Simulating Hybrid Location Models for Smart Agriculture Demonstrators

Master's Degree in Applied Computer Science

Faculty of Information Systems and Applied Computer Science

Chair of Mobile Systems

University of Bamberg

Author: David Jares

Matriculation Number: 1781350

Reviewers: Prof. Dr. Daniela Nicklas

Abstract

This study explores the applications of digitalization and smart agriculture in dairy farming. It aims to demonstrate a novel application that simulates and visualizes cow movement data, enabling farmers to perceive the value of implementing a digital sensor system for cow localization within their farms. The study leverages hybrid location systems to efficiently locate cows both in pastures and barns, aiming to address issues related to localization accuracy, battery runtimes, and sensor range inherent in current solutions. This work extends a pre-existing simulator that has initially been designed for smart city environments, to suit farming scenarios. The simulator utilizes real data sets from the WeideInsight project at the University of Bamberg, generating plausible dairy cow movement data sets for visualization. The primary objective is to provide farmers a realistic demonstration of the technology's benefits without immediate investment. Results indicate that the application is feasible and offers tangible value to farmers considering investment in this technology. In conclusion, this work advances the development of demonstrators in agricultural scenarios, highlighting the potential and viability of smart agriculture applications.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Digitalisation permeates all areas of our lives. There is hardly an area in which microchips and software do not play at least a supporting role. On the one hand, the technology has matured greatly in recent years. Microchips are smaller than ever before and as a result, the devices became smaller, more powerful and also consume less power. Battery-powered devices can thus be operated longer before they have to be recharged. Modern wireless technology such as the 5G transmission standard enables high-speed data transmission. Bluetooth enables data transmission with extremely low power consumption. Advances in robotics make it possible to provide machine support for many tasks, both for monotonous tasks and for dangerous or physically demanding tasks. Advances in artificial intelligence enable software development tasks to be handled in a new way, whereby algorithms that would otherwise be extremely complex and imprecise to program can now be developed automatically using modern methods. The necessary computing power, cloud infrastructure and training data for these data-intensive algorithms are now widely available. The large language models are now introducing a new era in terms of interactive communication between humans and machines. The networking of many small computing devices is known as the Internet of Things (IoT). Through networking, it becomes possible to develop systems that physically span a larger structure. Sensors can collect data and send it to cloud software for processing. The software can then in turn make decisions and send instructions for action to networked actuators. In this way, our physical world can eventually be networked and merged with the computer-based world. But the collection and provision of the real data sets necessary for a system can be cumbersome and resource-intensive, and thus potentially represents a bottleneck in the otherwise rather agile software development. This thesis delves into addressing this bottleneck by exploring the development of a simulator capable of generating qualitatively sufficient, artificially generated data sets. It attempts to illustrate the application of such a simulator in agricultural scenarios, specifically focusing on aiding farmers in visualizing and understanding the implications of integrating digital sensor systems in dairy farms.

1.1 Motivation

Digitalization is revolutionizing sectors globally and agriculture is making use of innovations like GPS, drones, and sensors to augment productivity, environmental protection, and animal welfare. The "WeideInsight" project exemplifies digital agriculture's potential, utilizing movement data to analyze behaviour of dairy cows to enhance their health and well-being.

This movement data can be collected using various technologies, such as GPS collars that each cow wears. The collars send a signal at time intervals to a base station, which records the cow's identification number, time and position. When these data are collected over a period of time, the result is a movement history of the respective cows. The data can then be transferred to the cloud and processed there. The evaluation can then classify the behaviour of the cows, whether a cow is resting or active, whether it is standing or lying down, whether it is eating or ruminating. In addition, anomalies in the behaviour can be detected and serve as a warning signal for possible diseases or stress.

Such an analysis enables farmers to better monitor the well-being of their cows and to intervene in time if a cow becomes conspicuous. In addition, the otherwise time-consuming manual observation process can be reduced. This could potentially allow more dairy cows to be kept at the same time. In addition, the individual needs of each cow can be taken into account, which helps to optimise feeding and milk yield.

1.2 Problem Statement

The aim of the *WeideInsight* project is to enable farmers to automatically track their cows both in the barn and on the pasture and then to use the data profitably. The positions should be as precise as possible, robust against disturbances due to external conditions, and the transmitters used on the cows' collars should consume as little energy as possible, so that as little maintenance as possible is required to recharge the transmitters.

There are currently several technologies that can be used for tracking cows. They all have their advantages and disadvantages. In fact, the conditions between barn and pasture are so different that some of the technologies are better suited to the indoor barn, while others work better on the pasture. A hybrid system of several technologies thus seems promising when it comes to realising the highest possible benefit at the lowest possible cost for the farmer. This concept is pursued in the *WeideInsight* project. Once the system is installed and configured, the user can interact with the system via a user interface, either through a mobile device or a computer. The farmer should should then be able to use the system to his advantage.

This is exactly where the problem arises, because the effort and costs involved in establishing such a system are so high that an economically sensible farmer would want to assess the added value as well as possible before making this risky investment. He should be able to project this prior assessment as tailor-made as possible onto his own farm. It would be best if he could see the system in action and test it before putting in all the effort and expense.

For this use case, the use of a simulation lends itself particularly well. Simulations are used in many disciplines for a variety of purposes. Often they make it possible to run through different scenarios, to identify possible weak points and optimisation possibilities at an early stage, but also to save costs by being able to look at and use the system even before it is implemented in reality. City construction planners, for example, can simulate the flow of traffic before they have construction projects for roads and traffic implemented. Flight pilots can use flight simulators to simulate different aircraft and scenarios that are costly and time-consuming to implement in reality. A farmer could use the simulator to represent his individual farm and to view and analyse the cows, their movements and their behaviour. He can configure the physical characteristics of his farm manually in the user interface. He configures the number and type of animals, as well as certain processes that take place at certain times. The simulation then artificially generates movement data over any given time frame for the configured cows. The farmer then has this data displayed visually in a UI and can interact with it there.

Another use case is to use the simulated data as input for the development phases of the actual software, because a lack of required input data can significantly affect the development and delivery of the Software. Here, simulated data can serve as an essential link in the development process.

Depending on the complexity of the problem domain to be modelled and the resources available, a simulation that simulates sufficiently authentic data can in practice prove to be an easy or a demanding task, sometimes even an impossible one. Often a simulation solution is tailored to a specific task domain. Thus, special simulators have been developed for many areas. Often there are several different simulators in a specific task area, which try to solve the problem or certain aspects of it with their own approach. For the mentioned project *WeideInsight*, the main task of the simulation would be the generation of artificial movement data of cows on the farm over a certain period of time. The layout of the farm should be configurable, as well as its occupation with cows and the recording of repetitive procedures that the cows have to complete, such as going to the milking place in a given time frame. With this input data, it should be possible to generate plausible data sets.

Several simulators for generating positional data for moving objects, including animals and humans, have been presented in research. As it turns out, some of them are interesting for the problem in the *WeideInsight* project, such as the *SmartSPEC* simulator. This is aimed at generating movement data for people in modern building complexes. There is also a research paper for the *WeideInsight* project that integrates this *SmartSPEC* simulator as a core system and at the same time explores whether the system can generate plausible data for cows in a farmer's barn. The results of the work seem to confirm this. However, as the system only serves one aspect of the use cases, further research is needed to evaluate the system's efficiency with regard to the overall project. It currently provides a working basis for integrating the *SmartSPEC* simulator into a cloud architecture that relies on microservices at its core. The result demonstrates a functional implementation and use of the system, which can be built upon in principle. However, the functionality is very limited if it is to actually serve the use cases described above and be used to demonstrate hybrid movement data. Firstly, although position data is generated in the barn area, the generation of position data for the outdoor areas is missing. Secondly, it does not yet support sequences that the animals perform at specific times in specific locations. If the simulation is to be able to serve demonstration purposes for farmers, essential extensions are still required with regard to data visualisation and interaction options with the system in the user interface.

The present research work therefore attempts to answer the following research question: "How can a plausible simulation of hybrid location systems support the demonstration of smart agriculture applications?"

Another problem arises from the fact that real ground truth data sets should ideally be available for training the simulation, but this cannot always be guaranteed due to the high effort involved. So how can the simulation be implemented and evaluated despite the lack of real data sets?

1.3 Methodology

Our approach to designing and implementing the simulation must be sound so that the goals can be achieved successfully. First we have to gain a deeper understanding of the *WeideInsight* Project and the existing Project of Pongratz [Pongratz, 2023] and the underlying *SmartSPEC* Project, as we would like to build and expand our work on this. We will also dive into the current state of relevant existing technologies. For the practical development, we will adopt an agile approach and conduct regular meetings with stakeholders to gather feedback. This will help us in shaping the design of the application and achieving the desired outcomes. Using this approach, we can refine the User Interface and also come up with a solution for the possible unavailability of real data sets.

1.4 Outline

We will try to extend the software system in such a way that the simulation functionality meets the basic use cases regarding the demonstration of the application to real farmers.

First we will have a look at the current research and background knowledge that may be relevant for us in designing possible problem-solving approaches.

Next, we will try to work out those aspects that need to be considered for a sufficient demonstration of the system for the purpose of convincing farmers. As a foundation, we will therefore work out the use cases for our thesis question.

After that, we will derive an abstract software architecture with which we would like to arrive at a solution to the problem.

Then we will devote ourselves to the concrete implementation of the system. The aim is to create a functional prototype that supports the formulated use cases as adequately as possible.

Subsequently, an evaluation of the prototype will be carried out, which should help us to assess to what extent the work contributes to answering the research question.

Finally, the results will help us to identify possible aspects for future research.

Chapter 2

Background

In this chapter, we will explore content from research literature that will aid us in designing and extending the simulator.

2.1 Overview of the WeideInsight Project

This paper is aligned with the ongoing *WeideInsight* project at the University of Bamberg. The subsequent section provides insight into the project and its present status.

Objectives and Motivation: While sensor-based monitoring of livestock is prevalent, particularly on extensive farms, the use of location and herd movement information is relatively rare. This information could yield considerable advantages to farmers, ranging from simple animal location to prolonged monitoring of movement patterns for health management. High costs and the constraints of existing technology serve as primary hindrances. Overall Objective of the Project: The *WeideInsight* project seeks to enhance value by incorporating affordable, energy-efficient localization solutions into combined pasture/barn operations. This integration is anticipated to advance animal welfare through the early identification of health and management-related situations and to allow farmers to operate grazing businesses economically with reduced effort Focus Areas: The project centers on:

- Developing and testing cost-efficient radio-based localization solutions for pasture and barn, with enhanced battery life
- Creating a simulator for the employment of the technology in distinct barn and pasture setups
- Incorporating and analyzing semantic location information in herd management systems to supply farmers with immediate added value from location and movement information
- Realizing labor savings and simplifications for farmers in pasture management procedures, thereby elevating animal welfare and transparency in farming processes.

This text delves into current research and technology concerning the use of location information in herd management. It is fundamentally about determining the locations of animals, especially within expansive pastures. Several techniques have been employed for this endeavor for numerous years. The advent of everyday GPS usage has revolutionized pasture management and has been pivotal in the development of global positioning systems.

An important advancement was the *GPS-GSM* positioning system devised by Blaupunkt Telematics GmbH in 2018. It is especially useful for large pastures and young animals but is energy-intensive. It is less appropriate for cows that graze intensively and are routinely brought to the barn for milking. Unfortunately, there is no reliable monitoring system currently available that offers high-frequency data on both animal behavior and location in barns and pastures.

Several systems are in development for outdoor location tracking. However, existing systems like *SMARTBOW* and *CowView*, locating cows within barns, exhibit limited efficiency in pastures. WLAN or Bluetooth technologies are inadequate due to their finite range. Low Power Wide Area Networks (LPWAN) are a promising alternative for pasture data transmission, offering extended range and reduced data rates.

The text further discusses localization models for herd management, involving geometric and symbolic models, and their combinations. However, present animal location systems are not designed to deliver a comprehensive solution for herd management. As a farm enlarges, the integration of animal and management information with location data, and its incorporation into existing herd management systems, become increasingly important.

Lastly, the discourse is about the *WeideInsight* project, which intends to formulate a system that combines the localization of animals in pastures and barns. A cost-effective LPWAN localization based on triangulation is implemented in pastures, and within barns, localization is executed via Bluetooth. The goal is to achieve high accuracy in location determination with minimal energy consumption and a high frequency of position data. This strategy, coupled with artificial intelligence, facilitates decision-making at both individual and herd levels.

2.2 Location Methods

The following section explores the main concepts of current location methods that are relevant for our Simulator Application. These are fundamental for the implementation of the requirements regarding position detection and power usage.

A localization system allows us to:

- Track the path of objects or people at a specific time
- Determine how long an object or person spends in a certain area
- Analyze interactions between objects and people

For Outdoor localization, a variety of technologies are available. The well-established GPS System is crucial for precise real-time localization. For our use cases, we are interested in regularly sending small amounts of data with low energy consumption. Thus, LPWAN (Low Power Wide Area Network) technology is applicable. More specifically, we will focus on a protocol called *Mioty*, which is based on LPWAN.

Indoor localization poses challenges due to a variety of obstacles impacting the results. Unlike outdoor positioning, indoor location systems require a combination of hardware and software to simulate GPS functionality within a bounded and enclosed environment. Essential components include sensors, RFID tags, and Bluetooth devices. There is no single dominant technology for indoor localization, as each processes sensor data differently [Pascale et al., 2021, p.1-2].

In our work, we focus on Beacons, a technology suitable for indoor environments, as our prototype will be based on a project that already successfully implements indoor localization using a Beacon-based system.

2.2.1 GPS

GPS, standing for *Global Positioning System*, is an outdoor location technology, serving as a space-based radio-navigation system that sends highly accurate navigation pulses to receivers on earth. A GPS receiver on earth can utilize these signals to determine its geographical coordinates by employing the principle of triangulation [Encyclopedia Britannica, 2022b].

Triangulation

A GPS receiver calculates the time a satellite signal takes to reach Earth and multiplies this time by the speed of a radio wave to determine the distance between the receiver and the satellite. This computation places the receiver on the surface of an imaginary sphere, with the radius being the calculated distance. Analyzing signals from at least four satellites simultaneously, the receiver computes the point where all four spheres intersect. This point defines the user's longitude, latitude, and altitude. Although theoretically, three satellites could provide a 3-dimensional fix, at least four are used to correct inaccuracies in the receiver's clock in practice. The receiver also computes its current velocity with accuracy up to one meter per second, accomplished by analyzing the Doppler effect shifts from the combined movement of the four satellites [Encyclopedia Britannica, 2022b].

Signal Distortion

GPS signals may experience distortion due to various factors. Consequently, numerous monitoring stations worldwide continuously track all visible GPS satellites to identify any deviations in their orbits. The data is scrutinized and modified to amend any orbital inaccuracies. This corrected information is transmitted back to the satellites via large antennas, allowing the satellites to rectify their own position calculations based on the base station's calculations, thereby mitigating the statistical errors of both [Encyclopedia Britannica, 2022b].

GPS is a dependable real-time positioning system; however, it is ineffective indoors due to the lack of satellite signals. Its high power consumption also presents a disadvantage for IoT applications like ours, rendering it unsuitable as a solution for production. Nonetheless, a GPS system is comparatively easy to set up and operate, which is why, within our scope, we will incorporate GPS for an alternative outdoor setup. This incorporation will enable us to optionally supply our simulator with GPS datasets as a fallback solution for demonstration purposes.

2.2.2 Beacon Proximity

The Bluetooth Low Energy (BLE) Beacon is a compact, wireless device powered by battery, transmitting a low-power signal utilizing the Bluetooth protocol. This makes it suitable for enclosed spaces like buildings. Nearby Bluetooth 4.0 devices, such as smartphones, can detect the broadcasted signal.

Devices can detect multiple signals from several BLE Beacons. Algorithms analyze the strength of these signals to calculate the precise location of the receiver. The accuracy of these calculations is often enhanced by utilizing a method known as multilateration [Ke et al., 2018, p.3]. Multilateration is a method to locate a target by measuring its different distances from multiple reference points.

The broadcasted signal comprises three primary identifiers: UUID, Major, and Minor. The UUID (Universally Unique Identifier) is a 128-bit string distinguishing Beacons within a network. Organizations usually define a unique UUID for their usage. The Major is a 16-bit string identifying a specific beacon or use case within the UUID-defined region. The Minor, another 16-bit string, allows further subdivision of regions or use cases. These three values together uniquely identify a beacon [Apple, 2014, p.2].

Bluetooth is a technology standard for wireless short-range communication between electronic devices. It operates on radio frequencies, allowing devices to communicate without direct line of sight, and is characterized by low power consumption and low costs [Encyclopedia Britannica, 2022a].

