance, most of the parameters need to be configured before an activity is created. Some of them, such as *Visit GAP* or *Min/Max Dwell Time* were not clear to them and lead

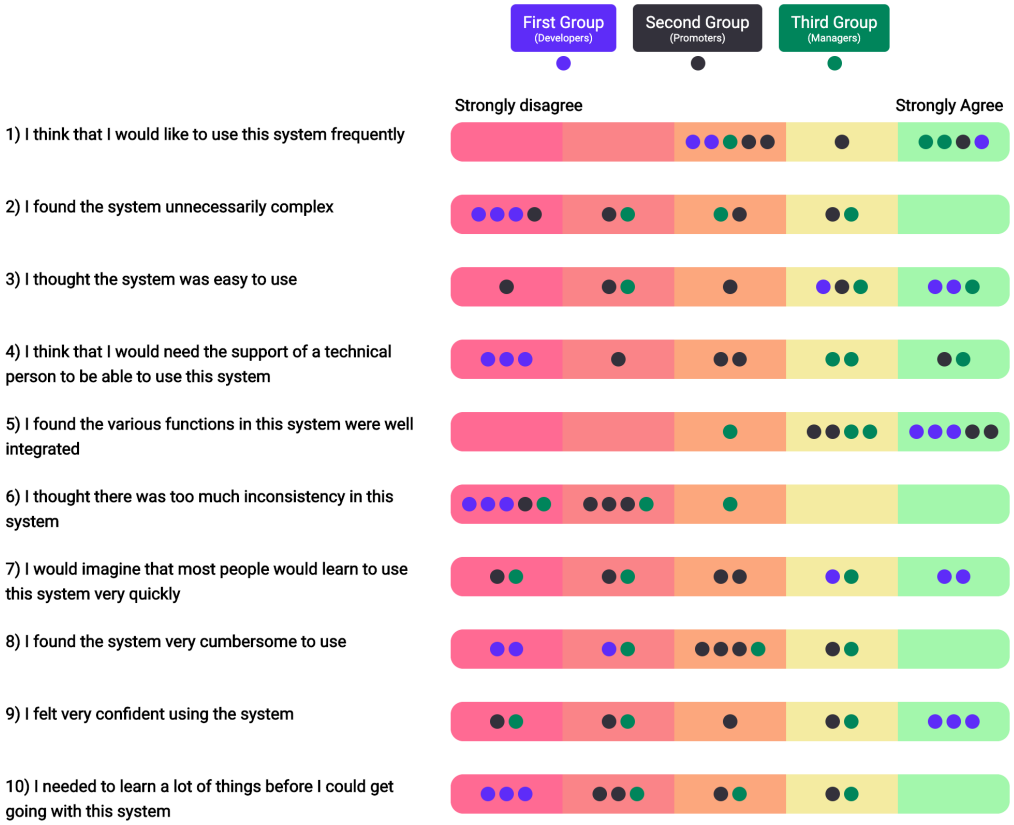

Figure 65: System Usability Scale Results

to the insertion of random intervals or numbers. They were not aware, that based on those metrics their activity would have setup in completely different ways. Another common issue was the wording. Interviewed people were confused about the terminology applied to the Stakeholders. Indeed, terms such as *activities*, *campaigns*, *brands* and *agencies* were new to them and lead to an initial disorientation while approaching the platform. For instance, they proposed to switch the term *activities* with *measurements*, since it is on the activity layer where sensors are configured and will afterward result in measurements.

Evaluated Screens Figure 66 illustrates the first step for creating a new activity in the front-end. First of all, one of the activity types needs to be selected.

Afterward, as shown by Figure 67 some activity details need to be configured. Firstly, the activity name has to be inserted. Then, values such as the environment (*i.e.,* activity format) and surroundings (*e.g.,* Indoor or Outdoor) have to be chosen. Finally, the activity place has to be located. By manually moving the map sensor, the geographical coordinates of the chosen location are calculated automatically. These values will be relevant to locate the activity more specifically.

Figure 68 illustrates the third step of the activity form. Within this step, the activity configuration is setup. In fact, using the colored slider, parameters such as radius and mix-max dwell times for different zone of interests can be configured. Finally, a bunch of available sensors can be assigned to the activity. These sensors will be then used for measuring metrics during the activity life cycle.

