

Activity Recognition and Localization in Smart Environments



**Università
di Genova**

Marco Limone

DIBRIS - Department of Computer Science, Bioengineering,
Robotics and System Engineering

University of Genova

Supervisors:

Prof. Antonio Sgorbissa

Prof. Carmine Recchiuto

Prof. Luca Oneto

In partial fulfillment of the requirements for the degree of

Laurea Magistrale in Robotics Engineering

October 26, 2023

Declaration of Originality

I, Marco Limone, hereby declare that this thesis is my own work and all sources of information and ideas have been acknowledged appropriately. This work has not been submitted for any other degree or academic qualification. I understand that any act of plagiarism, reproduction, or use of the whole or any part of this thesis without proper acknowledgment may result in severe academic penalties.

Acknowledgements

Special thanks to Professor Antonio Sgorbissa, who, by making the time spent working together enjoyable, guided and supported me in the development of my thesis work, even giving me the opportunity to write a paper. Likewise, a heartfelt thanks also to professors Carmine Recchiuto and Luca Oneto, for their help throughout this work.

Abstract

In recent decades, the evolution of digital technologies has led to the creation of smart environments, spaces in which objects and devices are interconnected and able to communicate with each other, working together to enhance the user experience. These environments, also known as "Internet of Things" (IoT) or "Internet of Everything" (IoE), have revolutionized several areas of our daily lives, including home, office, transportation and health.

This thesis work focuses on the concept of smart environments and its importance in identifying and recognizing what is happening within a specific environment. In particular, it explores how the systems in these environments are able to collect, analyze and interpret data from sensors and devices, enabling a better understanding of the dynamics and the activities taking place.

One of the key aspects of intelligent environments is the ability to recognize events and interactions occurring within the environment. This recognition is based on the use of advanced artificial intelligence techniques, such as machine learning and computer vision. Using these technologies, intelligent environments can discern human activities, monitor physical spaces, and identify users' behavioral patterns. This results in a more personalized and efficient experience for the individuals involved, improving quality of life and optimizing available resources.

In addition, analysis of the environment is important for the creation of safe and suited to users' needs environments. Through data collection and interpretation of information, hazardous situations or abnormal behavior can be identified. This is particularly relevant in areas such as home security, industrial automation and health care, where early detection of potential threats can prevent significant damage.

However, the implementation of smart environments also raises ethical and privacy issues. The collection of vast amounts of data carries the risk of privacy violations and misuse of personal information. Therefore, it is essential to strike a balance between the efficiency

and convenience of intelligent environments and the protection of the privacy of the individuals involved.

This thesis explores technological developments and challenges related to smart environments, highlighting the importance of recognizing what is happening within these spaces. Through extensive research and critical analysis, methods and strategies are proposed to address emerging challenges and maximize the benefits of smart environments.

Contents

1	Introduction	1
1.1	Context	1
1.2	Motivations	2
1.3	Objectives	3
1.4	Document’s Structure	3
2	State of the Art	5
2.1	Architecture for Ambient Intelligence	5
2.1.1	Architecture Cloud for Ambient Intelligence	7
2.2	Activity Recognition	11
2.2.1	Indoor Localization	11
2.2.1.1	Topological Localization	11
2.2.1.2	Geometrical Localization	13
2.2.1.3	Approaches	14
2.3	Machine Learning for Activity Recognition	19
2.4	Large Language Models	27
3	Software Architecture	29
3.1	Sensors	36
3.2	Data Acquisition	37
3.2.1	Cloud Architecture	39
3.2.2	Database Structure	41
3.3	Reasoning Server	43
3.3.1	Activity Detection with ML	44
3.3.2	Activity Detection with LLM	47
4	Experiments	51
4.1	Methodology	51
4.1.1	Strategic Arrangement of Sensors	51
4.1.2	Activities and Creation of Datasets	52
4.1.3	Database Access	54

CONTENTS

4.1.4	ML	55
4.1.5	LLM	57
4.1.6	Online Data Acquisition and Analysis	59
4.2	Results	59
4.2.1	ML Accuracy	59
4.2.2	LLM Accuracy	62
4.2.3	Real-time test	63
4.2.4	Communication times	66
4.3	Discussion	67
5	Conclusions	69
	References	76

List of Figures

2.1	Ambient Intelligence architecture from <i>Agate et al. (2019)</i>	6
2.2	AWS Cloud Architecture	9
2.3	GCP cloud architecture.	10
2.4	Example of a Topological map from <i>Zhu et al. (2019)</i>	12
2.5	Example of Trilateration from <i>ilçi et al. (2015)</i>	13
2.6	Example of Triangulation from <i>Montanha et al. (2019)</i>	14
2.7	Overlapped areas for human detection with PIR from <i>Wu et al. (2021)</i>	16
2.8	Schema of tests done with RFID from <i>Xu et al. (2018a)</i>	17
2.9	Schema of tests done with UWB from <i>Che et al. (2023)</i>	17
2.10	System architecture with BLE from <i>García-Paterna et al. (2021)</i>	18
2.11	Example of Random Forest	21
2.12	Pose machines flow chart	22
2.13	LSTM flow chart example	23
2.14	SVM functioning	23
2.15	HMM scheme	24
2.16	Transformers scheme from <i>Gavrilyuk et al. (2020)</i>	25
2.17	Prompt schema of the LLM-Planner	28
3.1	First evolution of the architecture.	30
3.2	Second evolution of the architecture.	31
3.3	Cloud architecture for data acquisition and activity recognition.	32
3.4	Sequence diagram showing data exchanges between architecture's components.	35
3.5	Database structure.	42
3.6	Structure of the prompt: the system field is constant for all activities, while the user field varies based on the specific activities and sensor readings.	49

LIST OF FIGURES

4.1	Smart Environment considered for experiments. In the image, the red dots represent the PIR sensors with their visibility range. Blue dots represent the magnetic door sensors. The robot is manually placed in different positions. In black we see the desks and the meeting table.	52
4.2	Environment Topological map	53

Chapter 1

Introduction

1.1 Context

When we talk about smart environments we refer to spaces in which the synergy between digital and physical technologies enables the creation of personalized and enhanced experiences for users. Smart environments have revolutionized several spheres of our daily lives, such as home, office, transportation, and healthcare, offering extraordinary opportunities for optimization and comfort.