BLE Beacons employ ”Bluetooth Low Energy (BLE)”, a power-efficient variant of classic Bluetooth technology, also referred to as Bluetooth Smart, to connect devices within a short range, usually up to 10 meters.

However, Bluetooth usage can face interferences; obstacles such as people or objects can impact signal strength measurement accuracy, leading to less precise position estimation [Pascale et al., 2021, p.4-6].

2.2.3 LPWAN

A *Low Power Wide Area Network (LPWAN)* refers to technologies and protocols enabling long-range wireless networking for devices with low power requirements, predominantly used for IoT and Machine-to-Machine (M2M) applications. The critical requirements for LPWAN technology include:

- Minimal energy consumption by the networked devices.
- Low performance requirements on end devices and their software/hardware.
- Ability to cover extensive distances.
- Capability to support a large number of users on the network.

Compared to standard WLANs or mobile networks, data transmission rates within an LPWAN are considerably lower. It’s especially suitable for networking IoT application devices like battery-powered sensors, which can operate for several years without the need for battery replacement.

LPWAN is not a single standard but encompasses proprietary and standardized technologies, network protocols, and implementations. Various companies and standardization bodies actively contribute to the development of different concepts in this area [Luber, 2022].

2.2.4 Mioty

Mioty is a LPWAN technology specifically crafted for IoT applications and it stands for "My Internet of Things". It was developed by the *Fraunhofer Institute for Integrated Circuits (Fraunhofer IIS)* to facilitate robust and energy-efficient wireless networking of IoT devices across extensive distances.

The architecture of *Mioty* features multiple straightforward sensor nodes, equipped with cost-effective standard radio chips, and a powerful receiver station. This receiver station decodes the incoming radio signals and transforms them into digital information utilizing standard hardware components.

The distinguishing features of *Mioty* include:

- The ability to connect up to a million transmitters with a single receiver.
- The capability of extending connections up to 15 kilometers.
- A potential battery life extending up to 20 years.
- Utilization of the license-free frequency band at 868 MHz in Europe and 916 MHz in North America for radio transmissions.
- The integration of its patented core technology, Telegram Splitting, which divides the data transmission into numerous small packets and dispatches them across different frequencies and times, enhancing transmission reliability and energy efficiency.

In a smart city application, *Mioty* can be used for monitoring environmental parameters, managing and monitoring supply networks, and traffic or parking information. Its robustness and high interference immunity make it useful even under extreme conditions in indoor or outdoor areas, even if those are difficult to access [Luber, 2021].

2.3 Location Models

Location Models are data structures representing locations and the spatial relations between them. For our IoT application project, we need a suitable location model, allowing us to define distances, spatial ranges, and inclusions of locations.

Existing models use a specific type of coordinates, each with its unique set of attributes. They can support queries for positions, nearest neighbours, and ranges. Every model has its pros and cons regarding query processing and modelling effort. Classifying these location models helps in designing software according to its requirements [Becker and Dürr, 2005, p.20]. In the following sections, we will present the several coordinate types and location models based on them.

2.3.1 Coordinates

The text outlines three basic properties of coordinates. A coordinate x is an identifier that specifies an object's position in relation to a coordinate system. A coordinate system is a set X of coordinates. There exist two principal classes of coordinates: geometric and symbolic coordinates [Becker and Dürr, 2005, p.21].

2.3.2 Geometric Coordinates

Geometric coordinates represent positions as coordinate tuples relative to a reference system. They are categorized into global and local geometric systems. For instance, The World Geodetic System 1984 (WGS84) serves globally, whereas Cartesian coordinates of the active bat system are often applicable locally, in specific rooms fitted with such a system. These coordinates are useful to calculate distances and determine spatial relationships like overlap, touch, or containment [Becker and Dürr, 2005, p.21].

2.3.3 Symbolic Coordinates

Symbolic coordinates, on the other hand, depict positions as abstract symbols like sensor identifiers or room and street names. Unlike geometric coordinates, the distance between two symbolic coordinates isn't implicitly defined, and you cannot determine topological relationships like spatial containment without additional information regarding the symbolic coordinates' relationship. Symbolic location models offer this extra data on symbolic coordinates [Becker and Dürr, 2005, p.21].

2.3.4 Queries

The location model utilized should support queries that accommodate the use cases of an application. By understanding the most common types of requests, we can categorize location models in terms of their use case support and determine which location models are the most suitable for our application. A detailed summary of the different models and their supported query types is provided in Table 3.

Position Queries

Position queries take into account the precise coordinates of objects, both static and mobile, to execute tasks. All location models must contain this information, and the models vary in the ways they represent it [Becker and Dürr, 2005, p.22].

Nearest Neighbour Queries

Nearest neighbour queries pertain to the search for objects that are closest to a specific position. It is necessary to define a distance function on the coordinates for this type of query. The model should accommodate complex distances, considering obstacles or impediments that may affect the actual physical distance. Symbolic coordinates necessitate explicit definitions of distances [Becker and Dürr, 2005, p.22].

Navigation Queries

Navigation systems in vehicles rely on location models to determine paths between different locations.

In addition to understanding the geometry (e.g., of roads), it is crucial to comprehend the connectivity between locations such as the segments of roads at junctions. Different navigational tasks, like finding the shortest or fastest path, necessitate the modelling of various attributes, including distance, the maximum speed on a road segment, or the presence of stairs for wheelchair accessibility [Becker and Dürr, 2005, p.22].

Range Queries

Range queries yield all objects within a specified geographic area and have applications in various contexts, such as verifying room occupancy. To address a range query, it's essential to know the object positions and model the topological relation *'contains'* to discern whether a coordinate is within a spatial area. For geometric coordinates, this information can be inferred from the known geometry; however, for symbolic coordinates, this relation must be explicitly defined [Becker and Dürr, 2005, p.23].

Visualization Queries

Visualization, such as map drawing, is a significant application of location models. Maps facilitate tasks like positioning and navigation, either manually or automatically. While all model information can be visualized, a map typically necessitates a geometric representation of these objects at varying levels of detail [Becker and Dürr, 2005, p.23].

To support queries for position, range, and nearest neighbour, a location model needs to meet the following requirements:

- Object positions: The model must depict object locations using various coordinates and reference systems, both local and global, incorporating geometric and symbolic coordinates.
- Distance function: The model must define distances between spatial objects, including the size of a location (e.g., the length of a road segment).
- Topological relations: The model must present topological relationships between spatial objects, including spatial containment for range queries and connections for navigation services.
- Orientation: The model may need to account for the orientation of mobile objects in horizontal or vertical dimensions, which is important for applications that require knowledge of a moving object's direction or a user's orientation.

The requirement for minimal modelling effort is determined by factors such as:

- Accuracy: The model should portray the real world as accurately as possible. Dynamic objects require high update rates to ensure precise positioning.
- Level of detail: This refers to the granularity of the model; fine-grained models provide detailed location information, while coarse-grained models deliver broader, less detailed descriptions. A flexible model can accommodate both levels of detail.
- Scope: This represents the area covered by the model, which can range from local models depicting single rooms to global models encompassing the entire world.

2.3.5 Geometric Location Models

Geometric location models define locations by utilizing geometric figures. These models are employed when both global and local coordinate systems are applied. To effectively translate coordinates from one system to another, the position and orientation of local systems, in relation to other local systems or the global system, must be defined.

By applying geometric coordinates, one can derive the topological relation *'contained in'*. However, the model fails to derive the *'connected to'* relation, such as doors linking rooms, from location geometries, and this relation has to be explicitly modeled. This explicit modeling can then be utilized to refine the understanding of distances. For instance, it might incorporate the actual travel distance of a user as opposed to the direct distance reflected by the geometry.

Despite this, it remains sensible for a geometric location model to explicitly store the spatial containment relation due to the high cost of geometric operations [Becker and Dürr, 2005, p.24].

2.3.6 Symbolic Location Models

An alternative to geometric location models is symbolic location models. Multiple popular approaches exist for symbolic location models. In the following, we will present set-based models, hierarchical models, graph-based models, and a combination of these models. A detailed summary of their properties is listed in Table 1. The graph-based approach and hierarchical models effectively support the containment relation and are therefore suitable for range queries. The graph-based approach is optimal in scenarios where distance is crucial, such as in nearest neighbor queries and navigation. The combined symbolic location model integrates the advantages of all other symbolic model types but requires a higher modeling effort [Becker and Dürr, 2005, p.27].

Set-based Models

In the set-based model, a set L of symbolic coordinates forms the foundation. Different locations are defined by subsets of L. For example, the different floors and rooms of a building can be modeled using this set L. This model can identify overlapping locations and containment relations through calculations of set intersection. Additionally, the concept of neighborhoods describes sets of neighboring symbolic coordinates. The distance between two coordinates can be determined based on the smallest neighborhood containing them. The finer the distance granularity, the more neighborhoods can be defined, allowing for the modeling of paths between locations. However, the model has drawbacks such as a large number of resulting sets and substantial modeling effort.

Moreover, the set-based model doesn't allow for a quantitative definition of distance, limiting its application for queries related to spatial distances, such as nearest neighbor queries and navigation [Becker and Dürr, 2005, p.24-25].

Hierarchical Models

Hierarchical models consist of a set of locations ordered based on the spatial containment relation. When locations don't overlap, a tree-based model is applied. If overlapping is permitted, a lattice-based model is suitable, wherein intersections of locations are represented by separate locations with multiple parent locations. Hierarchical models can also be interpreted as sets of symbolic coordinates, with overlapping locations defined by the intersection of sets. These models naturally support range queries, where a range is defined by a location and its descendants in the hierarchy. A notion of distance is applicable to hierarchical models, based on the containment relation. To compare distances between locations, the concept of supremum is suitable for determining closeness. However, hierarchical models face limitations in representing interconnections between locations, potentially leading to counterintuitive interpretations of distance. The notion of distance in hierarchical models is qualitative, similar to set-based models [Becker and Dürr, 2005, p.25-26].

Graph-based Models

In graph-based models, symbolic coordinates define vertices in a graph, and edges are added between vertices that represent connected locations. The graph can be weighted to represent distances between locations. This model supports the definition of the topological relation "connected to" and allows for the explicit definition of distances between symbolic coordinates. It is suitable for nearest neighbor queries and navigation, where edges or nodes can bear additional information such as speed limits or restrictions. For range queries, the range is defined by an area, and objects within that area are identified by their distance from a reference location. However, the model faces limitations in explicitly defining larger locations. Different types of symbolic location models can be combined to overcome this limitation [Becker and Dürr, 2005, p.26].

Combination of Graph-based and Set-based Symbolic Models

Combining graph-based and set-based symbolic models introduces further advantages. The graph-based approach supports distance queries and the establishment of connected locations, while the set-based approach is suitable for range queries with explicitly specified locations like floors and buildings.

A combined approach involves a set-based component that consists of a set of symbolic coordinates. Locations, symbolizing various entities such as rooms, floors, buildings, etc., are subsets of this set and are utilized for range queries as discussed in the section about set-based models.

The graph-based component of this combined model connects locations using edges if a real-world connection exists between them. For example, a door between two rooms or stairs between two floors would constitute an edge. Edge weights can be used to model varying distances.

The hybrid model enables support for different topological relations and distances, as well as range and nearest neighbour queries. Furthermore, this model allows generation of views with different levels of detail, for instance, views displaying the rooms on a specific floor, the floors of a building, or just the buildings and the paths between them. These views can be used in scenarios requiring different levels of granularity, from very detailed to coarse-grained location information, thus, permitting the generation of smaller models that can cover large areas such as a city district [Becker and Dürr, 2005, p.26-27].

2.3.7 Hybrid Location Models

Another approach is the fusion of geometric location models with symbolic location models. These models integrate the advantages of set-based and graph-based models into a common symbolic model and additionally incorporate geometric information. Adding geometric information has two main benefits: it enhances accuracy and precision for distance-related queries and allows defining ranges for nearest neighbour queries with arbitrary geometric figures. Symbolically defined locations, in contrast, are constrained by a given structure. Table 2 shows how two types of hybrid models are distinguished: the subspace approach, storing geometric information for every location, and another approach that only stores geometric information for some locations, leading to partial subspaces [Becker and Dürr, 2005, p.27].

Subspaces

The foundation of subspace models is a blend of a symbolic model and geometric data. This model records the geometric extent of locations, which can be framed either using a global reference system like *WGS84* or local reference systems applicable within a confined scope like a building or a room. These models create subspaces by integrating coordinate systems within other systems, defining the placement and orientation of the embedded systems. This allows the translation and comparison of coordinates across different systems [Becker and Dürr, 2005, p.27-28].

Partial Subspaces

Unlike the full subspaces approach, partial subspaces only model the geometric extent for selected locations. For example an outdoor location may have a detailed geometric model, while buildings only have symbolic models. These symbolic models are incorporated into the global geometric model by associating them with geometric data.

This approach has advantages when dealing with a geometrically defined range query, such as a polygon on a city plan. Users within a building may only recognize a symbolic location, for instance, a specific room in a building. However, through the known geometric extent of the whole building, a user's location can be estimated in geometric terms. This approximated geometric position can then be juxtaposed with the geometrically defined range of the query to provide an answer.

This approximation technique has limitations, such as using geometric areas within a symbolically modelled building, but it can reduce the overall modelling effort [Becker and Dürr, 2005, p.28].

2.4 Simulation

In the following we would like to show what simulations are, how they are constructed and why they are useful for us. We also introduce a possible view on the process of evaluating a simulator. As we intend to build up on the work of [Pongratz, 2023], we orient our research in alignment with his research about simulation.

Table 1: Properties of Symbolic Location Models

Symbolic model type	Supported coordinate types	Modelling effort	Distance support	'Connected to' relation support	'Containment' relation support
Set-based	Symbolic	High	Limited	Yes	Good
Hierarchical	Symbolic	Low to medium	Very limited	No	Limited
Graph-based	Symbolic	Low to medium	Good / very good	Yes	Limited
Combined	Symbolic	Medium	Good / very good	Yes	Good

Adapted from [Becker and Dürr, 2005]

Table 2: Properties of Hybrid Location Models

Symbolic model type	Supported coordinate types	Modelling effort	Distance support	'Connected to' relation support	'Containment' relation support
Subspaces	Symbolic, geometric	High / very high	Very Good	Yes (if modeled explicitly)	Yes
Partial subspaces	Symbolic , geometric	High	Good / very good	Yes (if modeled explicitly)	Yes

Adapted from [Becker and Dürr, 2005]

2.4.1 Definition

Simulation is a multifaceted concept with no single-line definition, as its interpretation evolves and varies depending on the context of application. Early definitions from the 1960s proposed simulation as a representation of real-world aspects, either semantically or through a computer, a view seen today as context-sensitive and not applicable universally. Around the 1980s, this definition morphed to perceive simulation as the ongoing creation of models over time, irrespective of a real-world source. Thus, simulation is understood as a continuous process that generates a representation of something "real". This broader definition aligns simulation with machine learning algorithms, which operate similarly [Pongratz, 2023, p.5]. [Ören et al., 2023, p.2-4] introduce a Framework of M&S (Modelling and Simulation) as a more accurate tool for understanding the basic entities and relationships in M&S. This framework called *MSF (Modelling and Simulation Framework)* provides a theory-based set of definitions, clarifying the precise meanings of terms like *Model* and *Simulator*.

2.4.2 Entities of MSF

Each entity of the framework has a distinct function that enables the creation of a simulation. The basic entities are the source system, model, simulator, and experimental frame, as can be seen in Figure 1. Inter-relationships between these entities, namely the

Table 3: Properties of Location Models

Model type	Supported coordinates		Supported queries			Modeling effort
	symbolic	geometric	Position	Range	Navigation	
Set-based	Yes	No	Good	Good	Basic	High
Graph-based	Yes	No	Good	Basic	Good	Medium
Hierarchical	Yes	No	Good	Good	Basic	Medium
Combined symbolic	Yes	No	Good	Good	Good	High
Subspaces (hybrid model)	Yes	Yes	Good	Good	Good	Very high
Partial subspaces (hybrid model)	Yes	Yes	Good	Good	Good	High

Adapted from [Becker and Dürr, 2005]

modelling and simulation relationships, are also important [Ören et al., 2023, p.6].

Source System

A source system is a real or virtual environment that is the subject of modelling. It is seen as a provider of observable data and is represented by time-indexed trajectories of variables.

The data is accessed or procured through experimental frames that are significant to the modeller [Ören et al., 2023, p.6-7].

Behavior Database

The behaviour database is the collection of accumulated data through observation of the source system or through experimentation with it [Ören et al., 2023, p.6-7].