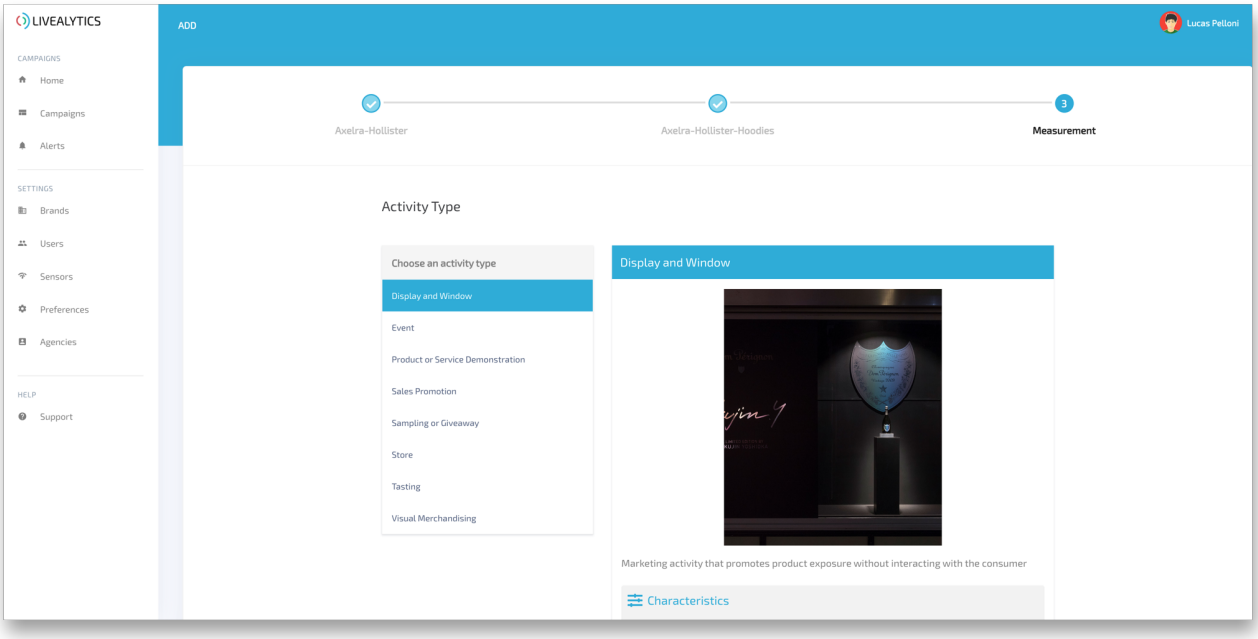

Figure 66: Activity Form - First Step

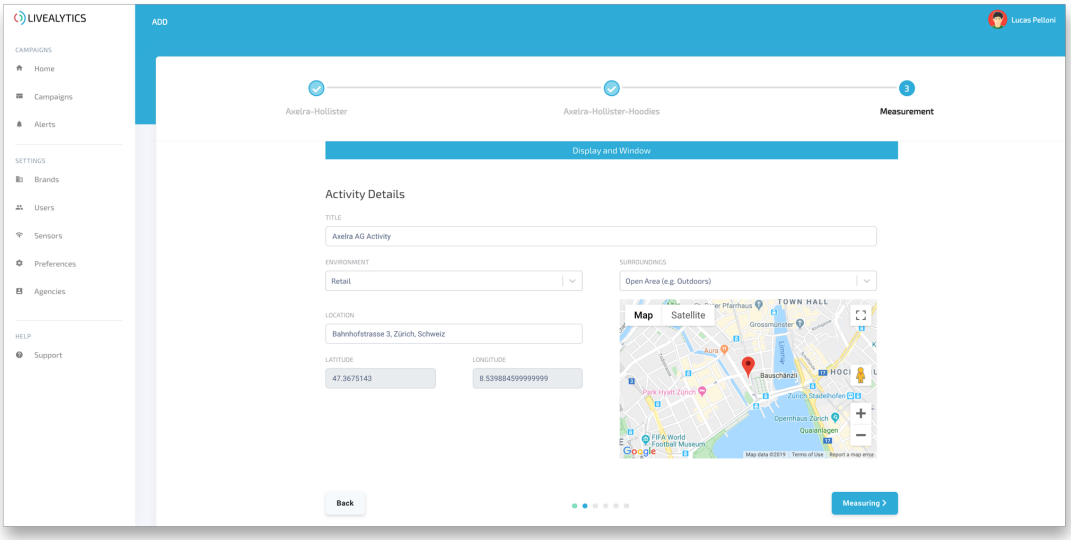

Figure 67: Activity Form - Second Step

Figure 69 and 70 show the activity dashboard. Charts can be sorted by date and time and exported as PNG or PDFs.

Figure 71 illustrates the agency overview in the perspective of root user. The overview follows a card-based design principle. Figure 72 illustrates the campaign overview. Within this tab, all activities belonging to the concerned campaign can be tracked. Activities can be stored as drafts or have an active status.

Finally, Figure 73 shows the sensor's overview from the perspective of a root account. Within this tab, all sensors can be monitored, configured, and assigned to other agencies.