This thesis focuses on the implementation of a human AR (Activity Recognition) and localization system within smart environments, using passive infrared (PIR), magnetic door sensors and cameras, whose data are analyzed by machine learning algorithms. The resulting architecture is a combination of a hardware part given by the sensors and a software part based on a Client/Server architecture, organized on several layers, to allow the proper exchange of data and make them available for evaluation.

Activity Recognition plays a central role in enhancing the user experience within an intelligent environment. The capability to automatically and accurately recognize human activities enables the system to adapt in real-time to the user's needs. The use of passive infrared (PIR) sensors, which detect body heat emitted by people in motion, offers an effective and cost-efficient solution for capturing activities related to specific locations in the environment without intrusive interactions with the user. Magnetic door sensors, providing information about the state (open or closed) of doors and windows, can complement PIR sensors. Robots equipped with cameras can furnish additional data about user posture, gestures, and gaze without compromising privacy, as users can explicitly request the robot to withdraw or "close its eyes".

By analyzing the data collected from these sensors, it is possible to identify and categorize behavioral patterns and gather information about the user's activities and their locations. Then, information about detected activities can offer valuable insights for optimizing the environment based on the inhabitants' habits, provide information to improve their safety (especially for older people living alone), or even assist the robot in assessing different ways to interact with them.

The contribution of this work lies in its approach to data analysis, conducted either through machine learning (ML) algorithms or large language models (LLMs), a different approach for activity recognition and localization. Additionally, we introduce a solution that consists of a hardware layer comprising sensors and processing units, as well as a software layer built as a cloud client/server architecture. This architecture facilitates the exchange of data, making it available for reasoning and evaluation. Furthermore, this architecture is designed to manage simultaneous internal environments, making it well-suited for handling different real-world scenarios.

1.2 Motivations

In order to effectively address the challenges associated with this approach, careful management of the data collected is essential. This entails integrating data from different sensors, such as PIRs and magnetics, and using machine learning algorithms to minimize errors and achieve precise outcomes. It is important to gain insights into individuals' activities to understand their unique needs across different domains. For instance, in the healthcare industry, this can help us gain a better understanding of patients' or elderly individuals' requirements, while in the workplace, it can assist in assessing the progress of work processes and evaluating the overall quality of the work environment.

When discussing the topic of data collection within smart environments, it is important to emphasize the importance of safeguarding user privacy. This must be the primary focus when implementing such environments. As the data collected can comprise sensitive information regarding users' lifestyles and whereabouts, it is imperative to take appropriate measures to guarantee that privacy is guaranteed under all circumstances.

1.3 Objectives

This thesis aims to create a versatile architecture that can handle easily multiple internal environments. Its goal is to accurately recognize activities and locations in a modular and robust manner. With a deep understanding of the challenges involved.

The objective of this study is to develop a reliable Client/Server framework that can efficiently manage data acquisition, storage, and activity recognition. Our goal is to create a system that works consistently in different settings and complex scenarios, achieved through careful planning and integration of essential components.

In order to accomplish our goals, we plan to use two different predictive models. Our first strategy involves utilizing conventional Machine Learning algorithms, like the Random Forest and versatile Support Vector Machine (SVM). These well-established techniques have demonstrated their effectiveness in various scenarios and can provide us with valuable insights into our specific area of interest.

We recognize the vast possibilities that advanced technologies offer and are currently exploring the potential of Large Language Models (LLMs). These modern models have transformed the way we process and comprehend natural language. Our aim is to harness the power of LLMs for activity recognition and localization, thereby unlocking valuable new perspectives.

Our research will center on comparing these two methods, providing a detailed evaluation of their respective advantages, drawbacks, and overall efficacy. Through this comparative analysis, we aim to extract important assessments that will guide us in future data-driven efforts.

This thesis aims to create an architecture that can manage multiple internal environments while providing modularity, robustness, and precision in activity recognition and localization.

1.4 Document's Structure

Following this, all the stages of the work done divided into various chapters will be presented in detail, specifically:

1.4 Document's Structure

1. Chapter 2: this chapter presents the state of the art by dwelling on existing architectures, activity recognition, localization, Machine Learning algorithms, and LLM
2. Chapter 3: this chapter presents how is structured the software architecture
3. Chapter 4: this chapter presents the experiments done and the results obtained
4. Chapter 5: this chapter presents the final conclusions

Chapter 2

State of the Art

2.1 Architecture for Ambient Intelligence

In the context of Ambient Intelligence (AmI), combining complex and innovative solutions is crucial. AmI-based systems must be dynamic, flexible, robust, adaptable to changing contexts, scalable, and easy to use and maintain. The use of Multi-agent systems(MAS), like in [Tapia *et al.* \(2009b\)](#) [Fraile *et al.* \(2008\)](#) [Tapia *et al.* \(2013\)](#) [Sanchez-Pi & Molina \(2010\)](#), plays a key role in this area, as they have important qualities such as autonomy, reasoning ability, responsiveness, social skills, and proactivity, which prove to be crucial for the development of AmI-based distributed systems. Another essential aspect in these developments is the use of context-sensitive technologies, which enable them to sense surrounding stimuli and react autonomously and effectively.

As part of the search for software solutions that can adapt to people's needs and specific situations, the development of Ambient Intelligence (AmI) systems has emerged. This new technological frontier aims to personalize human-machine interaction by incorporating pervasive computing elements that communicate with each other ubiquitously. AmI systems aim to capture and manage relevant information in the surrounding environment, enabling greater adaptability and interactivity between technology and individuals. Operating in a lot of different fields like in [Alonso *et al.* \(2010\)](#) [Cai *et al.* \(2019\)](#) [Bavafa & Navidi \(2010\)](#).

Today, society is increasingly accustomed to the use of compact, noninvasive technological devices that facilitate daily routines. These devices act as agents, constantly collecting dynamic data from the surrounding context in a distributed manner. However, the integration of such devices is not without challenges, requiring the development of innovative solutions that combine different approaches to create versatile and adaptable systems.

2.1 Architecture for Ambient Intelligence

With an intelligent and ubiquitous approach, AmIs aim to provide personalized support that is integrated into the surrounding environment, thereby enhancing people's life experiences.

The sensing component of AmI could be wired or wireless, like in [Kartakis *et al.* \(2012\)](#), with sensors either independent or embedded in a device such as a wearable or smartphone. Wireless sensor networks (WSN) are more flexible and require less infrastructural support than wired sensor networks, as in [Marin-Perianu *et al.* \(2007\)](#) [Tapia *et al.* \(2009a\)](#) [Malatras *et al.* \(2008\)](#). However, wired sensors can provide more reliable and secure data transmission.