Experimental Frame

The experimental frame refers to a pre-specified set of conditions which are applied during the observation or experimentation with the system. It embodies the objectives which drive the simulation. For example, when simulating a forest environment with its numerous variables, the subset of lightning, rain, wind, smoke could be an experimental frame focusing on simulating forest fires triggered by lightning. Multiple experimental frames can be designed for the same system, and conversely, the same frame can be relevant to multiple systems [Ören et al., 2023, p.7].

Model

A model is defined as a system specification. It is essentially a set of directives, rules, equations, or constraints formulated to generate I/O behaviour. It is constructed with

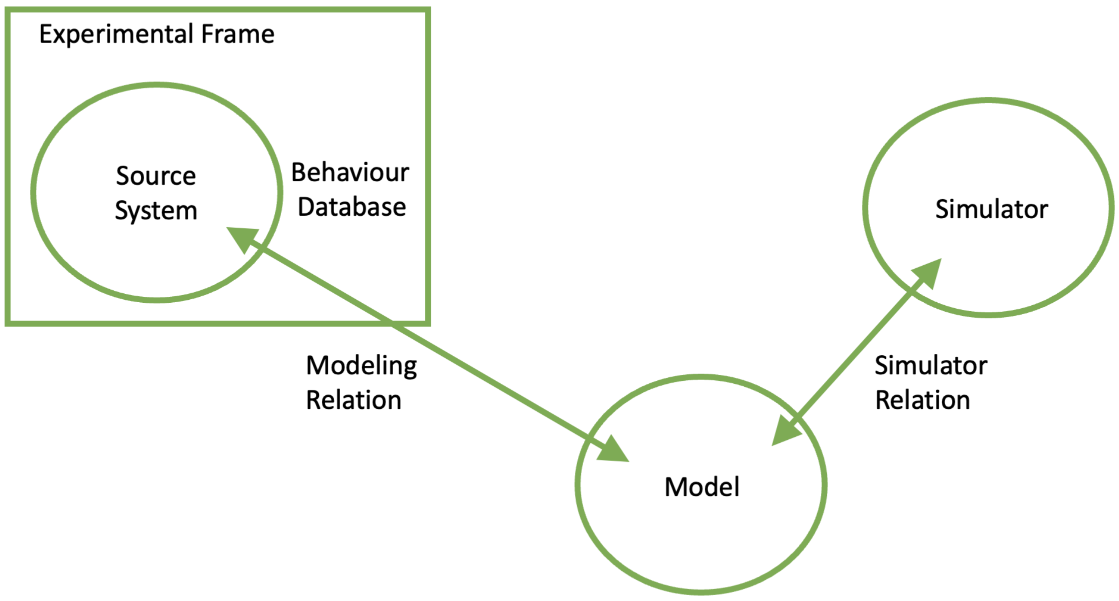


Figure 1: Entities and Relationships in the M&S Framework, adapted from [Ören et al., 2023, p.7]

state transition and output generation mechanisms to handle input trajectories and yield output trajectories based on its initial state. Another way to describe it is a function, which takes some input data and converts it into simulated output data [Ören et al., 2023, p.9].

Simulator

A simulator is a computational agent which can execute the model to generate the models output. Usually the simulator is a software application, but it can also stand for an algorithm, one or more processors or even for the human mind [Ören et al., 2023, p.10].

2.4.3 Advantages of Using Simulations

In the following we would like to highlight the advantages that arise from the use of simulations.

2.4.4 Simulation Reliability

For the evaluation of our application, we need a methodology to somehow assess the quality of the simulation. Just as there are many definitions of a simulation, there are also many approaches to evaluating a simulation. The authors of [Ören et al., 2023] elaborate a thoughtful approach based on the analysis of several other approaches. They use the term reliability for this:

> "(...) reliability comprises the property of a simulation model (and its respective simulator) to be reliable, i.e., the simuland (system being simulated by a simulation (...)) should be enough detailed and accurate, and faithfully

correspond to the system of interest being represented and/or developed. If the simulation model is reliable, conclusions and perceptions obtained from its execution can be reliably used to draw inferences about the simuland." [Ören et al., 2023, p.182]

From that definition we clearly can see that to determine the reliability of simulation we need to focus our attention on its model. Models are serving as simplifications and cut down the complexity for execution on simulators. The trustworthiness of these models is verified by ensuring each part of the model corresponds to an element in the actual system. Hence, validation and verification (V&V) activities are vital for establishing reliability of the simulation model.

Essentially, the reliability of a simulation model is determined by its level of accuracy in representing the actual system. In order for the model to be reliably consistent, both validation and verification are needed [Ören et al., 2023, p.182-183].

Validation

They define the term Validation as such:

> "Validation is the process of determining if a model behaves with satisfactory accuracy consistent with the study objectives within its domain of applicability to the simuland it represents." [Ören et al., 2023, p.185]

Simulation Models aren't flawless and can never be fully approved as valid. It is important to understand a model's validity as its applicability for a specific task. Is the model offering adequate data for the current needs? Validation should be seen as examining the model in various scenarios to figure out when and under what circumstances one can trust its use with satisfactory confidence [Ören et al., 2023, p.183-185]. Some authors distinguish three types of validity relevant to modelling and simulations Ören et al., 2023, p.189]:

1. Replicative validity, which refers to a model's ability to match the behaviour of the system it represents based on input and output, this is often tested by comparing the model to empirical data.
2. Structural validity, which is about a simulation model's ability to mimic the system's intermediate states and state transitions accurately.
3. Predictive validity, which is said to exist if the simulation model can provide accurate forecasts for situations that are not based on previous data.

Verification

In [Ören et al., 2023, p.185], they also define the term Verification:

> "Verification is the process of determining if an implemented model is consistent with its specification."

Verification is mainly about making sure that the simulation works properly. This includes checking that the functions of the simulation are implemented correctly and that they operate correctly, e.g. like how time moves forward, how it makes fake random numbers etc. [Ören et al., 2023, p.182-185].

2.4.5 Existing Simulators

As with any software system, there are two options when designing a simulation system: Either the software is developed from scratch or existing software solutions are used. In scientific research and also in software development it is preferable to build on previous solutions as long as this is compatible with the project goals. Thus we should consider a fundamental new implementation of a simulation software only if it is necessary for the realization of our project. In this work, we will not start from scratch, but we will build on the existing prior work of [Pongratz, 2023]. His work in turn builds on the existing *SmartSPEC* project, which we present below. The *SmartSPEC* project was started to implement a simulation system that was not available before. The intent behind it was to create a simulatior that could model and simulate people and their movement patterns, including schedules and event space, in spatially complex facilities. The researchers of *SmartSPEC* began their work by reviewing existing simulators and evaluating their applicability to their project.

The reviewed three primary areas: indoor and outdoor dataset generation, and generative models. Their literature review for indoor space datasets introduces multiple tools such as *Diasim*, *IE Sim*, *Persim-3D*, and *OpenSHS*. These tools, while effective in generating data for small-scale environments and individual activities, struggle to realistically model movements in larger environments.

Regarding outdoor environments, conventional models like *Levy flights* and *Preferential Attachment processes* aid in statistical analysis but fall short in creating authentic smart space trajectories. Other works in network-based trajectory generation have advanced the field, but their methodologies require significant configuration and lack consideration of complex semantic relationships. *Markov models*, while able to capture certain travel patterns, lack the granularity to model individual mobility.

The text also addresses generative models, notably *Generative Adversarial Networks (GANs)*. These models involve two neural networks: a generator producing synthetic data and a discriminator classifying this data as real or fake. Different variants of these models have been proposed, integrating features like LSTM networks, a "pooling" layer, multiple convolution and filter layers, and non-parametric copulas generative models. Despite these advancements, it remains challenging to enforce both space and people-event interaction semantics in the generated data. Training *GANs* is also notoriously difficult.

The makers of *SmartSPEC* aimed to overcome these limitations. *SmartSPEC* generates authentic *Smart Space* datasets and event-based semantic trajectories. Additionally, by learning the generation models directly from seed data, it reduces the manual effort which is typically needed in dataset generation [Chio et al., 2022, p.2]. The evaluation conducted in [Pongratz, 2023] demonstrated that the *SmartSPEC* project is a very good fit as a core component for our project.

2.4.6 SmartSPEC Simulator

SmartSPEC is the name of an opensourced software project which is focused on simulating humans and their movement behaviour in complex spaces.

It is a *Smart Space* simulator and data generator that creates customizable *Smart Space* datasets through the utilization of semantic models of *spaces*, *people*, *events*, and *sensors.* It utilizes machine learning techniques to analyze and understand the characteristics of the people and events within a space which is equipped with sensors. It uses an event-driven simulation method to generate plausible simulated datasets for the learned space [sma, 2023].

SmartSPEC Approach

The *SmartSPEC* system is comprised of two major elements: Scenario Learning and Scenario Generation. Scenario Learning employs initial data and prior knowledge to comprehend complex *SmartSPEC* concepts. Scenario Generation uses the acquired *SmartSPEC* data to produce a simulated dataset, which can be employed to generate datasets for smart spaces like sensor observation dataset or trajectory dataset. The definition of the scenario and its modifications are achieved using these data models.

The Scenario Learning component aims to achieve three objectives with the input sensor data and previous knowledge:

1. Extraction of semantic events and people that decode the observed phenomena
2. Learning *MetaModels* of events and people to facilitate the generation of new scenarios
3. Learning advanced *MetaTrajectories* to simulate the movement of people across spaces

These outcomes are then given to the Scenario Generation component.

The Scenario Generation component follows a three-step process. Initially, the Entity Generator utilizes the *Event* and *People MetaModels* to construct a new group of *Events* and *People*. The Synthetic Data Generator then uses this data and a configuration file to produce log files for data generators like Trajectory Generator and Sensor Observation Generator. These are then used to create *Smart Space* datasets.

SmartSPEC defines a smart space ecosystem as a 4-tuple of sets $(C; P; E; S)$, representing spaces, people, semantic events, and sensors. For generating new scenarios, *SmartSPEC* uses metadata models for events and people, named *MetaEvents* ME and *MetaPeople* MP. The system requires prior definitions for models of spaces and sensors as they are not learned or generated for new scenarios.

In the *Space Model*, a logical space in SmartSPEC is denoted by $C = (\mathrm{crd}; \mathrm{adj}; \mathrm{cap})$, where coordinates, neighboring spaces, and maximum capacity of the space are considered. The *Person Model* is defined as a 2-tuple $(\mathrm{TP}; \mathrm{aff})$, where TP indicates the expected entry and exit time of a person in the simulated space and aff is a probability model for the person's event affinity. The *Event Model* is defined by a 3-tuple $(\mathrm{TP}; \mathrm{Ce}; \mathrm{att})$, which explains when and where the event occurs and who can attend it. The *Sensor Model* is defined by a 3-tuple $(\mathrm{mob}; \mathrm{cov}; \mathrm{int})$, focusing on the mobility, coverage, and interval of sensor observations. Finally, *Time Profiles* describe the pattern of event occurrences or the entry and exit timings of people in the simulated smart space.

There are three operational modes of *SmartSPEC* to generate a smart space dataset. These modes differ in the degree of user involvement, automation, and control for the creation of *Models* and *MetaModels* for *People* and *Events*. Initially, the user defines the relevant Space and Sensor entities. Then, relevant *MetaModels* for *People* and *Events* are defined either manually or automatically using the Scenario Learning component. The automatically generated *MetaModels* can be modified by the user. After defining the

MetaModels, the user can manually or automatically generate the corresponding set of *People* and *Events*. Lastly, the user configures the Scenario Generation component and executes it [Chio et al., 2022, p.2-4].

SmartSPEC Scenario Learning

The *SmartSPEC* system employs sensor data and pre-existing knowledge about smart spaces to learn about semantic events and individuals. The methodology involves using change point detection algorithms to extract events and classify people based on the events they participate in. This information is used to create *MetaModels* for *Events* and *People*, termed as *MetaEvents/MetaPeople*, through the use of agglomerative clustering. Additionally, time-series methods are employed to create *MetaTrajectories* which are then used for the synthetic data generation.

A connectivity log dataset D, is defined as a sequence of triples $(dt; P; C)$, representing the person P present in space C at datetime dt. It can be derived from various wireless connectivity technologies like WiFi, Bluetooth beacons, or cellular. To obtain the most accurate output from SmartSPEC, the dataset should be extracted from the same scenario for which synthetic data needs to be generated.

Events and *MetaEvents* from the smart space data are learned using the connectivity log D. Individual events are identified and subsequently grouped into higher-order *MetaEvents*. For instance, a university campus might have a series of seminars, these could be initially extracted and later grouped into a seminar meta event.

A *MetaEvent* is formally a 4-tuple $(\mathrm{TP}; C_{\mathrm{me}}; \mathrm{att}; E_{\mathrm{me}})$, consisting of *Timeprofiles* and *Spaces* from which *Events* can be generated, a mapping to determine possible attendance set, and a set of *Events* characterized by the *MetaEvent*. The learning of *Events* involves using change point detection to determine when an *Event* ends and the next one starts. Agglomerative clustering with single linkage is used to cluster *Events* into *MetaEvent Models*.

Also learned are interactions between *People* and *Events* , and *MetaPeople*. The main characteristic of a person's behaviour is the set of *Events* they attend. *SmartSPEC* characterizes each *Person* with their attended *Events* and groups them into *MetaPerson models*.

A *MetaPerson* is a 3-tuple $(\mathrm{TP}; \mathrm{aff}; P_{\mathrm{mp}})$, containing a list of *Timerofiles*, *EventAffinity* which yields a probability model for attendance, and a set of associated *people*. *EventAffinity* and *Timerofile* for a *Person* are inferred from the set of attended *Events*. Like *MetaEvents*, *MetaPeople* are generated using agglomerative clustering with single linkage.

Finally *SmartSPEC* employs *MetaTrajectories* to describe potential paths a *Person* might take to move between two *Spaces*. Based on sensor data estimates are created for the start/end times that a *Person* spends in a *Space*. The time between consecutive entries is considered to define if the person is staying in the same *Space*. A person's trajectory is partitioned into sub-trajectories which start and end at a labelled *Space*. A path between two *Spaces* is selected from one of the previously learned trajectories [Chio et al., 2022, p.4-5].

SmartSPEC Scenario Generation

The Scenario Generation component comprises three steps:

1. Entity Generation creates a new collection of events and individuals by utilizing models that have been learned from prior *MetaEvents* and *MetaPersons*.
2. The Synthetic Data Generator merges these with spatial and sensor entities, forming a log dataset.
3. Various data generators like the Trajectory Generator and Sensor Observation Generator create trajectory and sensor observation datasets accordingly.

In *Entity Generation*, each *Event* is typified by a trio $(\mathrm{TP}; C_\mathrm{e}; \mathrm{att})$ which is extracted from a *MetaEvent* that has the attributes $(\mathrm{TP}; C_\mathrm{me}; \mathrm{att}; E_\mathrm{me})$. At first, a *MetaEvent* is chosen. Then a *Timeprofile* (TP) and a *Space* C_e are chosen at random. If C_e is unsuitable for hosting an *Event*, an alternate *Space* is sought. Attendees are selected by sampling normal distribution parameters for each *MetaPerson*.

The Synthetic Data Generation simulates individuals attending events in a virtual environment. For every individual, an event is chosen based on semantic models, and the person moves to attend the event while recording their movements. The selection process respects constraints such as space capacity, travel and wait times, attendee capacities, and prior engagements.

Finally, Trajectory and Sensor Observation Generation define a semantic trajectory as a series of 5-tuples $(P; E; C; \mathrm{ts}; \mathrm{te})$. They indicate the *Space* C and time t that trace a *Persons's* P movement to attend an *Event* E. This trajectory is extracted from a log by enumerating the timestamped sequence of visited *Spaces* and *Events*. Sensor observations are generated using synthetic data logs and the sensor model to identify captured movements [Chio et al., 2022, pp. 5–6].

SmartSPEC Evaluation

In our project evaluation we will need useful approaches to evaluate the quality of the resulting simulation. Since our work is based on *SmartSPEC*, their evaluation approaches are also interesting for us.

The authors discuss a methodology to evaluate how realistic smart space datasets are by comparing two datasets, one which has been recorded in the real world while the other has been generated synthetically. A *SmartSPEC* dataset is a log of sets of 3-tuples, where each tuple represents the presence of a person in a specific space at a certain time and date.

The first component of the methodology is called Space Occupancy Similarity. It involves comparing how similar the datasets are in terms of the number of unique people present in a particular space during a specific time frame. The time frame is divided into blocks, and the occupancy is computed for each block. The mean squared error is used to quantify the differences in occupancy between the datasets.

The second component is the People Trajectory Similarity. It compares the datasets based on the movement patterns of people across spaces. This involves the use of semantic trajectories, which are sets of spaces associated with specific times that a person occupies them. The trajectories are binned by time intervals to control for variation. A distance metric called the *Fréchet distance* is used to measure the dissimilarity between individual trajectories in the two datasets. The authors use a cost matrix to perform an unbalanced assignment problem that matches trajectories in a way that minimizes the overall distance between them.