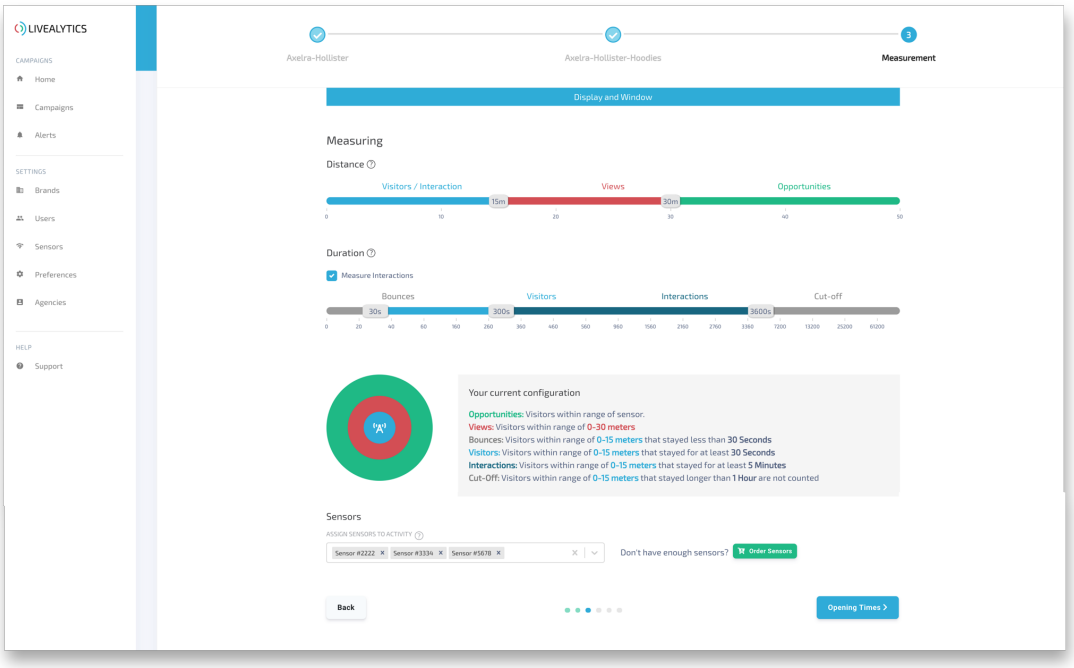

Figure 68: Activity Form - Third Step

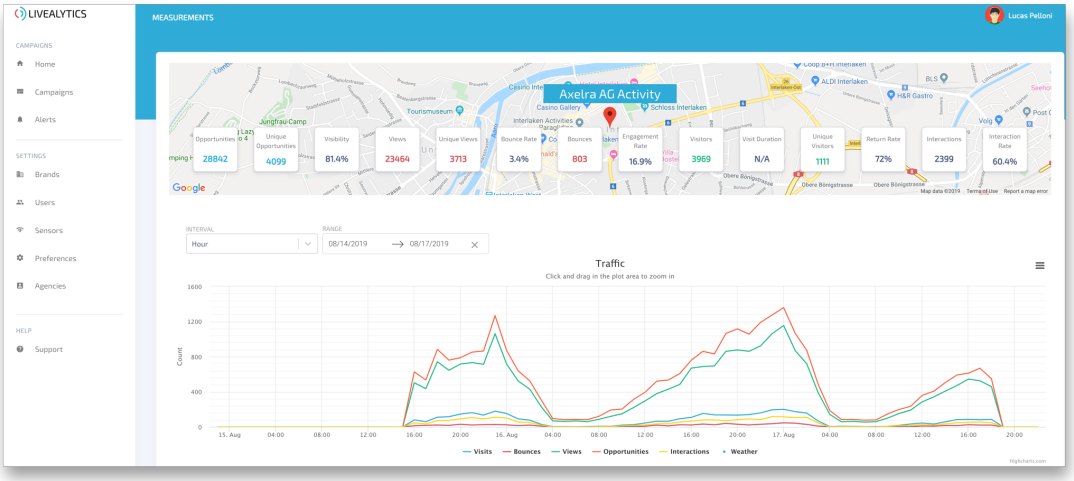

Figure 69: Activity Overview - First Part

Furthermore, the current configuration for each sensor can be downloaded and executed locally.

Performance to Display Metrics and KPIs As the complexity of the architecture and the quantity of accumulated data grew, the front-end started being less responsive and requests were taking longer to load. The culprit was the back-end, which needed more time to process the information and some Lambda functions started timing-out as well. After further inspection, the problem was to be found in the query executed to retrieve the data from the database and not in the code written in JavaScript contained in the Lambda functions.

Figure 70: Activity Overview - Second Part

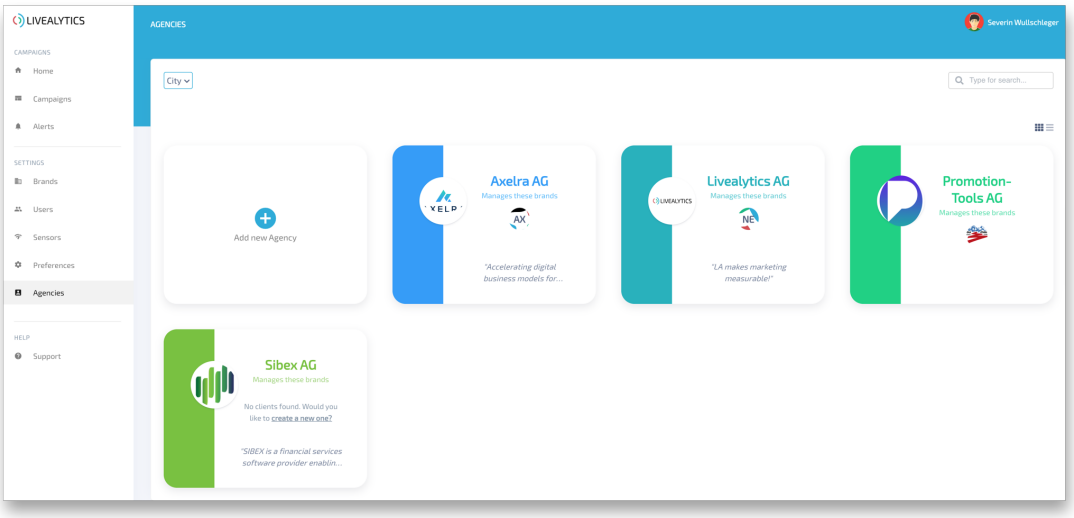

Figure 71: Agencies Overview

All queries run against the table `personCoordinates` which contains the data gathered from all the cameras. In average, this table grows by 73'000 newly inserted entries daily. Therefore, the performance degradation was more noticeable every day.
Two approaches were taken to address the slow response time problem:

1. **Database Indexing:** In the field of databases, an index is a data structure implemented with the goal of improving the speed of data search and retrieval in queries, at the cost of additional storage space to maintain the index data structure. In a table without indexes, every query forces the system to read all the rows contained in it. However, an index reduces the number of rows to be read in order to complete the query. Indexes can be generated using one or multiple columns of a table. Before deciding on which column indexes shall be created, it is necessary to evaluate which operations are the most frequent and critical. A wrong choice of indexes could degrade the overall perfor-