Among the many architectures present in the present state, a very interesting one is the one presented in [Agate *et al.* \(2019\)](#) where the idea from which the architecture was born is very close to the one we would like to develop in this thesis, that is, an architecture capable of understanding what is happening in the concerned environment and to sample activities and habits of those who live within it. An example of the architecture they devised can be found in Figures 2.1

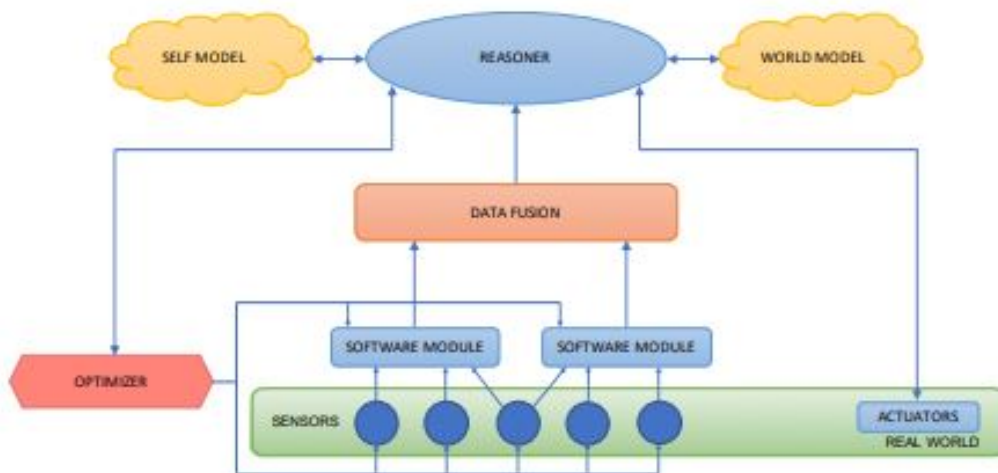


Figure 2.1: Ambient Intelligence architecture from [Agate *et al.* \(2019\)](#).

2.1.1 Architecture Cloud for Ambient Intelligence

A cloud is a remote server-based data storage and management system that offers a wide range of services and resources over the Internet. Data are stored and managed on virtual servers, allowing users to access them from anywhere at any time. Cloud connection is the process of connecting an organization's IT system to the cloud platform for storing, processing, and managing data and applications. Cloud connections can be implemented through several mechanisms, including virtual private network (VPN) connection, direct network connection (DNC), and hybrid network connection. In today's digital world, more and more organizations are adopting cloud technology to improve the productivity, efficiency and flexibility of their systems. Cloud connectivity is a critical aspect of this change, as it enables organizations to access and manage data and applications from anywhere, anytime, with an Internet connection. There are several benefits that characterize cloud connectivity and that need to be taken into account. Cloud connectivity offers several benefits to organizations. First, it offers greater flexibility and scalability, as organizations can access their data and applications from anywhere, anytime. Second, it reduces IT infrastructure costs, as organizations can use cloud resources according to their needs and pay only for what they use. Third, cloud connectivity offers greater data security, as data is stored safely and securely in the cloud platform.

Regarding the use of clouds to manage data from smart sensors in smart environments we could have some examples like [Cubo *et al.* \(2014\)](#), but two are the most prominent reference examples:

1. Amazon Web Services (AWS).
2. Google Cloud Platform (GCP).

In recent years, the evolution of digital technologies has led to the creation of increasingly intelligent and connected environments, known as smart environments or smart environments. These environments integrate a wide range of devices, sensors, data and digital resources to improve efficiency, convenience and the overall user experience. Amazon Web Services (AWS) has established itself as a leading cloud service provider, offering a range of tools and resources that are essential for creating and managing smart and intelligent environments.

AWS offers services in Intelligent and Smart Environments such as:

1. IoT Devices and Sensors: Smart environments rely on a vast network of IoT devices and sensors to collect real-time data. AWS IoT provides the

2.1 Architecture for Ambient Intelligence

ability to securely connect, manage and interact with these devices. AWS IoT Core enables secure, two-way communication between devices and the cloud, while AWS Greengrass enables local processing on devices to reduce latency and improve responsiveness.

2. **Data Storage and Analysis:** Data collection from devices and sensors requires powerful storage and analysis solutions. AWS S3 (Simple Storage Service) offers scalable and durable storage space, while Amazon DynamoDB is a NoSQL database for storing structured and semi-structured data. For data analysis, AWS offers services such as Amazon Kinesis for real-time data stream management and Amazon Redshift for large-scale data analysis.
3. **Artificial Intelligence and Machine Learning:** Intelligent environments often leverage artificial intelligence (AI) and machine learning (ML) to extract insights from data and make automated decisions. AWS provides Amazon SageMaker for developing, training and implementing ML models. In addition, services such as Amazon Rekognition enable image and face recognition, while Amazon Polly offers speech synthesis.
4. **Security and Privacy:** In connected environments, data security is critical. AWS offers tools such as Amazon Identity and Access Management (IAM) for authorization management, Amazon GuardDuty for threat detection, and AWS Key Management Service (KMS) for cryptographic key management.
5. **Management and Automation:** Centralized management is essential to maintain efficiency in smart environments. AWS offers AWS Management Console for simplified management, while AWS CloudFormation enables automated provisioning and resource management.

The use cases of AWS in Smart and Intelligent Environments is indeed considerable for example its use in smart cities can be used to create solutions for traffic monitoring, smart lighting, waste management and more, helping to make cities more efficient and sustainable; smart buildings for controlling heating, ventilation, and air conditioning systems, optimizing energy use, and managing security; or even in health and wellness where smart environments in healthcare can use AWS to collect and analyze patient data, support telehealth, and facilitate the management of connected medical devices.

From Figure 2.2 we can see the basic architecture used in AWS which precisely presents the Lambda tool in which there are all the applications that can be created by the developer to manage data or obtain specific information.