To interpret these similarity metrics, the authors suggest splitting the real-world dataset into smaller parts and generating simulated datasets. By comparing the real and simulated datasets, they gather sets of values which represent the distribution of similarities. They perform an *ANOVA* test to determine if the simulated dataset can be distinguished from the real dataset based on the values obtained from the similarity metrics.

Finally, the authors conduct a Sensitivity Analysis to understand how resistant their trajectory distance metric is to small changes in the trajectories. They analyze various cases where there is a modification in the trajectories and how this affects the overall distance metric. They find that the *Fréchet distance* is resistant to small variations in trajectories, which is beneficial as it doesn't over-penalize minor discrepancies.

This analysis allows for a detailed comparison between datasets which helps to understand the realism of the smart space datasets [Chio et al., 2022, p.6-7].

2.4.7 Paul Pongratz's Work

In the context of the *WeideInsight* project, this paper deals with the implementation of work package 2: Simulation of tracking systems on farms.

The goal here is the development of a simulator that mimics how the tracking systems works on farms. This simulator shall allow to demonstrate how these systems work, test them, and adjust them as needed. This can help smaller farms to see how these systems might work for them before they invest a significant amount of money and time into the system. The simulator may also be suited to train farmers using examples from their own farms.

Also, companies that make herd management systems can use the simulator to test their systems in different setups and sizes, like using more or less sensors and different types of technology. They can also test data handling and the user interface.

The simulator will be set up with the layout of the farm, a number of cows, and a set of location sensors. It will demonstrate how the herd might behave, along with the basic features of the tracking system like the range of the sensors, their locating accuracy and type, how often they locate, and battery life.

The plan of *WeideInsight* is to simulate three kinds of location systems: Beacon-based, LPWAN-based, and a hybrid of the two. We'll also simulate a GPS positioning system to catch any issues with the hybrid system early on [WeideInsight, 2021, p.10].

The work of [Pongratz, 2023] has already contributed to the progress on the bigger project. In his work the focus was on creating an application that can mimic the behaviour of cows in a barn. The basis for this was strongly influenced by the research conducted in [Chio et al., 2022], adding to and expanding on their ideas.

SmartSPEC's original application was modified to work in a service-oriented architecture (SOA) and was made available through a simple REST-API as part of a cloud-based microservice setup. Other parts needed for this application structure were also designed and implemented as microservices. All these steps resulted in a solid, cloud-based microservice system, which proved that the *SmartSPEC* program can be used effectively to mimic the movements of cows in barns.

However, there are some limitations to the system. As it only supports input data from BLE Beacons, it can only be used for barns and can't be used in open fields like the farms pasture. It also cannot handle the concept of *SmartEvents* yet. The possibilities to customize the simulation and to interact with the demonstrator are very basic. But it does offer great opportunities for improvements. Our goal is to take advantage of these opportunities and keep improving the application, in order to create a more convincing and practical demonstrator, such that all the required features of this module of the *WeideInsight* project can be met.

Chapter 3

Requirement Analysis

The requirement analysis chapter aims to identify and document the essential needs and expectations of the stakeholders involved in the development of a simulator component within a smart farming application.

In Practice, we will first attempt to conceive a list of any requirements that we think of might become relevant. In a follow up meeting with the actual stakeholders we discuss and refine our suggestions.

Since our implementation is embedded in the *WeideInsight* project and we also tie in with the preliminary work of [Pongratz, 2023], further requirements arise from their specification that are relevant to our design approach and implementation. Those requirements will be deducted from those works.

In the following, we will elaborate the requirements of these two sources. We ultimately want to produce two lists of functional and non-functional requirements. Functional requirements describe all use cases that the software should support. These can be obtained for example through user stories or through use case analysis. We will use user stories for our requirement elicitation. Non-functional requirements, on the other hand, describe the quality criteria that certain aspects of the software must meet.

The resulting requirements will guide us in designing our application architecture, selecting the right tools for our implementation and serve as a basis for measurements in our evaluation. Finally they shall help us answer our thesis question.

3.1 Stakeholders

While the barn and pasture's design and size are based on a real farm, the simulation service artificially generates the cow movement data for that environment.

The stakeholders for this component include farmers, computer scientists, and domain experts. This chapter sets the groundwork for developing an efficient simulation by collecting the requirements of these stakeholders. We will now present the individual stakeholder groups and their interests in context.

3.1.1 Farmers

Farmers are the main users of our application. They are responsible for the management and care of the cows in their barn and on the pasture and therefore have a deep understanding of the challenges and needs of their cows. Their main interest is to optimise their farming operations for better productivity, health and overall well-being of their livestock. For farmers, tracking and monitoring cows' positions and movements through our application can provide valuable insights into the behavioural patterns of individual cows and the herd as a whole. By identifying abnormal behaviours or changes in activity levels, farmers can quickly identify potential health issues or problems in the cows so they can intervene promptly and provide appropriate care. Overall, farmers want to maximise their operational efficiency.

3.1.2 Computer Scientists

Computer scientists bring their technical expertise and knowledge to the table. They are responsible for the design, development and maintenance of the software system. They ensure that the application is technologically sound, reliable and scalable. They deal with topics such as system integration, data acquisition and analysis. They also focus on optimising the performance of the application to ensure efficient data processing and reliable communication between the tracking devices and the software platform. Their goal is to create an intuitive and user-friendly interface for farmers that provides actionable insights and visualisations based on the collected data.

3.1.3 Domain Experts

Domain Experts have in-depth knowledge of animal husbandry, animal behaviour and the entire agricultural sector. They understand the specific challenges farmers face and have expertise in implementing innovative solutions to overcome these challenges. They are interested in closely tracking and analysing cow positions and movements to gain meaningful insights into cow behaviour and welfare. Their expertise can contribute to the development of algorithms and analytical models that identify important behavioural patterns and provide actionable recommendations to farmers. They focus on combining their knowledge with technological advances to improve animal health, reduce stress and increase overall farm productivity. They ensure that the software solution aligns with smart farming best practices and effectively supports farmers in their decision-making processes.

By considering the interests and perspectives of these stakeholders, we can gain valuable insights for our requirements analysis and ensure that our application meets farmers' needs while incorporating the technical and domain expertise of computer scientists and domain experts.

3.1.4 Stakeholder Meetings

In the beginning and throughout the project, we conducted multiple meetings with several stakeholders to gather their feedback and to specify their desired requirements for the simulator. They provided valuable feedback, e.g., on what the User Interface should be capable of, including how the setup of areas and sensors for the barn and pasture would be realized, how the visualization of multiple moving cows should look, and how the problem of missing ground truth data at the time of conducting this project can be temporarily worked around by generating the ground truth data manually through the user interface, and more

3.2 Specification of Project WeideInsight

Due to the surrounding project, *WeideInsight*, in which our work is embedded, there are some additional requirements. These mainly refer to the simulation's need to process different data sources simultaneously. This means that the Beacons in the barn and the *Mioty* system on the pasture can be processed by the simulation at the same time, and its results are based on the combination of both.

3.3 Previous Work and its Implementation Constraints

The work of [Pongratz, 2023] already provided a list of requirements listed in his project. The requirements that were not met in the mentioned work have also been adopted for this project, with a few adjustments.

3.4 Resulting Requirements

In the following we present the summary of the collected requirements.

3.4.1 Functional Requirements

The functional requirements will be the main source for shaping the application design.

1. As a user, I want the User Interface to be intuitive and easy to use, so that I don't need a higher level of technical expertise to interact with it.
2. As a user, I want to see a detailed representation of the architectural layout of the barns inner space so that I can clearly see in which part of the barn the cows are represented.
3. As a user, I want to mimic the layout and conditions of a real farm in the simulation, so that I can better relate to a real-world scenario.
4. As a user, I want to create and configure events that take place in a real scenario, e.g. feeding the cattle at a certain time in a certain place, so that the simulation better matches the actual movement patterns and daily routines, making the simulation results more plausible.
5. As a user, I want to see a realistic simulation of multiple cow's movements, so that I can get an impression of how their movement could look like with real movement data.
6. As a user I want to simulate both indoor and outdoor farm environments, so that I can get a complete picture of the involved procedures.
7. As a user, I want to simulate varying number of sensors, cows, barn sizes and pasture sizes, so that I can try out the impact of different farm configurations.
8. As a user I want to save and retrieve previous simulation configurations and results, so that I can try out different configurations and their impact on the results easily.

9. As a user, I want to be shown quantitative differences between simulations, so that I can make effective comparisons.

10. As a user, I want to draw ground truth data into the virtual map manually to imitate the movement of cows by a hand-drawn movement path, so that I don't need to rely on the presence of real ground truth data to test the simulator.

11. As a user, when drawing ground truth data manually, I want the User interface to prevent me from drawing movements outside of the physical boundaries of the farm space, so that I don't generate incorrect inputs.

12. As a user, when drawing ground truth data, I want to adjust the recording interval, so that I can mimic the real sensor systems realistically.

13. As a user, I want to adjust the visualization of the spaces and sensors, in particular colors and transparency, so that I can see them clearly but at the same time they won't be be distracting.

14. As a user I want to have multiple options to visualize the movement of cows to suit different needs. In particular the cows should be representable as simple symbols, as numbers representing the amount of cows in a particular subspace, as numbers representing the ID of each cow and as a circle whose radius represents the amount of cows in a particular field.

15. As a user, I want the cows movement in the barn to be restricted in such a way that it matches the real physical restrictions of that barn, so that the resulting movements are highly realistic.

16. As a user I want to zoom and pan the virtual map, so that I can better focus on specific subsections of the simulated space.

17. As a user, I want to adjust the speed of the running simulation, so that I can either focus on smaller details when the simulations pace is slow, or that I can have more broader impression by running the simulation in timelapse.

18. As a user, I want to start and stop the simulation, so that I can visualize the results.

19. As a user, I want to fetch real-time data from from WeideInsights Simtrack component and feed it as input for the simulation, so that I don't need to depend on the manual collection and preparation of data in order to use the simulation.

3.4.2 Non-Functional Requirements

The non-functional requirements will help to guide us towards a sufficient quality of the application.

1. The simulator should be easy to install and execute.

2. The software should be simple to maintain and update.

3. The software should be designed in a modular way to allow for easy expansion and addition of new features.

4. The software should be compatible with common operating systems and devices used by farmers.
5. The UI should support the user in his understanding of how to use the application.
6. The UI should prevent the user from making incorrect inputs.
7. The UI should respond quickly to user inputs to support a seamless user experience.
8. The software should be able to generate and visualize simulations quickly and accurately without long waiting times during the process.
9. The system should be able to forward simulated data to the Simtrack component.
10. The software should provide a clear documentation.

Chapter 4

Solution Architecture

The architecture of a software system is the blueprint that defines its structure, behaviour, and overall composition. It serves as a foundational map that outlines how the system's components interact with each other and with external entities. The quality of this architecture can influence the efficiency, scalability, and maintainability of the application.

Since, in our particular case we build on the research of the previous works by [Pongratz, 2023] and [Ghadami, 2022] in order to extend and improve the existing application, we largely adopt the architecture of the previous work and extend certain aspects of it to implement the desired functionality from our requirements analysis.

In this chapter, we delve into the architectural design of our simulator, which employs a service-oriented architecture (SOA) in the back end and a web application as its front end. We will explore how this design permits modularity, ease of extension, and a separation of concerns to lead an agile and robust system. By the end of this chapter, we will gain an in-depth understanding of:

- The guiding principles and design patterns that influenced the architectural choices.
- How the service-oriented back end supports the versatile simulation scenarios required by users.
- The role of the web-based front end in providing an intuitive user interface.
- How data flows between the front end and back end components.
- How the architecture supports scalability to handle increasing data loads and computational complexity.

4.0.1 Architectural Overview

The system consists of two main components:

1. A front end based on a web application that can be accessed via the internet in any browser. This provides a user-friendly interface with which the parameters of the simulation can be configured. It also graphically visualises the results of the simulation.
2. A back end that works on a server. It processes, stores and provides the data. It also preforms the calculation of the simulation.

4.0.2 Design Principles and Rationale

A web application which is accessible through a browser was chosen for the user interface with the focus on user-friendliness. The intention is to facilitate access, as the user can access the application on different devices and operating systems, independent of the platform they use and without having to install specific software upfront. It also significantly simplifies the delivery of updates to the application. This way, the user can always use the latest version of the software without having to worry about manually installing updates, as he does not need to perform any installation steps on his side. This centralised approach also promotes scalability and allows for easier data management. Likewise, the functionality of the back end can be changed and extended at will without the user having to become active. However, the use of the application is dependent on an internet connection. In addition, there are security risks with web applications that must be considered. Data protection also plays a greater role here, as the data is stored in a central database. Since for the time being the software is to be used for demonstration purposes, the security aspects for our project are negligible and will be left for future work if needed.

4.0.3 Service-Oriented Architecture in the Back End

For the back end, a service-oriented architecture was chosen, specifically microservices. This approach has gained popularity in recent years. A big advantage over the classic monolithic architecture is the modularity. The functionalities of the software are divided into logically related units that then run independently as microservices. The advantage is that these microservices can be developed, rolled out and scaled independently of each other. This makes the system more flexible overall and also simplifies further development and maintenance.

The disadvantage is that it makes the application more complex, especially if many of these services have to communicate with each other. In our particular case, however, the number of services required is manageable. Theoretically, a monolithic approach would also be viable. However, because our simulator is part of a larger project, the more flexible microservice architecture shall be the better choice for the future and for the further integration of the system.

Role of Services

In the back end, we use six microservices for our simulation application, as shown in Figure 2.

- Central Gateway: The Central Gateway channels all incoming requests to the appropriate back end components. It essentially serves as the main entryway to the back end and streamlines interactions by consolidating access to various services.

- Data Storage: This microservice is a database for retaining the diverse data types used in the simulator, from basic configurations to complex sensor information. It is an object-relational database that offers the necessary flexibility and ease of integration.

- Configuration Manager: This service manages the simulator's configuration settings. It is focused on configurations related to *SmartSPEC*. The service offers endpoints

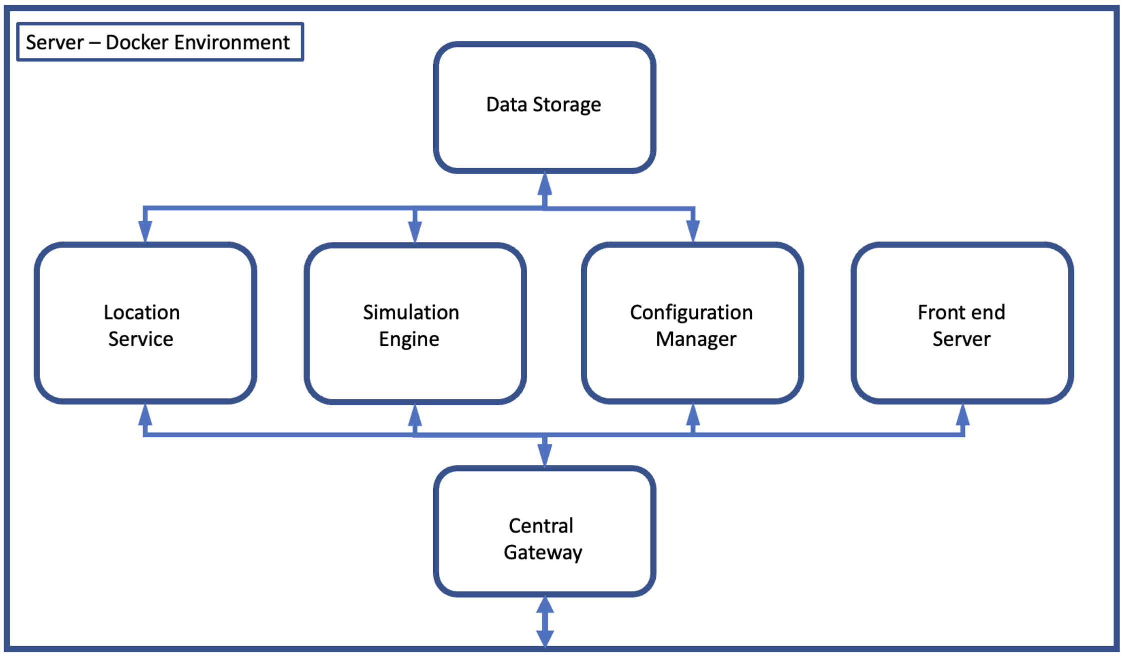


Figure 2: Back End Service Architecture

for retrieving and updating relevant settings in the Data Storage service.