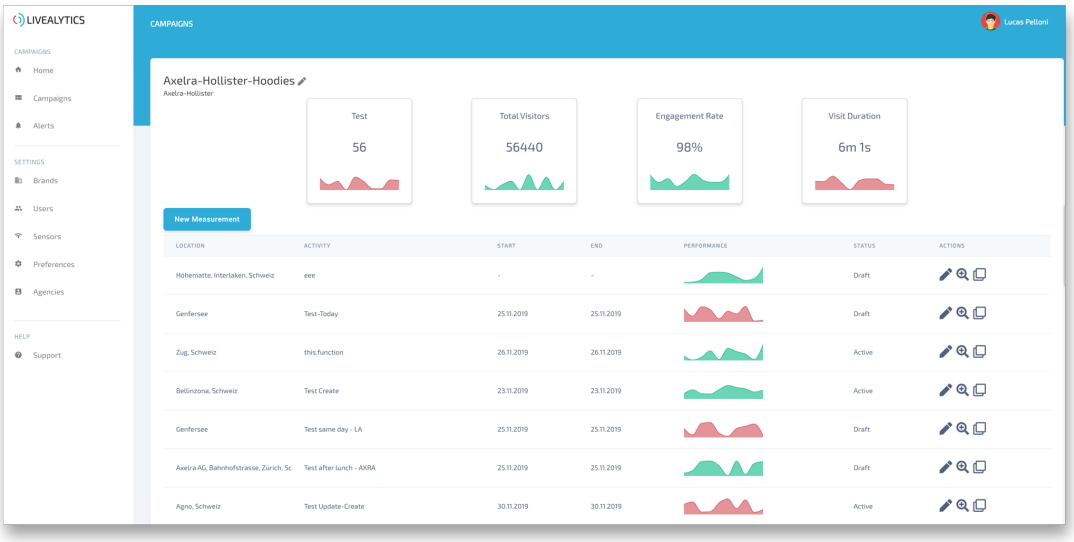

Figure 72: Campaign Overview

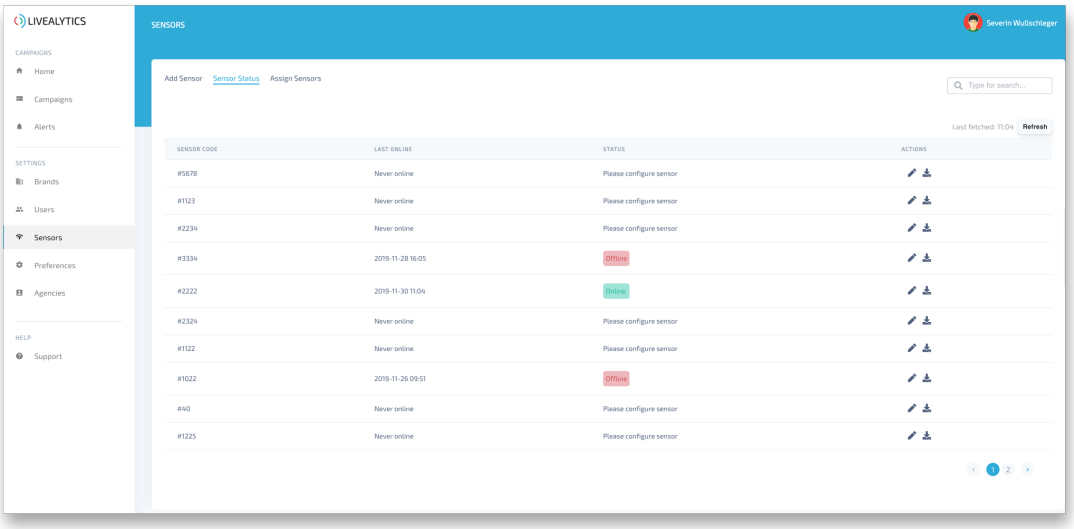

Figure 73: Sensors Overview

mance as well since overhead is introduced to maintain the index data structure.
In order to create a correct index or a set of indexes, it is required to know how typically the database tables are accessed. By analyzing the queries which run against the database when accessing the `personCoordinates` table, it can be concluded that a look-up returns rows that satisfy the following predicate in the `WHERE` clause: **filialeId, camera, timestamp**. Hence, an index composed of those three columns highly reduces the required time for execution. Table 16 shows how long it takes to retrieve 14'667 rows of an entire day without and with the index based on the columns aforementioned; almost 100 times faster in average.

2. **Cronjob to update a table from a view:** Incoming data from the camera has to be aggregated before being served to the front-end. Aggregation on the fly, as the data

Run	Without Index	With Index
1st	27.865s	321ms
2nd	22.478s	242ms
3rd	15.664s	170ms
Average	**22.002s**	**244ms**

Table 16: Elapsed Time to Retrieve 14'667 Rows of Data With and Without An Index From the *personCoordinates* Table Containing Around 5'700'000 Entries.

is retrieved from the database, was fast enough as long as the mole of data was not too much. With too many entries in the database (more than 5'700'000 entries), the performances started to quickly degrade. Therefore, an additional table containing the result of the aggregation was created. The table is updated every 15 minutes from view. The process of updating happens in the background, hence the operational cost

Run	Aggregation on the fly	Already aggregated
1st	117.067s	84ms
2nd	116.451s	36ms
3rd	115.772s	33ms
Average	**116.430s**	**51ms**

Table 17: Elapsed Time to Retrieve Data with Aggregation On-the-fly Versus Retrieving Data Aggregated

of the update can be omitted. Table 17 shows the difference between aggregating everything on the fly versus retrieving only the aggregated data. Important to mention is that the aggregation on-the-fly retrieves data from the table `personCoordinates`, which has already been optimized. Despite having a delay of 15 minutes before being able to have the latest aggregated data, the solution utilizing the *cronjob* is definitely required. Lambda functions do not timeout anymore and the data is retrieved almost instantaneously.

6 Discussions and Impacts

Overall, PasWITS achieved its intended results; however, it had been impacted by the pandemic. Therefore, these impacts are detailed below.

6.1 Methodology and System Design

The project was developed following the philosophy of proposing a modular and data-driven approach. Thus, the initial focus considered the main tracking sources, WiFi and Bluetooth, investigating in detail the possibilities of their use in passive tracking of mobile devices. During the testing of these prototypes, passive tracking of wireless signals poses several additional challenges related to data scarcity and signal variability due to various environmental conditions. Therefore, the project investigated additional sources such as RFID, LiDAR, and Cameras to expand the array of tracking solutions to ensure minimal privacy with an economically viable approach.

To ensure that the expansion of other sources would not affect the main innovation proposed in the project, *i.e.,* the data correlation and prediction engine, a standard data structure was proposed to ensure that the data provided by each source can be pre-processed to provide synchronized geometric coordinates on the same time scale. That is, they can provide timestamped coordinates based on the global clock. Another important consideration regarding the various prototypes developed for the data sources is the use of an overall distributed architecture based on data streaming to a sink. With this, several capture sources can be placed strategically, periodically sending data to the sink via a local network. The sink, in turn, must be a node with greater processing power (than nodes spread across the tracked site) to enable the possibility of correlating the data in real time or sending it to the Cloud, where it is processed and made available to the front-end. This process entails a relative delay for data visualization, implying that the term real-time is in the time scale of minutes and not seconds.

6.2 Pandemic Impacts

One of the major challenges was the COVID pandemic which coincided with a large part of the project's development. livealytics focuses exclusively on monitoring and analyzing marketing campaigns at public events, which were largely canceled or restricted during the pandemic. As a result, the project's execution was immediately affected, requiring minor alterations to assure its continuation, albeit on a smaller scale. Several steps were made to avoid these issues, including increasing the data input to include cameras and LiDAR and offering a monitoring system (*cf.* Cloud Counter - CCount).