2.1 Architecture for Ambient Intelligence

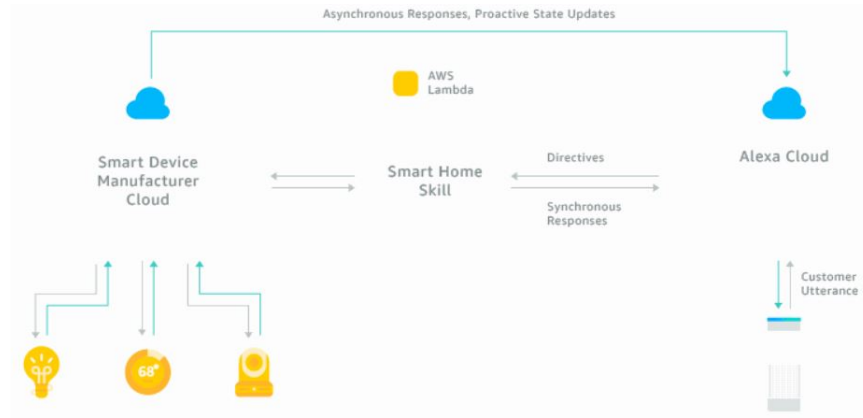


Figure 2.2: AWS Cloud Architecture

Google Cloud Platform (GCP) has established itself as a leading cloud service provider, offering a range of tools and resources that play a crucial role in creating and managing smart and intelligent environments.

1. Internet of Things (IoT) and Sensors: GCP provides Google Cloud IoT Core, a service for connecting, managing and monitoring IoT devices and sensors. It allows data to be collected from connected devices and securely transmitted to the cloud for analysis.
2. Data Storage and Analysis: GCP offers scalable storage solutions such as Google Cloud Storage and databases such as Google Cloud Bigtable and Google Cloud Firestore. For data analysis, Google BigQuery is a powerful tool for querying large amounts of data quickly and efficiently.
3. Artificial Intelligence and Machine Learning: The integration of artificial intelligence and machine learning is critical in smart environments. Google Cloud offers tools such as Google AI Platform for developing, training and deploying machine learning models. In addition, Google AutoML makes it easy to create models without the need to be a machine learning expert.
4. Geospatial Data Analysis: For environments requiring geospatial analysis, GCP offers Google Cloud Location Services, enabling the integration of mapping, positioning, and geocoding capabilities into services.
5. Security and Privacy: GCP provides advanced security tools such as Google Cloud Identity and Access Management (IAM), which allows granular management of permissions. Google Cloud Security Command Center provides an overview of the security of the environment.

2.1 Architecture for Ambient Intelligence

6. Management and Automation: Management and automation are simplified by services such as Google Cloud Console for centralized management and Google Cloud Deployment Manager for declarative resource creation and management.

In Figure 2.3 it is possible to see the basic architecture used by Google for its smart environments. It can be seen that the Google Cloud Platform contains all the services made available by Google including an environment for creating code and applications, and a storage section for data.

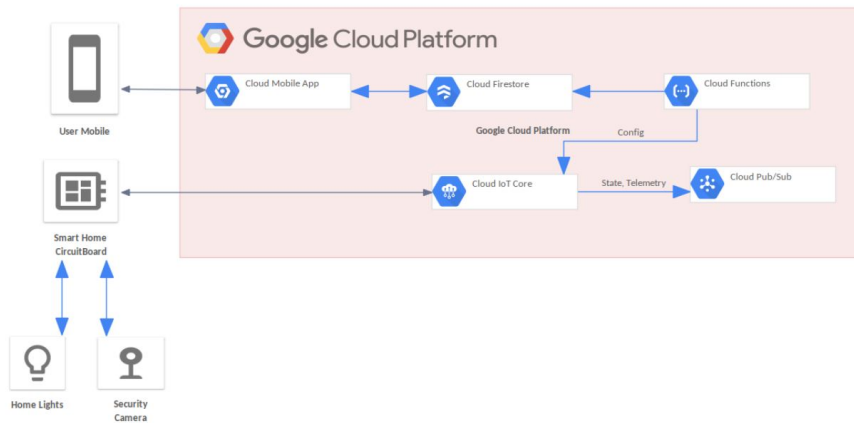


Figure 2.3: GCP cloud architecture.

As with AWS, the GCP use cases can be the same going from smart city to smart home or wellness and health.

In conclusion, Amazon Web Services and Google Cloud Platform offer a wide range of services and resources that are well suited to the needs of smart and intelligent environments. From data storage and analytics to artificial intelligence and security, they both play a key role in enabling the digital transformation of environments across different industries. The integration of these services can lead to a higher level of efficiency, sustainability, and convenience for users of these advanced environments.

2.2 Activity Recognition

2.2.1 Indoor Localization

Indoor localization, commonly known as indoor positioning, is a cutting-edge technology that enables the location of objects, devices, or individuals within buildings, enclosed spaces, and business facilities. Although outdoor location systems such as GPS have become very important in our everyday lives, they fall short in providing precise information indoors. To bridge this gap, indoor localization has emerged, utilizing a range of technologies and methods to accurately calculate the coordinates of individuals or devices in environments where satellite signals are not available.

Indoor location is rapidly gaining importance in several application areas. In shopping malls and airports, it offers personalized navigation services, providing precise directions to users regarding desired stores or boarding gates. In the corporate environment, indoor locations can be used for asset tracking, optimization of warehouse operations, and workflow management. In addition, in the entertainment and tourism sectors, visitor experiences in museums, theme parks or fairs can be improved through interactive apps that provide contextual information based on the user's location.

However, indoor localization still presents some challenges, such as the need to manage signal interference and ensure data privacy of localized devices. Nevertheless, thanks to technological advances and continuous innovations, indoor localization is gradually becoming more accurate, efficient, and reliable, paving the way for a future in which indoor navigation will be as smooth and intuitive as outdoor navigation.

2.2.1.1 Topological Localization

Topological localization is an approach to indoor localization that is based on the creation of a topological map of the environment in which localization takes place. The topological map describes the spatial relationships between different areas or points of interest within the environment, such as corridors, rooms, stairways, or access points.

In the context of topological localization, the main objective is to determine the relative position of an object or device within this map, rather than to determine its precise location in terms of spatial coordinates. For example, instead of providing the exact coordinates of a device, topological localization may indicate

that the device is in a specific room or along a specific path.

Topological localization relies on algorithms and techniques that exploit relative information, such as the path followed or points of interest traversed, to determine relative location. This information is usually collected using specific sensors or technologies, such as cameras, motion sensors, or Wi-Fi access points.