- Simulation Engine: This service acts as an interface to the *SmartSPEC* system and is responsible for executing the actual simulation tasks. It allows for the import of training data and provides methods for storing and formatting simulation results.
- Location Service: This service responds to any queries about location-related data from the simulation outcomes.
- Front end Server: This service provides end-users with access to the front end. It serves an the interactive web application which is accessible with a browser.

Communication Mechanisms

Communication between the front end and the back end, as well as between the individual services themselves, is realized via a RESTful API through HTTP requests. RESTful APIs are widely adopted and easy for developers to become proficient with. Since they are stateless by nature, they simplify server design. Their ease of testing and debugging is well suited for rapid development, as is the case with our prototype.

4.0.4 Web-based Front End

The User Interface is implemented as a Single-Page Application (SPA). This approach provides a seamless user experience as it eliminates the need for full-page reloads, making interactions feel more fluid, similar to a traditional desktop application. The front end is built on fundamental web technologies, particularly HTML, CSS, and JavaScript. We also utilize VueJS, which is a lightweight framework for building modular and flexible web user interfaces with ease. This approach allows us to create a feature-rich user interface efficiently and simplifies the implementation and debugging of changes.

The UI is designed in a way to cater the requirements identified in the Requirements Analysis chapter. At its core, it supports the following main tasks with which the user will interact:

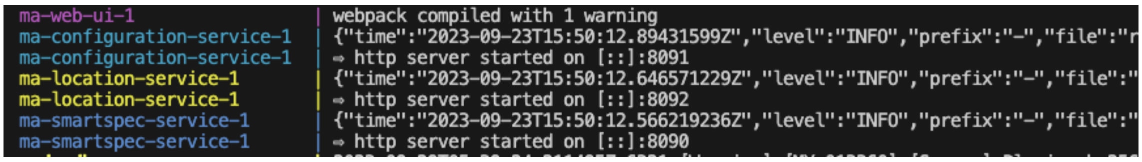
- Simulation Setup and Configuration: The UI provides options to configure numerous parameters which affect the simulation's behaviour. These include the spatial layout of the barn and pasture, its division into distinct spaces with sensors, and event configurations like milking schedules. The UI aims to simplify these configurations by providing instant visual feedback and minimizing the chances of invalid setups. Parameters for the learning phase and generation — such as the number of cows and the time frame for simulation — are also configured here. Users can then export and save these settings to the server through the UI.
- Executing Simulation Learning and Generation: Following configuration, the simulation engine learns from the provided settings and optional ground truth movement data. Upon completing the learning phase, the engine generates simulated data. Both steps may require some time to execute, and the UI guides the user through this process, providing status updates and error messages to address any issues that may arise.
- Visualizing Simulation Results: After data generation, results can be queried via REST-API from the server. An interactive map displays these results within a simulated time frame, allowing users to observe the movements of virtual cows throughout the farm. This visualization aims to provide deeper insights into cattle behaviour and to estimate the technology's potential benefits for the farmers.
- Additional Functionality: Beyond configuring the farm layout, the interactive map enables users to manually draw, record, and store cattle movements. This feature assists in generating ground truth data for the simulation service, especially when there is a lack of real data, as is the case during the timeframe of this work. It is not intended to replace real ground truth data but serves as a possible workaround for development and testing when real data is unavailable.

4.0.5 Data Flow

Data flow within the system is kept simple. Through the user interface, configuration settings are generated and exported to the server, where they are stored in the database. This is also true for ground truth data, which can either be collected from real sources or manually generated via the UI. Before storing the data in the database, several transformation steps are applied to convert the data into a format that the simulation engine can process. During the learning phase, the Simulation Service retrieves the stored configuration and input movement data to train the model, and subsequently stores the resulting learned model. In the generation phase, the learned model is used to generate simulated movement data, which is also stored in the database. This simulated data is then available for retrieval and further processing. In our case, the front end can request this data for visual representation, and it is also used for statistical analysis in our evaluation chapter.

4.0.6 Exception Handling and Logging Strategy

Ongoing development inevitably leads to errors and bugs that must be addressed. It is important that users are notified of issues affecting their experience. Any problems encountered during tasks that interact with the back end services trigger notification messages in the front end UI. For developers, the back end provides functionalities to stream output notifications from each service into a unified, color-coded terminal, as shown in Figure 3. This helps to quickly understand the communication flow among services and serves as a foundational debugging mechanism.



```
ma-web-ui-1                 | webpack compiled with 1 warning
ma-configuration-service-1  | {"time":"2023-09-23T15:50:12.89431599Z","level":"INFO","prefix":"-","file":"r
ma-configuration-service-1  | ⇨ http server started on [::]:8091
ma-location-service-1       | {"time":"2023-09-23T15:50:12.646571229Z","level":"INFO","prefix":"-","file":"
ma-location-service-1       | ⇨ http server started on [::]:8092
ma-smartspec-service-1      | {"time":"2023-09-23T15:50:12.566219236Z","level":"INFO","prefix":"-","file":"
ma-smartspec-service-1      | ⇨ http server started on [::]:8090
```

Figure 3: Combined Logging of all Microservices

4.0.7 Scalability and Performance Considerations

Although our focus is on demonstration, the modular, microservice-based architecture provides a solid foundation for future scalability. The performance of the simulation depends on various factors: the complexity of the configured simulation space, the volume of input data processed during the learning phase, and the volume of data generated. These computational tasks are dependent on the system resources available to the Simulation Service. Increasing available resources, especially CPU clock speed, the number of CPU cores, and RAM, will directly impact the execution time. In our example setup, which features a modest barn and pasture space on a server with 4 CPU-Cores and 8GB of RAM, computations take a few minutes to complete the learning and generation of two days of simulated time.

4.0.8 Deployment Overview

Having a reproducible setup is essential for both development and operation. The source code is managed via a Version Control System; in our case, Git, which is the de facto standard in most software development projects. For deployment, we use a Docker-based environment, a technology that has gained widespread adoption in recent years for its many advantages in this context. Installation and execution are highly automated and require only a few terminal commands to get the system up and running.

4.0.9 Summary

In this chapter, we have outlined the architecture of our cattle movement simulator.

The following chapter will focus on the details of implementing these designs. We will discuss the choice of technologies, programming languages, frameworks, as well as the actual code structure.

Chapter 5

Implementation

In this chapter, we delve into the implementation details of our cattle movement simulator. After laying out the architecture in the previous chapter, we now focus on the concrete technologies, programming languages, frameworks, and methodologies. The foundations for our architecture are adopted from the work of [Ghadami, 2022] and the particular implementation work of [Pongratz, 2023], in which the reasoning for the resulting architecture and its implementation can be read in finer detail. Our contribution is mainly an extension of those works. Nevertheless, we provide a concise sum up of the solution details and our contributed extensions in this chapter.

5.1 Back End

As discussed in the architecture chapter, the back end is realized in the form of multiple microservices which work together to provide the complete functionality of the whole application. The logical design pattern that those services implement is called Controller-Service-Repository Pattern (CSR).

Controller-Service-Repository Pattern

The Controller-Service-Repository (CSR) pattern is an architectural pattern widely used in software development, enabling a separation of concerns within the codebase. This separation allows for more maintainable, scalable, and testable code. Below, each component of the CSR pattern is outlined.

Controller The Controller acts as an intermediary between the user interface and the business logic (Service). It handles the input from the UI, processes it (with the help of Service), and returns the output to be displayed.

Service The Service layer, situated between the Controller and Repository layers, contains the core business logic. It processes the input received from the Controller, interacts with the Repository to perform CRUD operations, and returns the results.

Repository The Repository layer interacts with the database and performs CRUD operations. It retrieves the data from the database, processes it as needed, and sends it to the Service layer.

Advantages

- **Separation of Concerns:** The CSR pattern promotes the separation of concerns, which is generally a desirable goal in software development, as it makes the code more modular and manageable.
- **Maintainability:** The system becomes easier to maintain, as it is easier to update or modify one layer without affecting others.
- **Scalability:** Scaling the system horizontally becomes easier, as different layers can be scaled independently.

Disadvantages

- **Complexity:** The introduction of multiple layers leads to increased complexity, which increases the learning curve for new developers to understand the flow and the interaction between the different layers.
- **Performance Overhead:** The interaction between different layers can introduce performance overhead due to the additional processing involved. This might be a point to consider in very large projects but is negligible in our case.

Application in Our Project In our project, the CSR pattern is applied in each of the microservices and helped in maintaining a clean and organized codebase. The Controller layer always handles user requests and communicates with the Service layer, which encapsulates the core business logic of our application, interacting with the Repository layer for data operations.

Programming Language Golang

All of the back end services are implemented with the Go programming language, which is often referred to as Golang. The language is quick to pick up for developers and was created with the requirements of back end development in mind.

Echo Framework

Each of the services is wrapped in a boilerplate code-structure which stems from the utilization of a framework called Echo.

Echo is a high-performance, extensible, and minimalist web framework for Go. It facilitates the creation of robust and scalable RESTful APIs, web services, and HTTP servers.

While the use of such a framework increases the complexity at first, once understood it facilitates the development and maintenance of services as it provides a clear code structure and allows the developer to focus on implementing the business logic of the application.

Echo's Core Features Echo offers a numerous features to facilitate web development. The core features utilized in our project include:

- **Fast HTTP Routing:** Echo provides a fast and efficient router that maximizes the performance of HTTP services.
- **Middleware Support:** Middleware facilitates the processing of requests and responses and to manipulate the HTTP context.
- **Data Binding:** Echo supports the binding of payload data and helps to validate the incoming HTTP requests.

Application in Our Services The feature set and the clear documentation of Echo helped establishing a coherent code structure and an efficient mechanism to handle HTTP requests and route them to their respective controllers. It has helped with the clean separation of concerns and simplified the integration of the Controller-Service-Repository pattern.

5.1.1 Services

In this section we focus on relevant details pertaining to the several microservices.

Gateway Service

The Gateway Service functions as the central access point to the back end services and is implemented as a straightforward router in Golang, utilizing the Echo Framework to forward all API requests to the corresponding services.

Database

The back end database is a relational MySQL Database. Given the relational nature of the data processed and the seamless integration of MySQL Database in a Docker environment, it is an optimal choice. Its setup and usage are straightforward.

Configuration Service

This service provides endpoints to store and retrieve the simulation configuration. The configuration is stored as JSON-formatted strings in a separate database table, "smart-spec_conf", which contains columns for sensors, spaces, events, and learning and generation settings. The service exposes the following endpoints:

GET `http://serveraddress:8080/api/configuration`

POST `http://serveraddress:8080/api/configuration`

Simulation Service

The Simulation Service is central to the application, encapsulating the underlying *SMART-Spec* project. It provides endpoints to execute essential functions of that application relevant to our use cases. Training the simulation model is a preliminary step, and the method of training depends on the configuration established initially. There are two predominant methods of training the model:

- Learning by providing ground truth data: The learning is based on a setup of sensors and spaces together with a set of movement-data of cattle in those spaces during a specific time-period. The data will be stored and accessed in the Database in the table "learning_measurement", and contains columns for a timestamp at which a record was tracked, a client-id of the cow which this record relates to and an identifier of the closest sensor to the cow which recorded it. The learner uses the movement data as ground truth data, from which it learns the movement patterns in the provided spaces. Based on that the generator will then try to come up with plausible entities, that behave similarly as those that were used for the learning stage.
- Learning by providing events and entities: The learning is again based on a setup of sensors and spaces, but this time no ground truth data needs to be provided. Instead the configuration is extended by providing a set of Events and Persons, as well as *MetaEvents* and *MetaPersons* (more details can be found in the documentation of *SmartSPEC* [sma, 2023]). Based on these events that take place in a specific space at a specific time, the learner will train a model in which the configured entities try to move through the environment in such a way that they can attend at the events that are relevant for them at the right time.

Depending on the setup's complexity and the available hardware resources, the training duration typically spans at least several minutes.

After the learning phase is finished, the generation phase can be started, which will also take some time. During both of these phases, the status of execution can be retreived with an API call. Once the generation process is finished, one can then persist the results to the database, so that the generated data is available to be fetched by the location service. The Endpoints for these tasks are as follows:

POST `http://serveraddress:8080/api/learning/start`

GET `http://serveraddress:8080/api/learning/status`

POST `http://serveraddress:8080/api/generation/start`

GET `http://serveraddress:8080/api/generation/status`

POST `http://serveraddress:8080/api/generation/persist`

Location Service

The Location Service offers an endpoint enabling the client to request the generated results within a specified timeframe. The data is stored in the "location" table in the Database and can be accessed with the following request:

GET `http://serveraddress:8080/api/locations?start=<timestamp>&end=<timestamp>`

Frontend Service

As detailed in the preceding chapter, the Front end Service is a simple HTTP server, grounded in the Echo Framework, delivering the entire front end as a Single Page Application (SPA) to a client's browser. Subsequent communications between the SPA and the back end are facilitated through the APIs of the other services.

5.1.2 Setup, Deplyoment and Debugging

The deployment and orchestration of the various services is simplified through Docker. Docker operates by encapsulating software into containers. At the core of Docker is the Docker Engine which is responsible for building and running Docker containers. An Image in Docker is a stand-alone, executable package that includes everything required to run a piece of software, including the code, a runtime, libraries, and system tools. Containers are runtime instances of an image, running in isolation and managed by the Docker Daemon, running in isolation and managed by the Docker Daemon, which supervises the building, running, and managing of Docker containers.

The interaction with the Docker Daemon happens through the Docker Client, a command-line interface. While manual orchestration is feasible, Docker Compose, a tool for defining and running multi-container Docker applications, offers a more streamlined approach by using a YAML file to configure the application's services, networks, and volumes, allowing the deployment process to be executed with a single command.

To further abstract the use of Docker Compose, we employ Make, a build automation tool traditionally used for managing dependencies, which allows the encapsulation of Docker Compose commands within a Makefile. This abstraction simplifies the execution and management of Docker Compose configurations, enabling concise and human-readable command executions, such as "make up" to build and start all services and "make stop" to halt all running containers, minimizing the scope for errors.

Debugging is possible by redirecting the standard output of the running containers to the hosts terminal. Given the simultaneous execution of multiple services, consolidating the outputs with color-coded prefixes assists in differentiating between services and improves the clarity of the debugging process. This logging mechanism is activated by executing "make logs" after the services have been started.

In essence, Docker provides a streamlined platform for us to deploy and run our application and helps us to ensure that the software and its dependencies are cohesive.

5.2 Front End Implementation

In the following we want to focus on the implementation details of the front end, which is where the majority of development for this work took place.

5.2.1 SPA Architecture Overview

The front end is a Single Page Application (SPA). This allows our application to load a single HTML page in the clients Browser and then dynamically update the content as the user interacts with the app. This approach offers a smooth and interactive user experience because it frees the user from having to perform full page reloads which tend to be disruptive for the user experience. It makes the interaction seem more fluid and responsive. The front end application contains already the whole functionality from the beginning and only exchanges data with the server through the defined API-Endpoints.

5.2.2 API Integration

The API integration is a fundamental part of the communication between our front end and the back end services. For this task we utilize Axios, which is a promise-based HTTP client for JavaScript that simplifies the handling of requests and responses. Axios is preferable over the traditional AJAX in our case as it is easy to use and provides many capabilities, like intercepting requests and responses, cancelling requests, and automatic JSON data transformation. It facilitates the transmission of configuration data, the retrieval of simulation results and helps managing user interactions in an efficient manner.

HTML, CSS, JavaScript

The fundamental technologies behind the SPA are HTML, CSS, and JavaScript. HTML structures the content of the web app, CSS provides styling, and JavaScript handles the dynamic aspects of the user interaction, facilitating the creation of a responsive and interactive user interface.

5.2.3 TypeScript Integration

We use TypeScript, a superset of JavaScript that offers enhancements like static typing, which improves code reliability, detectability of errors during compile-time, and overall maintainability. In a build step it gets transpiled into JavaScript to ensuring browser compatibility.

VueJS Framework

To avoid the rigidity and bloat of vanilla JavaScript, we adopted the VueJS framework, a lightweight and adaptable alternative to React, which has been used in the previous iteration of our demonstrator. It offers modularity, flexibility, ease of debugging, and efficient implementation of feature-rich interfaces.

5.2.4 Pinia for State Management

Effective state management is important for the seamless operation of any SPA.

We leverage Pinia as state management solution in our VueJS front end due to its simplicity, intuitive API, and seamless integration with Vue's Composition API. Its developer-friendly experience and ease of setup contributed in keeping the codebase organized. It allows independent components in the UI to perform changes to the global state of the app, while at the same time components can react to any changes of global data that is relevant to them. While users interact with the application, Pinia ensures that the app's state is maintained accurately in order to provide a consistent and reliable user experience.