The inability to evaluate the created prototypes at events and trade fairs where other livealytics prototypes were placed directly had a major influence on the project's execution. MICE (Meetings, incentives, conferences, and exhibitions) events constituted the majority of the business that livealytics targeted, and as such, it impacted business and academic results. Consequently, techniques for evaluating data streaming systems were created using a combination of simulated data creation and, to a lesser degree, actual sensors in controlled conditions (university or private environments provided by livealytics as the Schlieren

Warehouse). In this regard, collaboration was great so that prototypes could be examined in actual surroundings, although on a smaller size.

6.3 Security Concerns — Privacy and Confidentiality

One of the important aspects addressed in the project's development is the contradiction between the search for increased accuracy in tracking individuals versus the privacy of the tracked individuals. Privacy in its strict sense, *i.e.,* not being observed by something or someone, is not always possible in public environments - where crowds are monitored by surveillance cameras or security personnel. On the one hand, increased accuracy requires the search for methods that allow objective identification of individuals using unique visible characteristics and unique characteristics of devices carried by individuals. On the other hand, it is necessary to ensure that information obtained by different tracking approaches is not invasive to the point of linking an individual to an identity. In this sense, privacy in the PasWITS project refers to data privacy of monitored people/devices, ensuring that obtained information is not linked to personally identifiable information.

Confidentiality concerns include who is authorized to access the information extracted from a campaign. In this sense, monitoring information extracted locally in a campaign is sent over an encrypted channel to a backend, and data (and metadata) is stored and encrypted individually per client. Thus, a client only has access to its campaign's data. Only super users have access to the data of all clients/campaigns (*cf.* role-based access modeling defined in Section 4). Another concern about distributed data streaming architecture is the possible interception of data sent by one or more sensors to a local or cloud sink at a campaign site (sniffing). In this sense, encrypted channels should be used to ensure that data is not intercepted and modified by third parties. It is also important to note that the prototype design allows the solution to be entirely hosted and run on-premises by the client once they have the necessary resources. This option depends on how a final product can be implemented by livealytics and bilateral agreements with customers who have strict requirements regarding the external storage of their campaign data.

7 Concluding Remarks

As a greater amount of data is generated by many mobile devices humans carry around on a daily basis, at the same time, there is the possibility for these data to be captured and analyzed for various purposes. As such, measuring the public interest in a particular product or service is of fundamental importance for the strategic planning of businesses in an increasingly digitized society. The mission of livealytics – the business and technology partner of this project – is to provide a full-fledged analytics solution for measuring and benchmarking the performance of sales and live marketing promotion activities, trade shows, and retail spaces. As such, the PasWITS project explored the possibility of merging multiple input sources with the geo-localization of devices. And it is now possible to increase the accuracy for the unique identification of devices within the data link layer.

PasWITS followed a modular development methodology that allowed for a relatively easy adaptation in face of the challenges faced (*e.g.,* pandemic) and could expand the data input vector (sensors) without compromising data correlation and prediction. Prototypes for the several input sensor types were developed with a distributed data streaming architecture based on (and requiring) a data structure for correlation (which was specified) and a scenario prediction to be performed. The use of a standard data structure implied a compromise in which it was necessary to include a pre-processing step in the collected data to ensure that it could be sent and processed in a uniform and convenient way for its correlation (implying greater tracking accuracy).

Although pandemic-related challenges have prevented large-scale testing and evaluation (as originally planned for), several positive observations have to be made from both a business and an academic, research point of view. From a business point of view, the Cloud Counter solution has become a product in livealytics' service portfolio. From an academic point of view, several academic and peer-reviewed publications have been prepared, based on the CSG's teamwork with students, to cover theoretical and practical opportunities in investigating the operation of sensors. Furthermore, the partnership between CSG and livealytics was very insightful, constructive, and cooperative, in which there was ample (when allowed due to health/safety regulations) opportunity for the evaluation of prototypes in the field, as well as the availability of hardware and software used for livealytics for experimentation in an academic setting. This partnership was extremely beneficial in refining the development of several prototypes and validating the results of those experiments performed.

Finally, the project points out that a data-driven solution is extremely favorable for increasing the accuracy of tracing through source correlation. In this sense, the concepts and prototypes developed within PasWITS can be refined to become products, such as Cloud Counter, and these can increase the service portfolio of livealytics. Furthermore, it is observed that additional sources, like the Ultra-Wideband (UWB) communication technology, can be developed further to increase the tracking accuracy at a relatively low cost, which could boost livealytics in the resumption of public events.

List of Figures

List of Tables

References

[1] M. Abid, P. Compagnon, and G. Lefebvre, "Improved CNN-based Magnetic Indoor Positioning System using Attention Mechanism," in *2021 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, 2021, pp. 1–8.

[2] S. Ahuja and P. Potti, "An introduction to RFID technology," Vol. 02, No. 3, pp. 183–186. [Online]. Available: http://www.scirp.org/journal/doi.aspx?DOI=10.4236/cn.2010.23026

[3] J. B. Andersen, T. S. Rappaport, and S. Yoshida, "Propagation Measurements and Models for Wireless Communications Channels," *IEEE Communications Magazine*, Vol. 33, No. 1, pp. 42–49, 1995.

[4] I. Ashraf, M. Kang, S. Hur, and Y. Park, "MINLOC: Magnetic Field Patterns-based Indoor Localization using Convolutional Neural Networks," *IEEE Access*, Vol. 8, pp. 66 213–66 227, 2020.

[5] Axelra, "We Accelerate Tech Venture Building at early stage for Corporates and Startups with skin in the Game." 2022. [Online]. Available: https://www.axelra.com/

[6] A. Bangor, P. T. Kortum, and J. T. Miller, "An Empirical Evaluation of the System Usability Scale," *Intl. Journal of Human–Computer Interaction*, Vol. 24, No. 6, pp. 574–594, 2008.

[7] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software*. John Wiley & Sons, 2004.

[8] A. Bekkali, H. Sanson, and M. Matsumoto, "RFID indoor positioning based on probabilistic RFID map and kalman filtering," in *Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2007)*. IEEE, pp. 21–21. [Online]. Available: http://ieeexplore.ieee.org/document/4390815/

[9] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson Correlation Coefficient," in *Noise Reduction in Speech Processing*. Springer, 2009, pp. 1–4.