One of the advantageous aspects of topological localization is its relative simplicity compared to other approaches, such as geometric localization. It does not necessarily require complex infrastructure or measurement tools. Furthermore, topological localization can be useful in situations where it is sufficient to know the general area in which an object or device is located, rather than its exact location.

Topological localization is used in many different ways; in [Zhu *et al.* \(2019\)](#) for example, a method is used that has the same concept behind it that will be developed later for the method proposed in this thesis. Specifically, a Convolutional Neural Network trained on a dataset of images, representing all rooms in the chosen environment, is used to perform topological localization using images from a smartphone camera. The following is an example of a topological map used in [Zhu *et al.* \(2019\)](#).

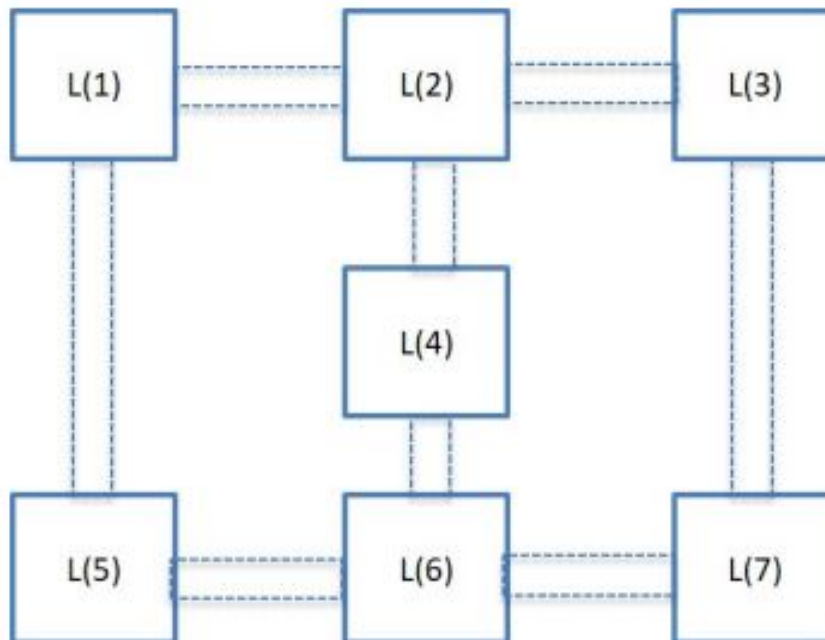


Figure 2.4: Example of a Topological map from [Zhu *et al.* \(2019\)](#)

Figure 2.4 demonstrates that a topological map only exhibits the core rooms and their interconnections, without considering the size of the rooms or the distances between them. The map's intent is to convey how the rooms are linked to each other, rather than providing specific measurements or dimensions.

2.2.1.2 Geometrical Localization

Geometric localization is a process of determining an object's position or spatial coordinates in a three-dimensional space. This type of localization relies on geometric information, such as distances or angles, to calculate the object's position relative to a predefined reference system.

In the context of indoor localization, geometric localization is often used to determine the location of mobile devices or objects within a building or enclosed space. This can be done using various technologies and techniques, including:

1. Trilateration: uses distances measured from at least three known landmarks (such as Wi-Fi access points or Bluetooth beacons) to calculate the object's position through the intersection of spheres or circumferences. Trilateration for example is used in [ilçi *et al.* \(2015\)](#) where a geometric localization is made using Received Signal Strength (RSS) measurements taken from Wi-Fi. The RSS measurements make it possible to calculate the distance of the object from the sent signal and consequently obtain the position of the object as in Figure 2.5

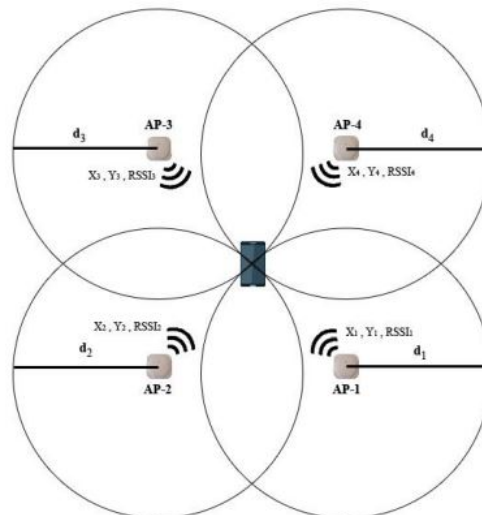


Figure 2.5: Example of Trilateration from [ilçi *et al.* \(2015\)](#)

2. Triangulation: uses angles or directions measured from at least three known landmarks to calculate the position of the object through the intersection of triangles. Triangulation is a very complex but also a very precise method. Because it is so complex as an approach, the data obtained from the sensors can be processed in a variety of different ways to obtain the most accurate position possible. For example, five methods of analysis are even presented in [Montanha *et al.* \(2019\)](#). In Figure 2.6 it is possible to observe the result of the method based on Polar Points Centroids.

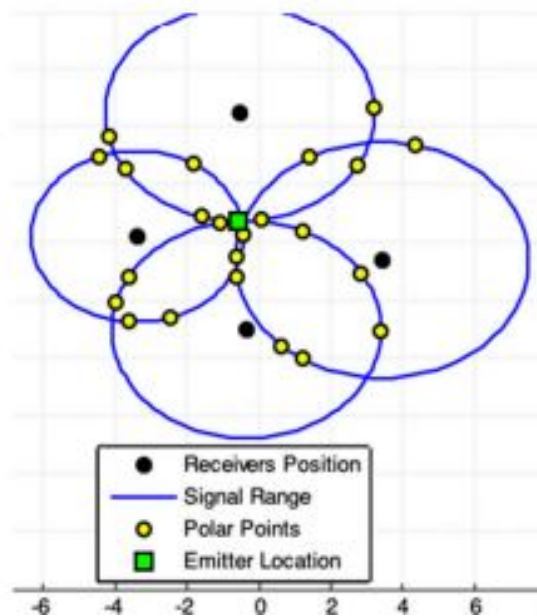


Figure 2.6: Example of Triangulation from [Montanha *et al.* \(2019\)](#)

3. Fingerprints: A technique for tracking the location of devices inside a building or an enclosed space involves using reference data, also known as "fingerprints," that were previously gathered. The process of collecting this data usually involves mapping the environment where location tracking is needed. The data is then utilized to create fingerprints for each location which contain information about the strength of the wireless signal in that specific area. Fingerprints are a popular approach for indoor localization because, when properly calibrated and mapped, they can be fairly precise.