5.2.5 Error Handling and User Notifications

Effective error handling and user notifications enhance the user experience. Errors or issues that arise during user interaction are handled to avoid disruption in user experience. The front end displays errors to users through intuitive notifications, so users stay informed about any issues or actions they need to take.

5.2.6 User Interface Design

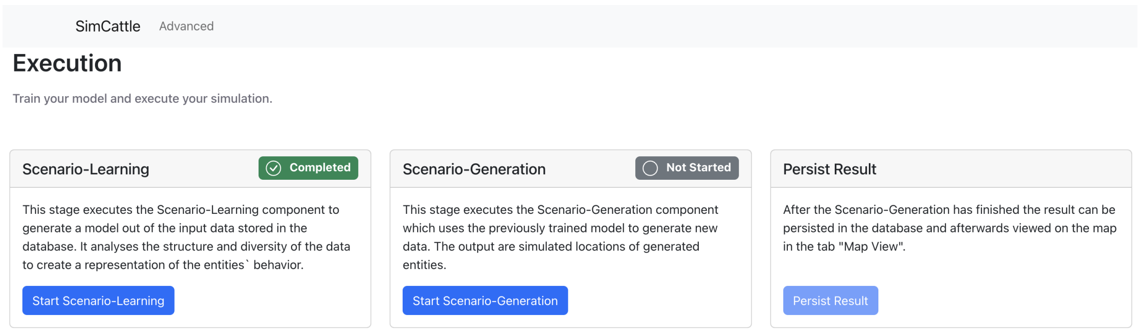
The design of the UI is is focused on a seamless and intuitive user experience. It is tailored to accommodate the requirements gathered in the Requirement Analysis chapter, incorporating user-friendly navigation, clear visual hierarchy and responsive design. It aligns aesthetically and functionally with the overall objectives of the application, focusing on user-centric design principles to enhance user engagement and satisfaction.

5.2.7 User Interface Overview

In the following we explain further details on how the UI is meant to be utilized by the user.

Main Page

On the Main Page of our application the user can find information about the simulation procedure and the current simulation status. The user can initiate the execution of the procedures manually in this section, which is shown in Figure 4.



Execution

Train your model and execute your simulation.

Scenario-Learning

This stage executes the Scenario-Learning component to generate a model out of the input data stored in the database. It analyses the structure and diversity of the data to create a representation of the entities` behavior.

Scenario-Generation

This stage executes the Scenario-Generation component which uses the previously trained model to generate new data. The output are simulated locations of generated entities.

Persist Result

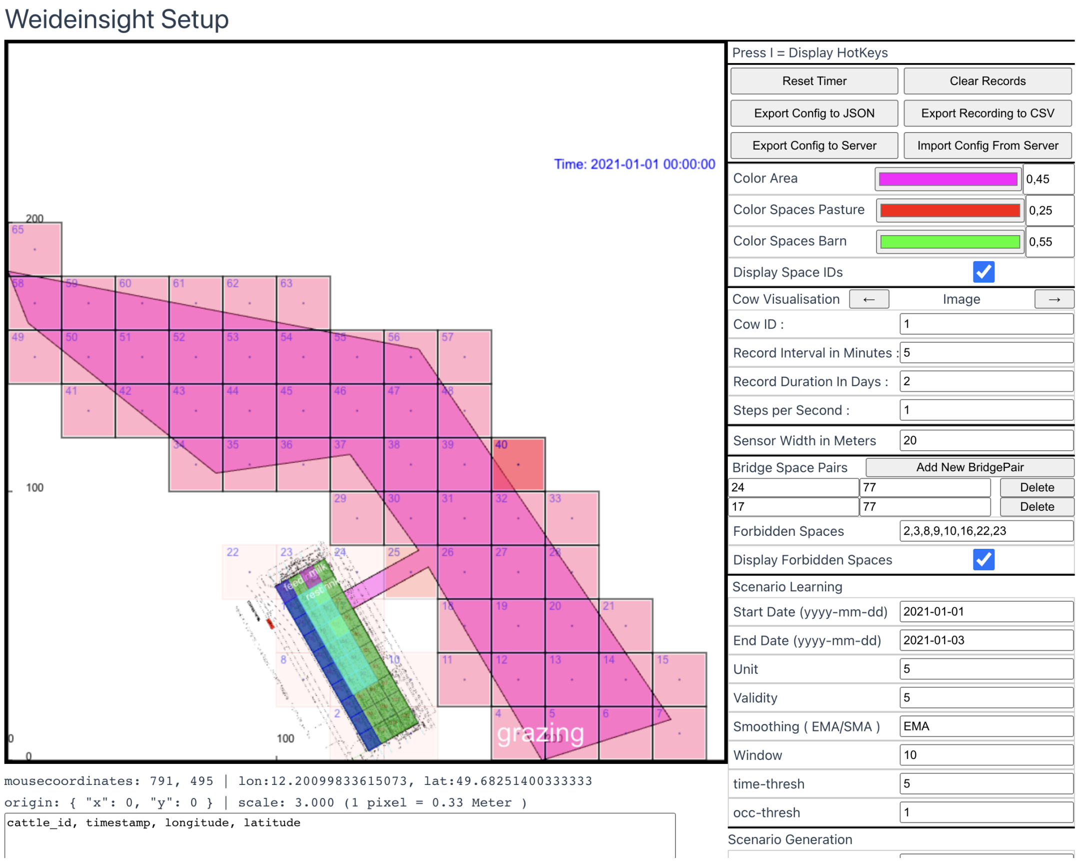
After the Scenario-Generation has finished the result can be persisted in the database and afterwards viewed on the map in the tab "Map View".

Figure 4: Main Page

Advanced Page

The Advanced Page, shown in Figure 5 serves multiple purposes. Its main purpose is the configuration of the simulation settings. The Left side displays an interactive map section in which a geographical representation of a concrete farm layout with barn and pasture spaces can be inspected. It allows panning and zooming to focus on the barn as shown in Figure 6. On the right side of the map there is a list of adjustable settings. These include the configuration of spaces and sensors, scenario learning and scenario generation parameters and the configuration of events. Further there are settings for the visualisation of the interactive map. Finally there are settings to support the user with manual creation of ground truth data for the cattle movement behaviour.

An example flow of interaction is described next. The simulation engine operates with spaces and sensors, not with geographical coordinates.


Weideinsight Setup

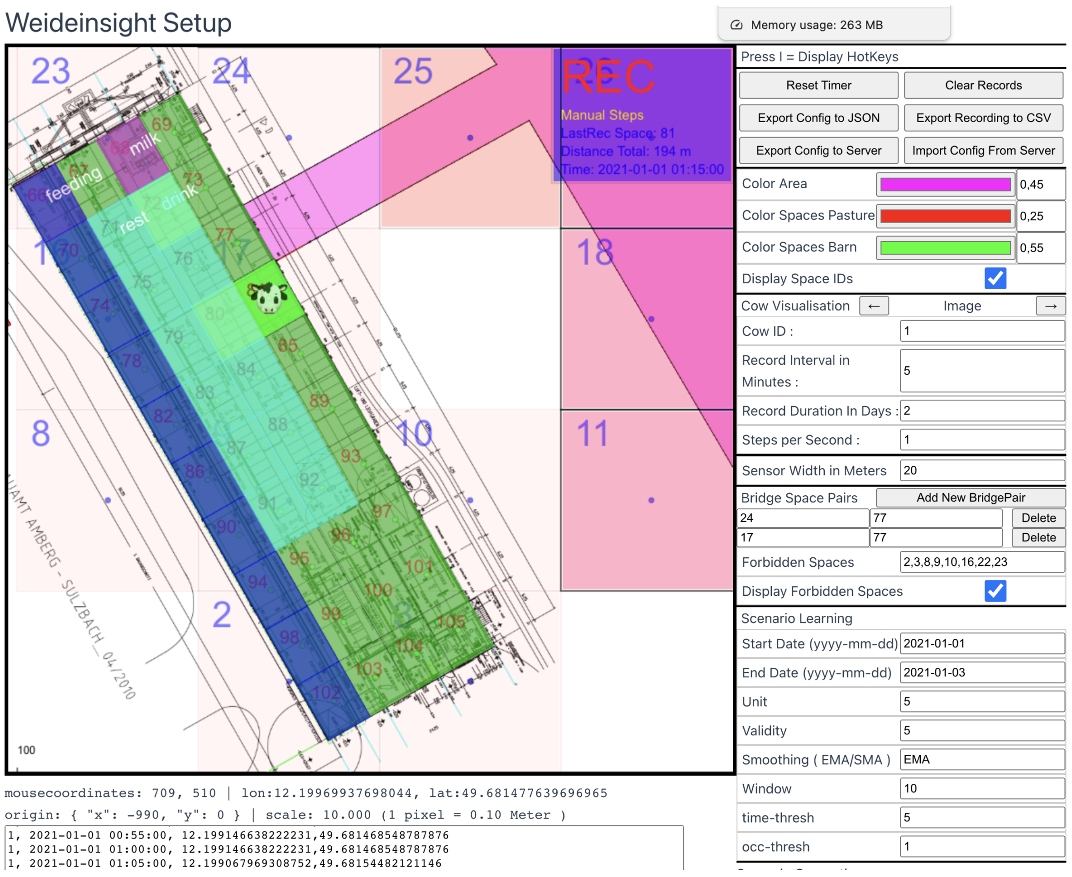
```
mousecoordinates: 791, 495 | lon:12.20099833615073, lat:49.68251400333333
origin: { "x": 0, "y": 0 } | scale: 3.000 (1 pixel = 0.33 Meter )
cattle_id, timestamp, longitude, latitude
```

Press I = Display HotKeys

Reset Timer	Clear Records
Export Config to JSON	Export Recording to CSV
Export Config to Server	Import Config From Server
Color Area	0,45
Color Spaces Pasture	0,25
Color Spaces Barn	0,55
Display Space IDs	✓
Cow Visualisation	← Image →
Cow ID :	1
Record Interval in Minutes :	5
Record Duration In Days :	2
Steps per Second :	1
Sensor Width in Meters	20
Bridge Space Pairs	Add New BridgePair
24	77 Delete
17	77 Delete
Forbidden Spaces	2,3,8,9,10,16,22,23
Display Forbidden Spaces	✓
Scenario Learning	
Start Date (yyyy-mm-dd)	2021-01-01
End Date (yyyy-mm-dd)	2021-01-03
Unit	5
Validity	5
Smoothing (EMA/SMA)	EMA
Window	10
time-thresh	5
occ-thresh	1
Scenario Generation	

Figure 5: Advanced Page for Configuration, Visualisation and Generation of Manual Data

From the simulation's point of view, a cows position is either within one of these areas or not. Since the pasture area spans several hundred metres in size, one first wants to divide the pasture area into logical subsections so that the simulation can determine more precisely in which of these sections a cow is located. For this purpose, the user chooses a "sensor distance" in meters. The application then calculates a subdivision of the pasture into square subsections, which have a side length corresponding to this sensor width. These squares represent the spaces and at the same time the sensors. Each sensor is responsible for one space. Sensors and spaces are automatically assigned the same unique identification number. The default setting is an edge length of 20 meters. The sensors generated in this way represent the location outside the barn in the open, representing the Mioty system. The positioning inside of the barn is representing the Beacon system. The configuration of the barn areas and sensors is currently configured manually in the source code, where it can be adjusted if desired. In the default setting for our showcase



Memory usage: 263 MB

REC
Manual Steps
LastRec Space: 81
Distance Total: 194 m
Time: 2021-01-01 01:15:00

Press I = Display HotKeys

Reset Timer | Clear Records
Export Config to JSON | Export Recording to CSV
Export Config to Server | Import Config From Server

Color Area	🟪	0,45
Color Spaces Pasture	🟥	0,25
Color Spaces Barn	🟩	0,55
Display Space IDs	✓	
Cow Visualisation	← Image →	
Cow ID :	1	
Record Interval in Minutes :	5	
Record Duration In Days :	2	
Steps per Second :	1	
Sensor Width in Meters	20	
Bridge Space Pairs	Add New BridgePair	
24	77	Delete
17	77	Delete
Forbidden Spaces	2,3,8,9,10,16,22,23	
Display Forbidden Spaces	✓	
Scenario Learning		
Start Date (yyyy-mm-dd)	2021-01-01	
End Date (yyyy-mm-dd)	2021-01-03	
Unit	5	
Validity	5	
Smoothing (EMA/SMA)	EMA	
Window	10	
time-thresh	5	
occ-thresh	1	

```
mousecoordinates: 709, 510 | lon:12.19969937698044, lat:49.681477639696965
origin: { "x": -990, "y": 0 } | scale: 10.000 (1 pixel = 0.10 Meter )
1, 2021-01-01 00:55:00, 12.199146638222231,49.681468548787876
1, 2021-01-01 01:00:00, 12.199146638222231,49.681468548787876
1, 2021-01-01 01:05:00, 12.199067969308752,49.68154482121146
```

Figure 6: Zoomed View on the Barn Space

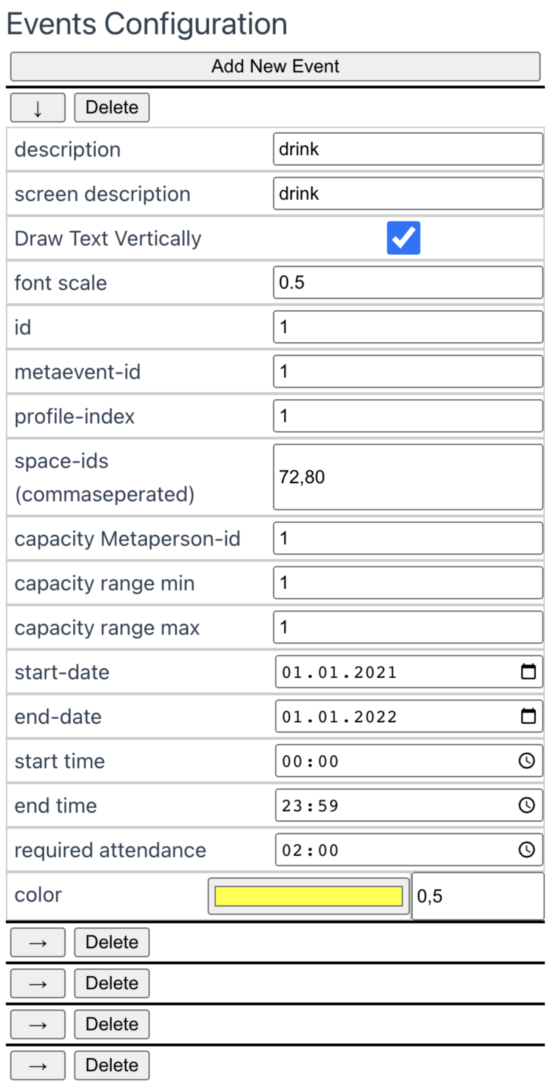
farm space, this results in 50 sub-areas in the barn and 65 sub-areas in the pasture area.

In the next step, the user configures the list of "Bridge Space Pairs". This list allows the configuration of spaces in the barn that are *connected* to spaces in the pasture. The simulation then considers these as connected neighbours that a cow can traverse. Since the rasterization algorithm of the pasture calculates a few too many spaces for reasons of complexity and these should not belong to the accessible area, the user can enter the IDs of these spaces in the list of "Forbidden Spaces" in the next step. These spaces then are not considered by the simulation and are also masked visually.

If the user prefers to include events in the configuration, he can now do so. The UI provides a variable list of events, in which some events are already preconfigured by default. However, the user can expand this list, delete or adapt events as shown in Figure 7. The configured events are then highlighted on the map and labelled with a text.

Finally, as can be seen in Figure 8 the user can configure the settings for scenario learning and scenario generation. The entire configuration can then be exported and transferred to the server.