[10] K. Bernardin and R. Stiefelhagen, "Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics," *EURASIP Journal on Image and Video Processing*, Vol. 2008, pp. 1–10, 2008.

[11] S. Bhatti and J. Xu, "Survey of Target Tracking Protocols Using Wireless Sensor Network," in *5th International Conference on Wireless and Mobile Communications (WMC 2009)*. Cannes, France: IEEE, 2009, pp. 110–115.

[12] Biondi, Philippe, "Scapy: the Python-based interactive packet manipulation program & library." [Online]. Available: https://github.com/secdev/scapy

[13] *Bluetooth Core Specification v5.2*, Bluetooth SIG, Dec. 2019.

[14] Bluetooth SIG Inc., "Understanding Bluetooth Range." [Online]. Available: https://bit.ly/3Irpm2Q

[15] C. A. Boano, N. Tsiftes, T. Voigt, J. Brown, and U. Roedig, "The Impact of Temperature on Outdoor Industrial Sensornet Applications," *IEEE Transactions on Industrial Informatics*, Vol. 6, No. 3, pp. 451–459, 2009.

[16] S. Chai, R. An, and Z. Du, "An Indoor Positioning Algorithm using Bluetooth Low Energy RSSI," in *2016 International Conference on Advanced Materials Science and Environmental Engineering*. Atlantis Press, 2016.

[17] C. Chen, Y. Chen, H.-Q. Lai, Y. Han, and K. R. Liu, "High accuracy indoor localization: A WiFi-based approach," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 6245–6249.

[18] L. Cheng and J. Wang, "How Can I Guard My AP? Non-Intrusive User Identification for Mobile Devices Using WiFi Signals," in *Proceedings of the 17th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. MobiHoc '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 91–100.

[19] M. Chernyshev, "An Overview of Bluetooth Device Discovery and Fingerprinting Techniques–assessing the Local Context," in *13th Australian Digital Forensics Conference*. RI Security Research Institute, Edith Cowan University, Perth, Western Australia, 2015.

[20] B. Consulting, "MAC-Adresse und der Datenschutz – Fingerabdruck des Computers?" 2018. [Online]. Available: https://brands-consulting.eu/mac-adresse-und-der-datenschutz-fingerabdruck-des-computers

[21] D. Coppens, E. D. Poorter, A. Shahid, S. Lemey, and C. Marshall, "An overview of ultra-wideband (uwb) standards(ieee 802.15.4, fira, apple): Interoperability aspects and future research directions," 2 2022. [Online]. Available: https://arxiv.org/abs/2202.02190

[22] T. Creutzenberg, "Prototyping: An Overview of Current Trends, Developments, and Research in Prototyping," pp. 40–48. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.227.7242&rep=rep1&type=pdf#page=47

[23] A. Das, J. H. Schwee, E. S. Kolvig-Raun, and M. B. Kjærgaard, "Dataset," in *Proceedings of the 2nd Workshop on Data Acquisition To Analysis - DATA'19*, Unknown, Ed. New York, New York, USA: ACM Press, 2019, pp. 43–46.

[24] A. De Carli, M. Franco, A. Gassmann, C. Killer, B. Rodrigues, E. Scheid, D. Schoenbaechler, and B. Stiller, "WeTrace–a Privacy-preserving Mobile COVID-19 Tracing Approach and Application," *arXiv preprint arXiv:2004.08812*, 2020.

[25] F. de Ponte Müller, "Survey on Ranging Sensors and Cooperative Techniques for Relative Positioning of Vehicles," *Sensors*, Vol. 17, No. 2, p. 271, 2017.

[26] P. Dickinson, G. Cielniak, O. Szymanezyk, and M. Mannion, "Indoor Positioning of Shoppers using a Network of Bluetooth Low Energy Beacons," in *International Conference on Indoor Positioning and Indoor Navigation (IPIN 2016)*. Madrid, Spain: IEEE, 2016, pp. 1–8.

[27] C. Duan, L. Yang, and Y. Liu, "Accurate Spatial Calibration of RFID Antennas via Spinning Tags," in *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2016, pp. 519–528.

[28] R. Faragher and R. Harle, "Location Fingerprinting with Bluetooth Low Energy Beacons," *IEEE Journal on Selected Areas in Communications*, Vol. 33, No. 11, pp. 2418–2428, 2015.

[29] Fedlex, "Federal Act on Data Protection (FADP)," https://www.fedlex.admin.ch/eli/cc/1993/1945_1945_1945/en, 2021, Accessed: 2021-10-17.

[30] A. Filgueira, H. Gonzalez-Jorge, S. Laguela, L. Diaz-Vilarino, and P.Arias, "Quantifying the Influence of Rain in LiDAR Performance," *Measurement*, Vol. 95, pp. 143–148, 2017.

[31] K. Finkenzeller, *RFID Handbook: Fundamentals and Applications in Contactless*

Smart Cards, Radio Frequency Identification and Near-field Communication. John wiley & sons, 2010.

[32] FiRa, "Our members — fira consortium," 2022. [Online]. Available: https://www.firaconsortium.org/about/members

[33] V. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello, "Bayesian Filtering for Location Estimation," *IEEE pervasive computing*, Vol. 2, No. 3, pp. 24–33, 2003.

[34] M. Franco, B. Rodrigues, C. Killer, E. J. Scheid, A. De Carli, A. Gassmann, D. Schönbächler, and B. Stiller, "WeTrace: A Privacy-preserving Tracing Approach," *Journal of Communications and Networks*, pp. 1–16, 2021.

[35] M. F. Franco, B. Rodrigues, and B. Stiller, "MENTOR: the Design and Evaluation of a Protection Services Recommender System," in *2019 15th international conference on network and service management (CNSM)*. IEEE, 2019, pp. 1–7.

[36] M. S. Gast, *802.11 Wireless Networks: The Definitive Guide, Second Edition*. O'Reilly Media, Inc., 2005.

[37] Great Scott Gadgets, "Ubertooth," https://github.com/greatscottgadgets/ubertooth, 2020, revision: c2cc373, accessed: 2021-07-19.