2.2.1.3 Approaches

Indoor localization methods can be different and are often combined to achieve optimal results.

1. PIR (Passive InfraRed): Locating people indoors has become an increasingly relevant challenge in recent years. One of the methods used to address this challenge is the use of PIR(Passive InfraRed) sensors and machine learning algorithms. PIR sensors are devices that can detect human presence by analyzing heat changes in the surrounding environment. These sensors are widely used for lighting control and security, but they can also be used to locate people as in [Lai *et al.* \(2018\)](#).

The use of machine learning algorithms allows meaningful information to be extracted from the data collected by PIR sensors. These algorithms can learn the movement patterns and spatial characteristics of people within a specific environment. For example, the algorithm can be trained on training data that contains information about the layout of spaces, such as the position of doors, windows and fixed objects in the environment. Then, the algorithm can be used to analyze PIR sensor data in real time and estimate the position of people within the environment.

The combination of PIR sensors and machine learning algorithms offers many advantages in locating people in indoor environments. This approach can be used for monitoring elderly or disabled people, managing the flow of people in public or commercial buildings, improving energy efficiency through automated control of lighting and heating, and much more. The use of PIR sensors and machine learning algorithms enables accurate and reliable tracking of people, thus helping to create safer, more efficient and personalized indoor environments.

In [Wu *et al.* \(2021\)](#) they propose a non-wearable system for cooperative indoor human localization using a PIR detector and sensing signal processing algorithms. The system utilizes the information of overlapping FOV of multiple sensors to improve localization accuracy, in Figure 2.7 are visible the detected areas. Signal processing algorithms and refinement schemes such as the Kalman filter are used to improve the system performance with good results.

2. RFID (Radio Frequency Identification): The RFID tracking system uses RFID tags (labels) that are attached to objects or worn by people. These tags emit radio signals that can be detected by RFID readers placed at strategic locations in the environment. People's location is determined based on proximity to the RFID readers. This system is explored in [Sanpechuda & Kovavisaruch \(2008\)](#) and it is often used in environments such as hospitals or airports for asset tracking or resource management.

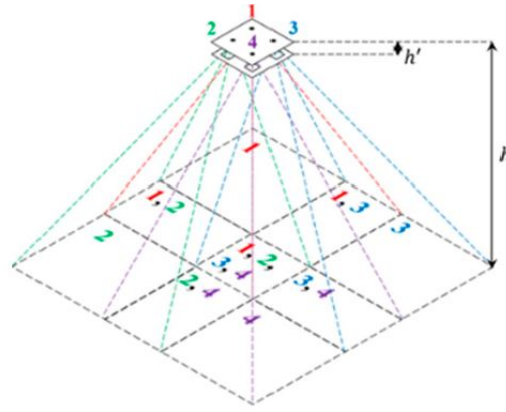


Figure 2.7: Overlapped areas for human detection with PIR from [Wu *et al.* \(2021\)](#)

In [Xu *et al.* \(2018a\)](#) they propose an algorithm to improve the positioning precision of the LANDMARC algorithm [Ni *et al.* \(2003\)](#), which uses RFID tags and readers to implement an Indoor Positioning System (IPS). The proposed algorithm uses the weighted path length and support vector regression to improve the accuracy of the LANDMARC algorithm. The RSSI of the target tag is read in different directions of the antenna, and the position is estimated using support vector regression. In [Figure 2.8](#) is visible a schema of the tests done in which we can see in green are the reference tags that transmit to the antennas the RSSI received from the unknown tag in yellow whose location is desired. These data are subsequently interpolated to obtain the most accurate location possible.

3. Ultrasound: Ultrasonic-based tracking systems use ultrasonic transmitters and receivers placed in the environment. The transmitters emit ultrasonic signals, and the receivers detect the arrival time of these signals from the various transmitters. Using the arrival time of ultrasonic signals, the localization algorithm can calculate the distance between people and transmitters, determining their location.

[Che *et al.* \(2023\)](#) highlights the advantages of UWB technology over narrowband-based technologies such as Bluetooth and Wi-Fi, including a very large bandwidth, very high data rate, short signal transmission length, low transmission energy, and high penetration capability. The paper also discusses the functionality of indoor localization with UWB, including client-based positioning for indoor navigation and server-based positioning for asset tracking. The accuracy of UWB-based IPS is reported to be 10-30 cm,

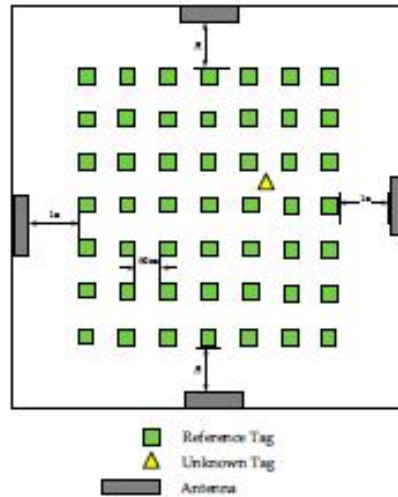


Figure 2.8: Schema of tests done with RFID from [Xu *et al.* \(2018a\)](#)

which is considerably better than other technologies such as beacons and Wi-Fi. In Figure 2.9 is represented in a simple way the proposed localization method.

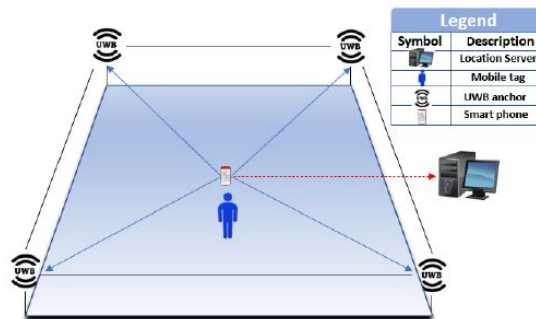


Figure 2.9: Schema of tests done with UWB from [Che *et al.* \(2023\)](#)

4. Bluetooth Low Energy (BLE): BLE is a short-range communication technology that is used in many mobile devices. Using the BLE signal emitted by mobile devices, people's location can be estimated. This system is based on detecting the strength of the BLE signal received from BLE beacons placed in the environment. The location algorithm uses the received signal strength to calculate the distance between people and the beacons, and then estimate their location like in [Baronti *et al.* \(2018\)](#).