If the user decides to incorporate events, he can now start the scenario learning, as in this case no ground truth data is needed. If, however, he does not configure any

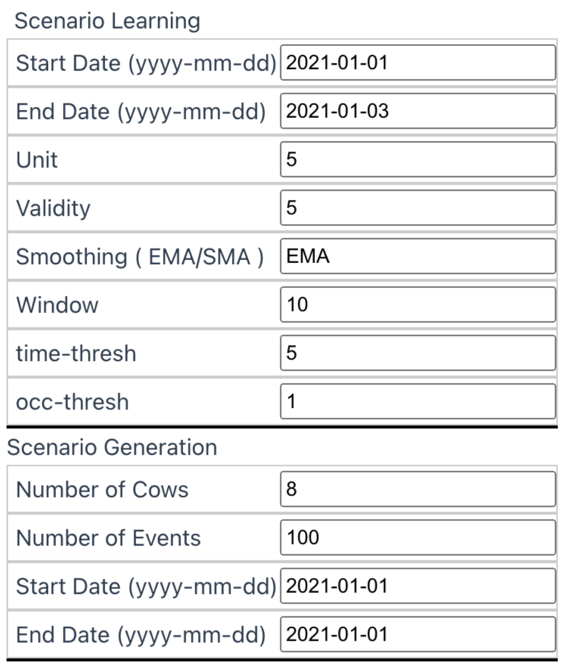

Events Configuration

description	drink
screen description	drink
Draw Text Vertically	✓
font scale	0.5
id	1
metaevent-id	1
profile-index	1
space-ids (commaseperated)	72,80
capacity Metaperson-id	1
capacity range min	1
capacity range max	1
start-date	01.01.2021
end-date	01.01.2022
start time	00:00
end time	23:59
required attendance	02:00
color	🟨 0,5

Figure 7: Setup of SmartEvents in the User Interface

events, then a data set of movement data must be provided as ground truth data to the simulator. Since in our specific case due to technical obstacles we had to do without data, the tool offers a possibility to manually draw and record movement data directly onto the map interactively and then export it to the server. For this purpose, a recording mode is started in which the user moves his mouse over the map view while a virtual clock simulates the progression of time. The position of the mouse and the underlying sensor at a certain point in time is recorded, which represents the movement of a cow with a specific ID. We can thus use that imitated data to evaluate the plausibility of the simulation itself in the evaluation chapter.

When the learning phase and the generation phase of the simulation are completed, the UI can be used to query the location service for simulated cattle movement data in a specific time range and to visualise it on the map.



Scenario Learning

Start Date (yyyy-mm-dd)	2021-01-01
End Date (yyyy-mm-dd)	2021-01-03
Unit	5
Validity	5
Smoothing (EMA/SMA)	EMA
Window	10
time-thresh	5
occ-thresh	1

Scenario Generation

Number of Cows	8
Number of Events	100
Start Date (yyyy-mm-dd)	2021-01-01
End Date (yyyy-mm-dd)	2021-01-01

Figure 8: Setup of Scenario Learning and Scenario Generation

5.3 Summary and Next Steps

In this chapter, we provided detailed insights about the implementation details of our application. Next, we will evaluate how well the simulator works by testing it in the evaluation chapter.

Chapter 6

Evaluation

6.1 Introduction

This chapter is dedicated to the systematic evaluation of the developed simulator's capacity to plausibly replicate real-world trajectories of moving cattle in a farm space with a hybrid setup of tracking technologies. We assess the simulator's plausibility to ensure its reliability and credibility. The evaluation is based on a thorough comparison between the simulator-generated data and the real-world data with a focus on similarities and discrepancies in the movement trajectories. The real-world data consists of movement trajectories represented by unique cattle IDs, timestamps, sensor IDs, and geographical coordinates.

Disclaimer There is an important disclaimer that we have to emphasize up front as it has an impact on how to interpret the evaluation outcomes that we are going to present:

The evaluation for the present work was initially based on two basic assumptions, which unfortunately could not be fulfilled at the time of the evaluation:

1. The availability of real data sets from the *Mioty* system which was set up on the experimental farm during the work. Unfortunately, due to technical difficulties and resulting delays with our project partners, no real data sets from the *Mioty* system were available at the time of the evaluation.
2. The functionality of the *SmartSPEC* simulator so that the simulation can be carried out correctly. Even though the functionality was ensured at the beginning of the work, technical problems with the *SmartSPEC* simulator unfortunately occurred at the time of the evaluation and could not be resolved in time so the simulations could not be carried out as planned.

Both problems should be solved in the future, but unfortunately not within the time frame of this thesis. Even though these problems put a spanner in the works of the evaluation, we made use of two alternative options to still be able to evaluate and answer the core of our thesis question.

1. We tilted the focus of the project from data simulation to data generation. We have extended the front end with additional features so that the user can draw and record manual cow movements and export them to the system. To recreate plausible

movement data, we have worked closely with domain experts who are very familiar with the typical movement behaviour of cows. This is how we arrive at our ground truth data.

2. Instead of automatically generated simulation data from our simulation service, we also use manually generated data sets for evaluation, analogous to the first point.

The reader must therefore be aware that in the following section when we refer to ground truth data and simulated data, both of these datasets are hand-made. The statistical aspects of the evaluation should therefore be seen as a demonstration of what the evaluation might look like when carried out with real data.

6.2 Statistical Analysis

In the first part of the evaluation, we examine the simulation results with the help of several metrics that are relevant to the significance of the results.

As a basis for our computations, we have manually recorded 4 cows for the ground truth and 4 more cows for the simulated data. Each of these was created for a virtual duration of 48 hours. The created movements were continuous and took place roughly 80% in the barn space and 20% in the pasture space. The resulting datasets each contain 2304 rows of data, with 5 minutes distance between each row.
The sensor configuration is such that sensors 1-65 refer to sensors outside of the barn while sensors 66-105 are located inside the barn.

Spatial Distribution Analysis

As a first metric, we want to get an impression of sensor counts between the two data sets, i.e. how often each sensor gets visited by a cow. By comparing the frequencies, we aim to identify disparities and convergences in movement patterns. The outcome should provide insights into the plausibility of the simulation. If the simulators frequency of occurrences aligns closely with the real-world data, it would support the assumption that the simulation is plausible and accurately represents real-world movement patterns. If there are significant disparities in frequencies, it may either indicate areas where the simulation model needs refinement or it might indicate that the simulator does not represent the movements accurately.

As can be seen in figure 9 and similarly in figure 10, the overall frequency of sensor hits for the sensors 1-65 seems to be very similar. For the barn sensors (66-105) we have a similar tendency, although there seem to be a few stronger discrepancies indicated by the long bars. Those can most likely be explained: The long bars indicate that one or more cows have spent a lot of time in that space, which usually would be expected during resting. In the barn space, there are many resting spaces and they span multiple sensors, so when two cows rest in different sensor spaces, such long bars in the chart may be expected.
The computed correlation between the real and simulated sensor counts is 0.305, indicating that there is a moderate correlation between the data.

Overall the outcome of the Spatial Distribution Analysis here seems to suggest that the simulated data is rather plausible.

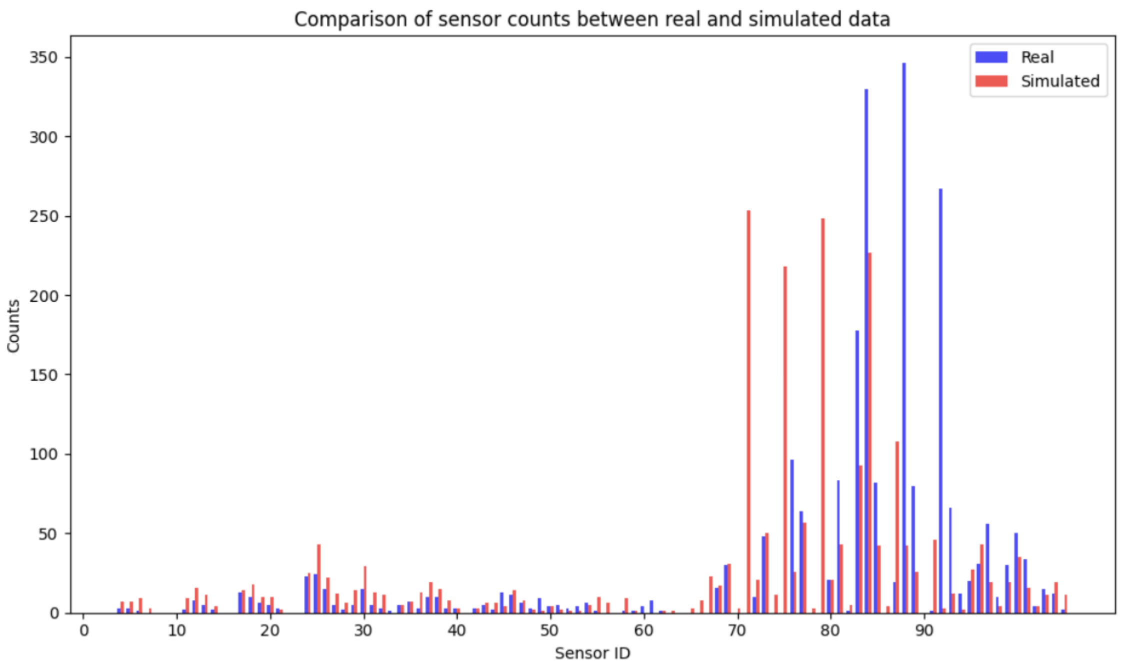


Figure 9: Distribution of Sensor Hits

Temporal Analysis

As a second metric, we inspect the distribution of sensor hits over the day between both datasets. In this case, we first filter the datasets for consecutive sensor hits in the same space, i.e. if a cow stays at a certain pace for 30 minutes, the dataset would contain six consecutive recordings of that cow with the same sensor. In that example, we only count the first recording as a sensor hit and disregard the following five recordings. By doing that, we are left with a filtered dataset that contains only the transitions of a cow between different sensors. Finally, we group the hits by the hour of the day at which they occurred, so we will have at most 24 groupings for each set of data.
In figure 11 we plot the amount of sensor hits throughout the timespan of a day so we can see the amount of movement that is happening at a specific time.
The data indicates a similar amount of movement activity in the first half of the day, with some discrepancies in the afternoon and evening hours. Still, the general movement pattern between the two datasets seems to align rather cohesively, indicating that the simulation produces plausible results. Another detail that becomes apparent from the figure is some events the cows participate in throughout the day. Particularly the milking in the morning hours around six o'clock followed by the opening of the gates to the pasture space. The chart indicates a spike in movement at that time of day, indicating that both datasets participate in those events.

Amount of Distinct Visited Locations

Next, we compare the amounts of distinct locations that have been visited in both datasets, once for all cows in a dataset and also the average of visited locations per cow. As we have

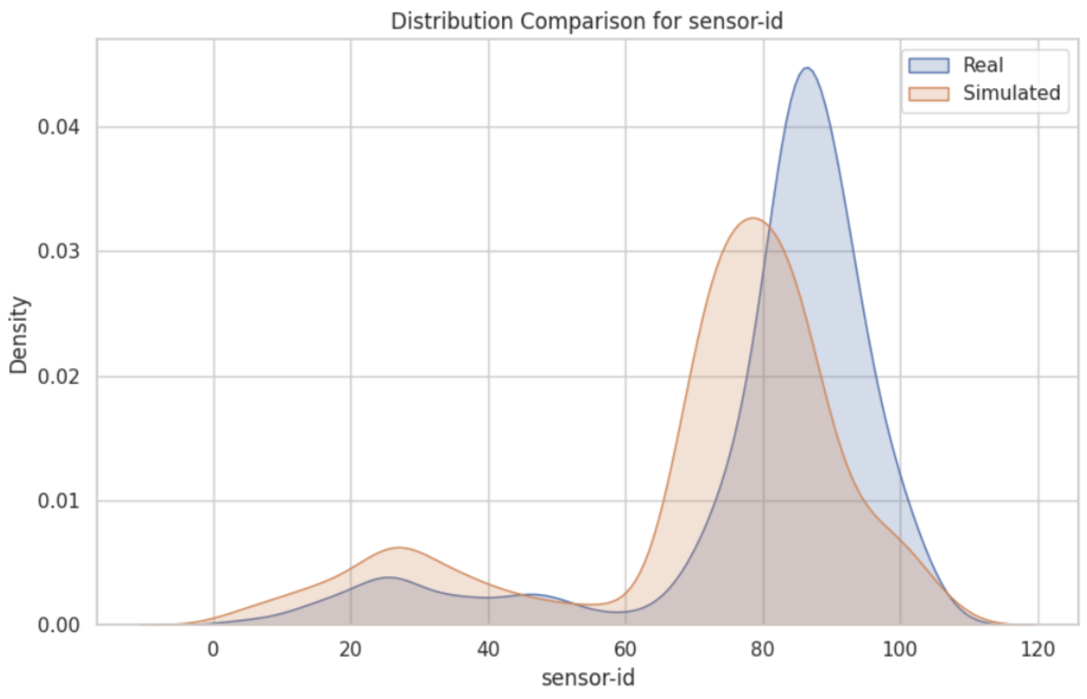


Figure 10: Density of Sensor Hits

105 Sensors that can be visited in total, in figure 12 one can see that not all spaces have been explored in our datasets. The numbers indicate that the datasets are comparable in this regard, although interestingly in the simulated dataset the cows have explored the available space slightly more.

Total Distance Travelled

Finally, we compare the sum of accumulated distances which have been travelled by each cow in both datasets. In order to derive those numbers, we first transform the geographical coordinates of each data entry to meters while considering that the mapping of longitude and latitude to meters depends on the location on earth where the measurements are collected. Then we sum up the distances that each cow travelled whenever it transitions from one sensor to another. The total sum of those distances leads to the results shown in figure 13. One can see that both datasets have a very similar travel distance, indicating that the simulation results would be plausible. In this scenario, the average travelled distance per cow is approximately one kilometer per day, which would be in accordance to the estimation of the domain experts we consulted.

6.3 Limitations and Assumptions

As already pointed out in the disclaimer, the evaluation results are of demonstrative nature as both the ground truth data and the simulation data was not available yet due to technical problems which have not been solved in the given timeframe of this work. This lead to the fact that the capability of *SmartSPEC* to configure and simulate events take

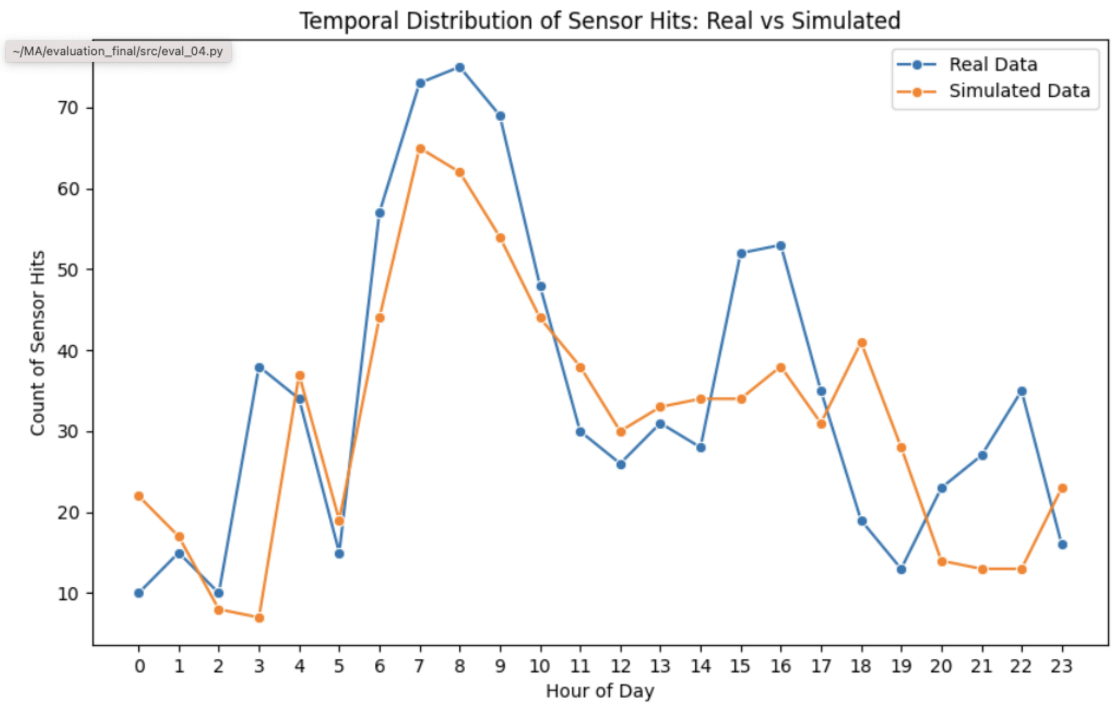


Figure 11: Discriminate Sensor Hits during a Day

place at specific locations at specific times could not be tested and evaluated as planned.

6.4 Requirements Fulfillment Analysis

In this final section we will evaluate how well the requirements which we gathered in an earlier chapter have been fulfilled. The derived conclusions were gathered from stepwise evaluation sessions with our stakeholders. We will elaborate on each of those requirements step by step.

1. **Fulfilled**: *As a user, I want the User Interface to be intuitive and easy to use, so that I don't need a higher level of technical expertise to interact with it.*
   A continuous effort of this work has been dedicated to providing a UI that allows the user to perform the tasks involved to demonstrate the simulator application. By continuously collecting and implementing feedback from sessions with domain experts, the resulting flow of the UI is structured in a way that is easy to grasp. Although there are many settings the user can configure, each setting provides a tooltip that explains how the setting is meant to be configured and how it affects the resulting simulation.