[38] F. Gringoli, M. Schulz, J. Link, and M. Hollick, "Free Your CSI: A Channel State Information Extraction Platform For Modern Wi-Fi Chipsets," in *Proceedings of the 13th International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization*, ser. WiNTECH '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 21–28.

[39] B. Habtemariam, R. Tharmarasa, T. Thayaparan, M. Mallick, and T. Kirubarajan, "A Multiple-Detection Joint Probabilistic Data Association Filter," *IEEE Journal of Selected Topics in Signal Processing*, Vol. 7, No. 3, pp. 461–471, 2013.

[40] Hacker Warehouse, "Ubertooth one," https://hackerwarehouse.com/product/ubertooth-one/, 2020, accessed: 2021-08-18.

[41] W. S. Harms and J. Strobel, "Usability Evaluation Von Web Angeboten Mit Dem Web Usability Index." DGI, pp. 283–292.

[42] C. Heindl, Toka, and J. Valmadre, "py-motmetrics." [Online]. Available: https://github.com/cheind/py-motmetrics

[43] IEEE, "IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture," *IEEE Std 802-2014 (Revision to IEEE Std 802-2001)*, pp. 1–74, 2014.

[44] Impinj, "Speedway Connect Capabilities," https://www.impinj.com/products/software/speedway-connect, 2021, Accessed: 2021-09-25.

[45] A. Inc., "Apple's Report on Wi-Fi Privacy," 2021. [Online]. Available: https://support.apple.com/guide/security/wi-fi-privacy-secb9cb3140c/web

[46] Inpixon, "Ultra-wideband (uwb) positioning & sensor technology," https://www.inpixon.com/technology/standards/ultra-wideband.

[47] Intel, "Intel Realsense LiDAR Camera L515," Sep 2021. [Online]. Available: https://www.intelrealsense.com/lidar-camera-l515/

[48] International Data Corporation (IDC), "Smartphone Market Share," https://www.idc.com/promo/smartphone-market-share/vendor, 2020, accessed: 2020-07-30.

[49] jj, "Generating Uniformly Distributed Points on Sphere." [Online]. Available: https://medium.com/@all2one/generating-uniformly-distributed-points-on-sphere-1f7125978c4c

[50] A. Juels, "RFID Security and Privacy: A Research Survey," *IEEE Journal on Selected Areas in Communications*, Vol. 24, No. 2, pp. 381–394, 2006.
[51] R. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *The American Society of Mechanical Engineers- ASME*, Vol. 82, pp. 35–45, 1960.
[52] Y. Kim and H. Bang, "Introduction to kalman filter and its applications," in *Introduction and Implementations of the Kalman Filter*. IntechOpen, 2018.
[53] Kinexon, "Everything You Need to Know About UWB Technology," https://kinexon.com/uwb-technology/.
[54] J. Košecká, F. Li, and X. Yang, "Global Localization and Relative Positioning based on Scale-invariant Keypoints," *Robotics and Autonomous Systems*, Vol. 52, No. 1, pp. 27–38, 2005.
[55] M. Kutila, P. Pyykonen, H. Holzhuter, M. Colomb, and P. Duthon, "Automotive LiDAR Performance Verification in Fog and Rain," *IEEE Intelligent Transportation Systems Conferenc*, 2018.
[56] R. Labbe, "Kalman and Bayesian Filters in Python," https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python, commit 91f8010bee8cd7e07bdca338ecd90c3dd7735e92, 2014.
[57] M. Lalmas, H. O'Brien, and E. Yom-Tov, "Measuring User Engagement," *Synthesis lectures on information concepts, retrieval, and services*, Vol. 6, No. 4, pp. 1–132, 2014.
[58] J. Larsson, "Distance Estimation and Positioning based on Bluetooth Low Energy Technology," Master Thesis, 2015. [Online]. Available: https://www.diva-portal.org/smash/record.jsf?pid=diva2:859549
[59] G. Li, E. Geng, Z. Ye, Y. Xu, J. Lin, and Y. Pang, "Indoor Positioning Algorithm Based on the Improved RSSI Distance Model," *Sensors*, Vol. 18, No. 9, p. 2820, 2018.
[60] Y. Li and T. Zhu, "Gait-Based Wi-Fi Signatures for Privacy-Preserving," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 571–582.
[61] K. Lim, P. Treitz, M. Wulder, B. St-Onge, and M. Flood, "LiDAR Remote Sensing of Forest Structure," *Progress in physical geography*, Vol. 27, No. 1, pp. 88–106, 2003.
[62] H. Liu, Y. Wang, J. Liu, J. Yang, and Y. Chen, "Practical User Authentication Leveraging Channel State Information (CSI)," in *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*, ser. ASIA CCS '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 389–400.
[63] G. LLC, "Privacy: MAC Randomization," 2020. [Online]. Available: https://source.android.com/devices/tech/connect/wifi-mac-randomization
[64] M. Lourakis, "A Brief Description of the Levenberg-Marquardt Algorithm Implemented by Levmar," *Foundation of Research and Technology*, Vol. 4, No. 1, pp. 1–6, 2005.
[65] J. Luomala and I. Hakala, "Effects of temperature and humidity on radio signal strength in outdoor wireless sensor networks," in *2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, 2015, pp. 1247–1255.
[66] K. Madsen, H. Nielsen, and O. Tingleff, "Methods for Non-Linear Least Squares Problems (2nd ed.)," p. 60, Jan. 2004.