2.2 Activity Recognition

In [García-Paterna *et al.* \(2021\)](#) the authors propose a simple and low-cost architecture, visible in Figure 2.10, for a room-level indoor positioning system based on Bluetooth Low Energy (BLE) technology. The system is designed to provide location services for BLE-enabled smart devices worn by the person to be localized. The paper describes the design, installation, and testing methodology of the system, which has been developed for simplicity, ease of installation, development, maintenance, and low cost. The authors also provide the design decisions and explain them in detail, highlighting key ideas and lessons learned to reproduce the solution. The main contributions of the paper are the proposed architecture for a room-level indoor positioning system based on BLE technology, the design decisions, and the testing methodology.

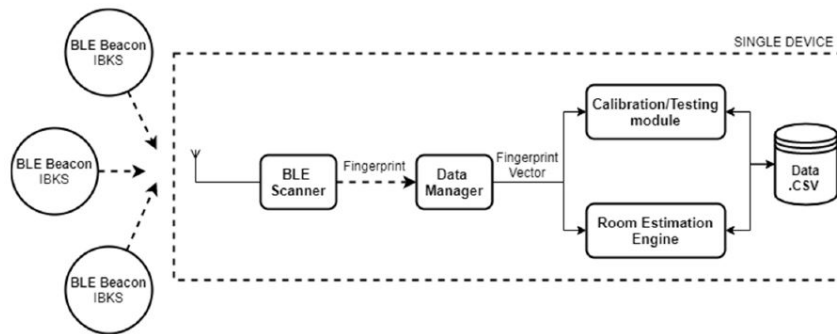


Figure 2.10: System architecture with BLE from [García-Paterna *et al.* \(2021\)](#)

5. Wi-Fi: Wi-Fi-based location systems take advantage of Wi-Fi networks in the environment to estimate the location of people. This system uses Wi-Fi signals emitted from mobile devices or Wi-Fi access points placed in the environment. The location algorithm uses the strength of the Wi-Fi signal received from mobile devices to calculate the distance and estimate the location of people.

[Roy & Chowdhury \(2022\)](#) provides an overview of WiFi-based indoor localization systems for smartphones based on RSSI values. The paper discusses the advantages of using smartphones for indoor localization and the challenges associated with it. The paper also provides a comprehensive review of the existing literature on WiFi-based indoor localization systems, including the techniques used for localization, the algorithms used for data processing, and the accuracy of the systems. The paper concludes that WiFi-based indoor localization systems are a promising approach to achieve ubiquity since

2.3 Machine Learning for Activity Recognition

smartphones are widely available today and WiFi access points are ubiquitous. The paper also highlights the need for further research to improve the accuracy and reliability of WiFi-based indoor localization systems.

The PIR sensor approach to indoor localization has unique advantages that make it preferable to other methods such as RFID, ultrasound, BLE, and Wi-Fi. PIR sensors offer simple and cost-effective implementation, allowing for rapid installation and requiring little maintenance. In addition, because of their ability to detect body heat, PIR sensors are particularly effective at detecting the presence of people, making them suitable for low-light environments. Unlike RFID, the PIR approach does not require the installation and management of tags or labels on the subjects to be located. Compared with ultrasound, the use of PIR sensors does not require the installation of numerous transmitters and receivers, reducing the cost and complexity of the system. In addition, PIR sensors are not affected by ultrasonic reflections or diffractions, providing greater reliability in localization. Compared with BLE and Wi-Fi, the PIR approach does not require additional mobile devices or access points, simplifying deployment and reducing costs. Finally, PIR sensors offer an immediate and real-time response, enabling rapid and responsive localization. In conclusion, the PIR sensor approach stands out for its simplicity and efficiency in indoor localization, making it a preferable choice over other solutions such as RFID, ultrasound, BLE, and Wi-Fi. The only aspect that slightly penalizes the PIR approach compared to others is accuracy, this gap, however, can be easily bridged by using other types of sensors due to the versatility of the approach itself.

2.3 Machine Learning for Activity Recognition

Machine Learning (ML) changes in a consistent manner the way we interact with technology and the way automated systems can learn from data to make decisions and take intelligent actions. One of the most relevant areas where ML has found application is in human activity recognition. This field of research focuses on the development of machine learning algorithms and models that can identify and classify different activities performed by an individual or a group of people.

Human activity recognition is a complex and interesting challenge. It has a wide range of practical applications ranging from areas such as robotics and industrial automation to human health and well-being. For example, in a collaborative robotics context, an activity recognition system can enable a robot to understand human actions and intentions, thereby improving human-machine interaction. In drones and autonomous vehicles, activity recognition can help

2.3 Machine Learning for Activity Recognition

better understand the behavior of pedestrians and other vehicles, increasing road safety.

In the medical field, activity recognition can be used to monitor the well-being of the elderly and patients by detecting abnormal behavior or emergency situations. For example, a home monitoring system could recognize when a person falls and send a distress notification. In addition, activity recognition is used in the field of security surveillance to identify suspicious or potentially dangerous activities, thereby improving the safety of public and private environments.

There are several Machine Learning approaches used for human activity recognition like in [Jain *et al.* \(2023\)](#). Each approach has its strengths and weaknesses, and the choice often depends on the nature of the data available and the specific needs of the application. Among the main approaches:

1. **Structured Data-Based Classification:** This approach involves the use of classification algorithms such as Support Vector Machines (SVM) or Random Forest to label tasks. The data used to train these algorithms can be represented by structured features extracted from raw data or sensors. This approach is particularly useful when activities can be distinctly separated into well-defined categories.
2. **Deep Learning:** Deep neural networks, a subfield of Machine Learning, have been shown to be extremely powerful in activity recognition. In particular, convolutional neural networks (CNNs) are used to automatically extract meaningful features from raw data, such as images or sensor data. Recurrent neural networks (RNNs), on the other hand, are suitable for recognizing activities based on temporal sequences of data. Deep neural networks are prized for their ability to learn complex representations and have achieved outstanding results in many applications, such as human activity recognition.
3. **Unsupervised learning:** Some activity recognition approaches use unsupervised learning techniques, such as clustering or dimensionality reduction. These approaches try to discover hidden patterns or structures in the data without the use of labels. Unsupervised learning is useful when activity labels are expensive or difficult to obtain.