2. **Fulfilled**: *As a user, I want to mimic the layout and conditions of a real farm in the simulation, so that I can better relate to a real-world scenario.*
   The application mimics the layout of the showcase farm in Allmersbach im Tal based on concrete geographical coordinates derived from corner points of that property.

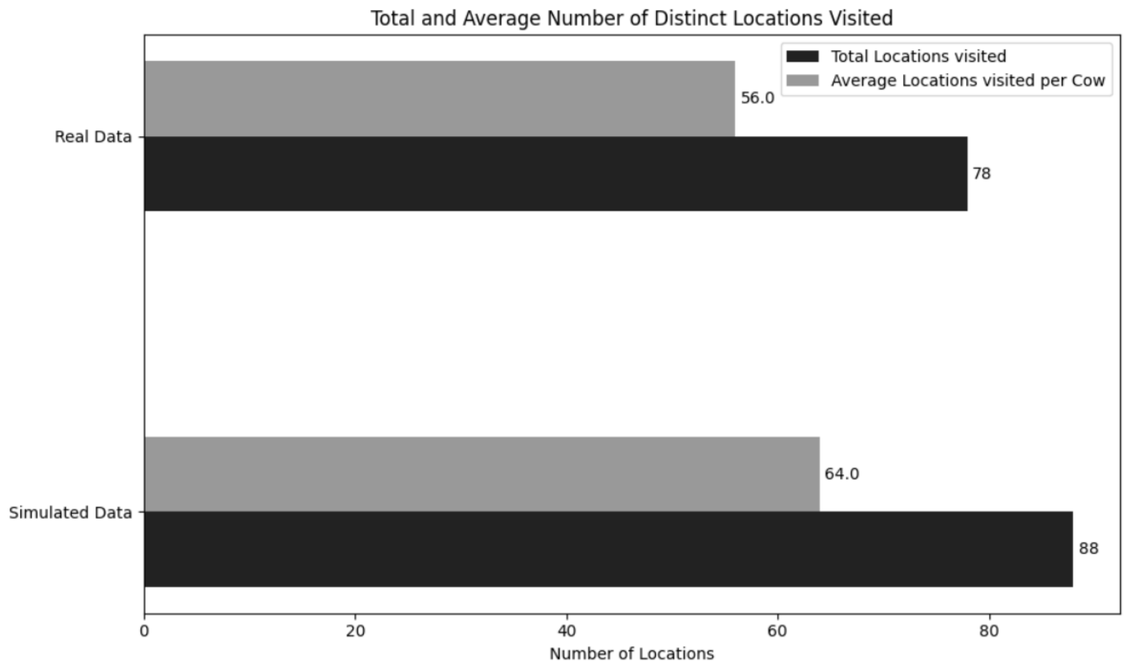


Figure 12: Amount of Distinct Visited Locations

3. **Fulfilled**: *As a user, I want to see a detailed representation of the architectural layout of the barn's inner space so that I can clearly see in which part of the barn the cows are represented.*
   The interactive map of the application visualizes the architectural floor plan at the barn space with higher level detail.

4. **Not Fulfilled**: *As a user, I want to create and configure events that take place in a real scenario, e.g. feeding the cattle at a certain time in a certain place, so that the simulation better matches the actual movement patterns and daily routines, making the simulation results more plausible.*
   While basic configuration possibilities for events are available in the UI, they have no effect on the simulation procedure in the back end. We were not able to test or evaluate the functionality at all, because we encountered the aforementioned technical problems in running the simulator.

5. **Partially Fulfilled**: *As a user, I want to see a realistic simulation of multiple cow's movements, so that I can get an impression of how their movement could look like with real movement data.*
   While the UI is ready to display multiple cows moving through the farm space, the current absence of simulated data prevents to showcase a realistic simulation.

6. **Fulfilled**: *As a user, I want to simulate both indoor and outdoor farm environments, so that I can get a complete picture of the involved procedures.*
   The application demonstrates a hybrid combination of an indoor and an outdoor environment that is connected so that the cow can move seamlessly between both types of environments.

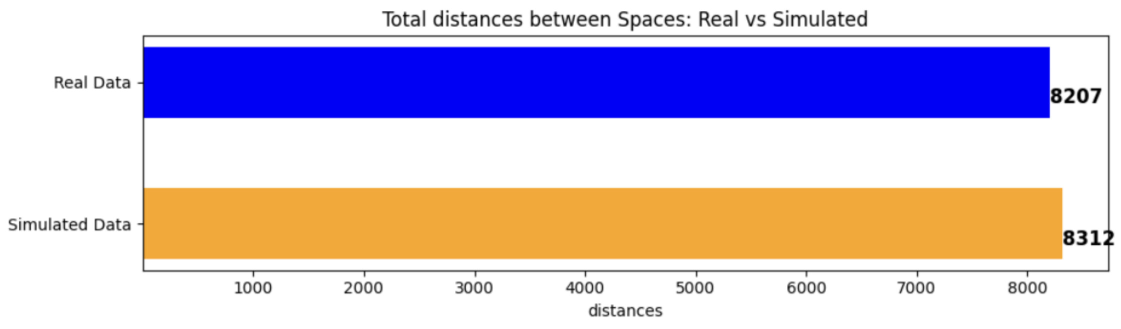


Figure 13: Total Distance Travelled

7. **Partially Fulfilled**: *As a user, I want to simulate varying number of sensors, cows, barn sizes and pasture sizes, so that I can try out the impact of different farm configurations.*
   While the user can configure the simulation to use custom amounts of cows and sensors, the sizes of barn and pasture and the positions of sensors in the barn are configured in the applications source code but not exposed to the user. Further work on the UI is needed to allow the user to customise those spaces to his needs.

8. **Partially Fulfilled**: *As a user, I want to save and retrieve previous simulation configurations and results, so that I can try out different configurations and their impact on the results easily.*
   Currently only one configuration at a time can be stored on the server for the simulation engine. While any configuration can be easily exported and stored locally as file in a JSON-Format, loading those JSON-Files into the application remains to be implemented.

9. **Not Fulfilled**: *As a user, I want to be shown quantitative differences between simulations, so that I can make effective comparisons.*
   There is yet no functionality to compare different simulation results and this is being left as an option for future work.

10. **Fulfilled**: *As a user, I want to draw ground truth data into the virtual map manually to imitate the movement of cows by a hand-drawn movement path, so that I don't need to rely on the presence of real ground truth data to test the simulator.*
    This requirement emerged out of necessity due to a lack of data and has therefore been implemented.

11. **Fulfilled**: *As a user, when drawing ground truth data manually, I want the User interface to prevent me from drawing movements outside of the physical boundaries of the farm space, so that I don't generate incorrect inputs.*
    The user can only draw movements inside the boundaries of the configured farm space.

12. **Partially Fulfilled**: *As a user, when drawing ground truth data, I want to adjust the recording interval, so that I can mimic the real sensor systems realistically.*
    The recording interval of the sensors can be configured easily in the UI, but at

the current state it only allows one setting which is applied to all sensors. In a real scenario when there are multiple sensor systems, they may have independent recording intervals which differ in their duration. Supporting that feature is possible, but currently not implemented.

13. **Fulfilled**: *As a user, I want to adjust the visualization of the spaces and sensors, in particular colors and transparency, so that I can see them clearly but at the same time they won't be distracting.*
The UI provides multiple settings to configure the visual appearance of the farm space and its configuration properties.

14. **Fulfilled**: *As a user, I want to have multiple options to visualize the movement of cows to suit different needs. In particular the cows should be representable as simple symbols, as numbers representing the amount of cows in a particular subspace, as numbers representing the ID of each cow and as a circle whose radius represents the amount of cows in a particular field.*
The UI provides an option to select how the simulated cows should be visualized.

15. **Not Fulfilled**: *As a user, I want the cow's movement in the barn to be restricted in such a way that it matches the real physical restrictions of that barn, so that the resulting movements are highly realistic.*
That feature is currently not implemented but might be added in a future iteration by providing an option to manually overwrite the neighbour-relationships of sensors within the UI.

16. **Partially Fulfilled**: *As a user, I want to zoom and pan the virtual map, so that I can better focus on specific subsections of the simulated space.*
The user can pan the map with hotkeys, and can toggle to zoom in and out of the barn, but a traditional zoom feature where the user can zoom in and out as much as he wants is not yet implemented in a user-friendly manner.

17. **Fulfilled**: *As a user, I want to adjust the speed of the running simulation, so that I can either focus on smaller details when the simulation pace is slow, or that I can have a broader impression by running the simulation in timelapse.*
The user can control the simulation speed of the visualization through the use of hotkeys.

18. **Partially Fulfilled**: *As a user, I want to start, and stop the simulation, so that I can visualize the results.*
The functionality has already been implemented, but can currently not be used due to the mentioned technical problems.

19. **Not Fulfilled**: *As a user, I want to fetch real-time data from WeideInsights Sim-track component and feed it as input for the simulation, so that I don't need to depend on the manual collection and preparation of data in order to use the simulation.*
Connecting to the track component has not been implemented yet.

20. **Fulfilled**: *The simulator should be easy to install and execute.*
The dockerized approach to setting up the application and the use of makefiles allows for an easy setup process.

21. **Partially Fulfilled**: *The software should be simple to maintain and update.*
While the main architecture of the backend supports this requirement and while the frontend is built with VueJS as a framework that facilitates maintenance and updates, the functionality of the interactive map might take a bit more effort on the developer's side to understand, as there are lots of computations going on under the surface with multiple data sources, transformations of coordinate spaces, etc. Nevertheless, the code documentation provides guidance on how to work with it.

22. **Fulfilled**: *The software should be designed in a modular way to allow for easy expansion and addition of new features.*
Both back end and front end were designed and implemented with flexibility in mind.

23. **Fulfilled**: *The software should be compatible with common operating systems and devices used by farmers.*
The client side of the application can be used on any platform that offers a browser and an internet connection.

24. **Fulfilled**: *The UI should support the user in his understanding of how to use the application.*
The UI provides an organized structure as well as infoboxes, tooltips, and notifications to support the user during the interaction.

25. **Partially Fulfilled**: *The UI should prevent the user from making incorrect inputs.*
While most configuration settings in the UI are restricted to prevent wrong inputs on an individual level, the summary of multiple settings might still lead to a configuration at which the SmartSPEC simulator would produce inappropriate results.

26. **Fulfilled**: *The UI should respond quickly to user inputs to support a seamless user experience.*
The computations that drive the interactive UI are optimized sufficiently to support a fluid user experience.

27. **Fulfilled**: *The software should be able to generate and visualize simulations quickly and accurately without long waiting times during the process.*
When the system is working as expected, it can produce results that span days of virtual time in a matter of minutes.

28. **Not Fulfilled**: *The system should be able to forward simulated data to the Simtrack component.*
Communication with the Simtrack component has not been tackled yet and is left for future iterations.

29. **Fulfilled**: *The software should provide a clear documentation.*
The software is documented sufficiently to help new developers get on board quickly.

Chapter 7

Conclusion

In the last section, we will summarise our research and the results. Finally, we will explain what possibilities could be explored further in the future or how our work can be followed up.

7.1 Summary

This thesis attempted to answer the research question, "How can a plausible simulation of hybrid location systems support the demonstration of smart agriculture applications?". The primary aim was to explore and develop a simulator application, leveraging a microservice architecture in the back end and an SPA application in the front end, to effectively represent hybrid location models in a smart agriculture demonstrator. One focus point was to extend an existing simulator such that it can handle a hybrid mix of sensor technologies, while at the same time demonstrating that capability in the front end. Another aspect of this work was to provide an intuitive, feature-rich user interface to the users, while at the same time making sure that the data generated by the simulator is useful and plausible. As we encountered technical problems during our evaluation phase, we could not validate the simulator's plausibility, although previous works already seem to indicate that it should be sufficient. For the same reasons we missed the chance to evaluate the simulator's capability to incorporate events into the trained model, which should provide even more value to the clients once they are implemented correctly. Due to the lack of available ground truth data, we opted to extend the UI such that users could draw their ground truth data by hand. When guided by domain experts like in our case, manually generated data can be sufficiently plausible to serve as input for the training phase of the simulation engine.

To get a clearer answer to our thesis question, we conducted multiple demonstration sessions with our domain experts throughout the project, so we were able to gather valuable feedback. The overall tone was an appreciation of the possibilities such an application can provide. It seemed important to demonstrate the functionality actively with an interactive user interface, such that they could see the system in action and thus were able to intuitively grasp the possibilities and advantages such a simulation can bring to the table on their own. As long as the simulated movement data was sufficiently plausible the simulator was perceived as a beneficial component for supplementing the *WeideInsight* Project.

7.2 Future Work

Our work led to another iteration of software development on an existing simulator prototype. Our requirement analysis guided us throughout the development, and while many of the requirements were successfully implemented, some of them were not or were only partially implemented. About the latter, we suggest several possible improvements to the simulation that may be of interest for future work:

- Feeding the simulation with real ground truth data from hybrid sensor system setups.
- Exploring the simulator's capabilities to generate movement data based on events.
- Extending the front end to make the simulation even more configurable, e.g. enabling users to manually configure the barn and pasture layout as well as traversable spaces.
- Allowing the user to generate, compare, and analyze the results of different setups in a user-friendly manner.
- Connecting the front end visualization to *WeideInsights* Simtrack component.
- Introducing features to replicate technical peculiarities of the sensor systems, e.g. battery life, signal strength, failures, etc.

As can be seen from these suggestions, there are many more possibilities for further research and development to explore.

Bibliography

(2023). Smartspec. Accessed: 2023-22-6, `https://github.com/andrewgchio/SmartSPEC`.

Apple (2014). Getting started with ibeacon. `https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf`.

Becker, C. and Dürr, F. (2005). On location models for ubiquitous computing. *Personal and Ubiquitous Computing*, 9:20–31.

Chio, A., Jiang, D., Gupta, P., Bouloukakis, G., Yus, R., Mehrotra, S., and Venkatasubramanian, N. (2022). Smartspec: Customizable smart space datasets via event-driven simulations. In *2022 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 152–162.

Encyclopedia Britannica (2022a). Bluetooth. Encyclopaedia Britannica, Inc. Accessed: 2023-6-6.

Encyclopedia Britannica (2022b). Gps. Encyclopaedia Britannica, Inc. Accessed: 2023-6-6.

Ghadami, A. (2022). Designing a data generator for hybrid location models in precision agriculture. Master's thesis, University of Bamberg, Germany.

Ke, C., Wu, M., Chan, Y., and Lu, K. (2018). Developing a ble beacon-based location system using location fingerprint positioning for smart home power management. *Energies*, 11(12). Copyright - © 2018. This work is licensed under https://creativecommons.org/licenses/by/4.0/ (the "License"). Notwithstanding the ProQuest Terms and Conditions, you may use this content in accordance with the terms of the License; Last updated - 2022-06-13; SubjectsTermNotLitGenreText - Taiwan.

Luber, S. (2021). Was ist mioty? Accessed: 2023-12-6, `https://www.ip-insider.de/was-ist-mioty-a-dcfebe13062221bf17a430e4cb78b8f3/`.

Luber, S. (2022). Was ist lpwan? Accessed: 2023-12-6, `https://www.ip-insider.de/was-ist-lpwan-a-1092749/`.

Ören, T., Zeigler, B. P., and Tolk, A. (2023). *Body of Knowledge for Modeling and Simulation*. Springer Nature Switzerland AG, Cham, Switzerland, first edition.

Pascale, F., Adinolfi, E. A., Avagliano, M., Giannella, V., and Salas, A. (2021). A low energy iot application using beacon for indoor localization. *Applied Sciences*, 11(11):4902. Copyright - © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/). Notwithstanding the ProQuest Terms and Conditions, you may use this content in accordance with the terms of the License; Last updated - 2022-09-22.

Pongratz, P. (2023). A simulator for location-based agriculture scenarios. Master's thesis, University of Bamberg, Germany.

WeideInsight (2021). Weideinsight: Mehrwert im herdenmanagement durch kostengünstige, hybride lokalisierung und intelligente datenintegration.

Attachment

Sources

The repository which includes the thesis report, the source code, the documentation and the configuration which has been used for the evaluation can be found at:
https://gitlab.rz.uni-bamberg.de/mobi/theses/2023-ma-david-jares.git

Erklärung

Ich erkläre hiermit gemäß § 9 Abs. 12 APO, dass ich die vorstehende Masterarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Des Weiteren erkläre ich, dass Inhalt und Wortlaut der beiden Fassungen (digital/in Papierform) identisch sind und zur Kenntnis genommen wurde, dass die digitale Fassung einer durch Software unterstützten, anonymisierten Prüfung auf Plagiate unterzogen werden kann.

Datum	Unterschrift