[67] Mathuranathan, “Log Distance Path Loss or Log Normal Shadowing Model - GaussianWaves,” 9 2013. [Online]. Available: https://www.gaussianwaves.com/2013/09/log-distance-path-loss-or-log-normal-shadowing-model/
[68] C. Matte, “Wi-Fi Tracking: Fingerprinting Attacks and Counter-Measures,” Ph.D. dissertation, INSA Lion, 2017.
[69] P. McManamon, “History of LiDAR,” in *LiDAR Technologies and Systems*, 2019, pp. 29–34.
[70] T. Meng, X. Jing, Z. Yan, and W. Pedrycz, “A Survey on Machine Learning for Data Fusion,” *Information Fusion*, Vol. 57, pp. 115–129, 2020.
[71] L. Mueller, “LaFlector’s Source Code,” 2021, https://gitlab.ifi.uzh.ch/rodrigues/laflector, last visit May 21st, 2021.
[72] ——, “Single Person Evaluation Videos of LaFlector,” 2021, https://owncloud.csg.uzh.ch/index.php/s/LrqgR3sAa93ozdL, last visit April 20th, 2021.
[73] B. Pestourie, “Uwb secure ranging and localization,” 2021. [Online]. Available: https://hal.archives-ouvertes.fr/tel-03136561
[74] Qorvo, “Technology - Qorvo,” https://www.qorvo.com/innovation/ultra-wideband/technology.
[75] R. Ribeiro, B. Rodrigues, C. Killer, L. Baumann, M. Franco, E. J. Scheid, and B. Stiller, “ASIMOV: A Fully Passive WiFi Device Tracking,” in *IFIP Networking 2021*. Espoo, Finland: IFIP, jun 2021, pp. 1–3.
[76] B. Rodrigues, C. Halter, M. Franco, E. J. Scheid, C. Killer, and B. Stiller, “BluePIL: a Bluetooth-based PassIve Localization Method,” in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2021, pp. 28–36.
[77] Shanghai Slamtec. Co., Ltd, “SLAMTEC Mapper M1M1 Datasheet,” 2019, https://bit.ly/3tBsqCX, last visit April 20th, 2022.
[78] M. Singh, M. Roeschlin, E. Zalzala, P. Leu, and S. Capkun, “Security analysis of ieee 802.15.4z/hrp uwb time-of-flight distance measurement,” 2021.
[79] G. Sithole and S. Zlatanova, “Position, Location, Place and Area: An indoor Perspective,” *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci*, Vol. 3, No. 4, pp. 89–96, 2016.
[80] SkyRFID LLC., “RFID Range Overview.” [Online]. Available: https://bit.ly/3ufVcL9
[81] J. Sommers and P. Barford, “Cell vs. WiFi,” in *Proceedings of the 2012 ACM Conference on Internet Measurement Conference - IMC ’12*, J. Byers, J. Kurose, R. Mahajan, and A. C. Snoeren, Eds. New York, New York, USA: ACM Press, 2012, p. 301.
[82] D. Spill and A. Bittau, “BlueSniff: Eve Meets Alice and Bluetooth,” *WOOT*, Vol. 7, pp. 1–10, 2007.
[83] theAIGuysCode, “Everything You Need To Get YOLOv3 Up and Running in the Cloud.” [Online]. Available: https://github.com/theAIGuysCode/YOLOv3-Cloud-Tutorial
[84] Tzutalin, “Tzutalin/labelimg: LabelImg is a Graphical Image Annotation Tool and Label Object Bounding Boxes in Images.” [Online]. Available: https://github.com/tzutalin/labelImg
[85] Ubiquiti Inc., “G4 Pro Beats the Leading Home Brand Cameras in Image Quality Shootout.” [Online]. Available: https://blog.ui.com/2019/07/05/g4-pro-beats-the-leading-home-brand-cameras-in-image-quality-shootout/
[86] M. Vanhoef, C. Matte, M. Cunche, L. S. Cardoso, and F. Piessens, “Why MAC Address Randomization is Not Enough: An Analysis of Wi-Fi Network Discovery Mechanisms,”

in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 413–424.

[87] D. Vasisht, S. Kumar, and D. Katabi, "Decimeter-Level Localization with a Single WiFi Access Point," in *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation*, ser. NSDI'16. USA: USENIX Association, 2016, p. 165–178.

[88] M. Versichele, L. De Groote, M. C. Bouuaert, T. Neutens, I. Moerman, and N. Van de Weghe, "Pattern Mining in Tourist Attraction Visits Through Association Rule Learning on Bluetooth Tracking Data: A Case Study of Ghent, Belgium," *Tourism Management*, Vol. 44, pp. 67–81, 2014.

[89] P. Voigt and A. Von dem Bussche, "The EU General Data Protection Regulation (GDPR)," *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, Vol. 10, p. 3152676, 2017.

[90] H. Wang, B. Wang, B. Lui, X. Meng, and G. Yang, "Pedestrian recognition and tracking using 3D LiDAR for autonomous vehicle," *Robotics and Autonomous Systems*, Vol. 88, pp. 71–78, 2017.

[91] Y. Wang, X. Yang, Y. Zhao, Y. Liu, and L. Cuthbert, "Bluetooth Positioning Using RSSI and Triangulation Methods," in *2013 IEEE 10th Consumer Communications and Networking Conference (CCNC)*. Las Vegas, USA: IEEE, 2013, pp. 837–842.

[92] E. Weisstein, "Sphere Point Picking." [Online]. Available: https://mathworld.wolfram.com/SpherePointPicking.html

[93] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," *Chapel Hill, NC, USA*, p. 16, 2006.

[94] G. Welch, G. Bishop *et al.*, "An Introduction to the Kalman Filter," 1995.

[95] Wikipedia, "Non-linear least squares." [Online]. Available: https://en.wikipedia.org/wiki/Non-linear_least_squares

[96] Wireshark, "tshark: Terminal-based Wireshark." [Online]. Available: https://www.wireshark.org/docs/man-pages/tshark.html

[97] Xovis, "Guaranteed Data Privacy," https://www.xovis.com/de/technology/detail/guaranteed-data-privacy/, 2021, Accessed: 2021-09-25.

[98] V. Yajnanarayana, S. Dwivedi, and P. Händel, "Ir-uwb detection and fusion strategies using multiple detector types," 2015.

[99] Z. Yang, Z. Zhou, and Y. Liu, "From RSSI to CSI: Indoor Localization via Channel Response," *ACM Comput. Surv.*, Vol. 46, No. 2, 12 2013.

[100] Y. Zhuang, J. Yang, Y. Li, L. Qi, and N. El-Sheimy, "Smartphone-based Indoor Localization with Bluetooth Low Energy Beacons," *Sensors*, Vol. 16, No. 5, p. 596, 2016.