The following are examples of algorithms used for activity recognition

1. Random Forest is a widely used machine learning algorithm for classification and regression [Breiman \(2001\)](#) [Biau \(2012\)](#). It is based on a technique

2.3 Machine Learning for Activity Recognition

called "bagging," which combines the results of many decision trees (known as "weak" decision trees) to improve the overall prediction. In a Random Forest, many decision trees are created using random subsets of the training data and features, thus reducing the risk of overfitting (overlearning) and improving generalization. When making a prediction, each decision tree contributes its vote, and the final result is a weighted combination of these votes. This technique is effective in handling complex data and noise in the data and is used in a wide range of machine learning applications, from image analysis to financial forecasting. It is also a very efficient and fast algorithm as described in [Nurwulan & Selamaj \(2020\)](#). In [Xu *et al.* \(2018b\)](#) for example, it is used to do activity recognition through the use of accelerometers attached to various parts of a person's body. Below in [Figure 2.11](#) is an example of the operation of the Random Forest.

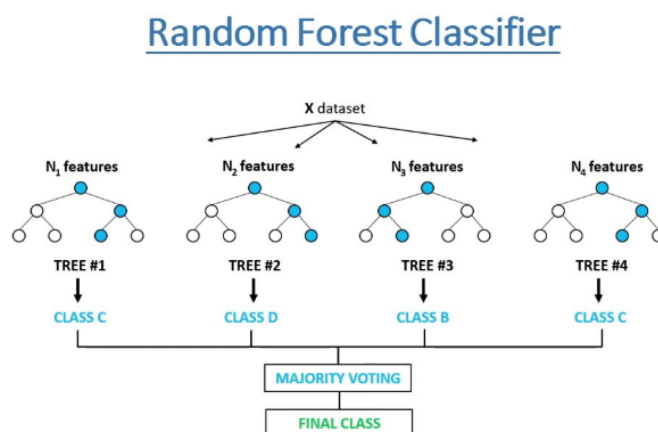


Figure 2.11: Example of Random Forest

2. Convolutional Pose Machine (CPM) is a deep learning model used in the field of human pose recognition and extraction from images or videos. This approach relies on convolutional neural networks (CNNs) to identify key joints and positions of the human body in an image [Thilakarathne *et al.* \(2022\)](#). Unlike other architectures, such as kinetic chain models, CPM is able to capture the spatial relationships between joints, thus improving the accuracy of pose recognition. This makes it useful in applications such as motion monitoring, gesture analysis, camera-based surveillance, and advanced human-computer interaction. The model takes advantage of the ability of CNNs to automatically learn features from images, making CPM a popular choice for computer vision problems related to human pose recognition. It is often used for activity recognition based on human posture, for

2.3 Machine Learning for Activity Recognition

example, identifying specific gestures or postures. An example of this is in [Wei *et al.* \(2016\)](#) where was tested the implementation on the MPII Human Pose dataset which consists of more than 28000 training samples and contains images of a lot of different activities. In [Figure 2.12](#) we can see the pose machines flow chart from the paper.

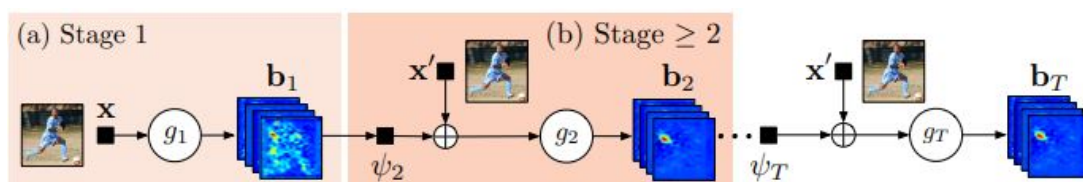


Figure 2.12: Pose machines flow chart

3. LSTM, short for "Long Short-Term Memory," is a type of recurrent neural network (RNN) widely used in the field of machine learning, especially in natural language processing (NLP) applications and time series recognition. What makes LSTM particularly powerful is its ability to capture and maintain long-term information within the network, avoiding the "gradient disappearance" problem that plagues traditional RNNs, problem that is explored and solved with a LSTM in [Murad & Pyun \(2017\)](#). This architecture can store and retrieve information from previous time steps, which is critical for tasks such as machine translation, text prediction and speech recognition. LSTM has specialized internal layers called "gates" that regulate the flow of information within the network, making it capable of learning complex sequences and maintaining long-term memory, making it a critical tool for sequence-based machine learning applications. In [Figure 2.13](#) we can see an example of a flow chart of LSTM. An example of using LSTM for activity recognition is in [Pienaar & Malekian \(2019\)](#) where they use a dataset that includes labels for each of six activities, each including the x, y and z axis values for the tri-axial accelerometer during the labeled activities. The activities available in the dataset include standing, sitting, walking, jogging, ascending and descending stairs.
4. SVM, short for "Support Vector Machine," is a machine learning algorithm used for classification and regression problems. Its distinguishing feature is its ability to find an optimal "separation hyperplane" between two classes of data in a multidimensional space. The main goal of SVM is to maximize the distance between training examples closest to this hyperplane, called "support vectors," which helps to improve the generalization of the model. SVM

2.3 Machine Learning for Activity Recognition

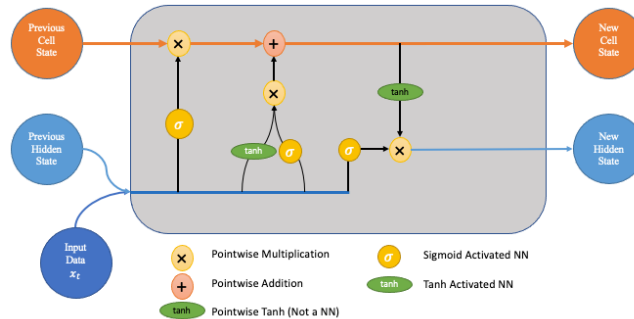


Figure 2.13: LSTM flow chart example

can also be extended to handle nonlinear classification problems through the use of kernel functions, which map data into a high-dimensional space so that they become linearly separable. This makes SVM a powerful tool for complex data classification and for problems where clear separation between classes is essential, such as image recognition, text classification, and computational biology. In Figure 2.14 we can see the way in which SVM works. In [Nawal *et al.* \(2023\)](#) discusses the use of SVM as a method for activity recognition of activities such as cooking, leaving home, sleeping and eating in a smart home.

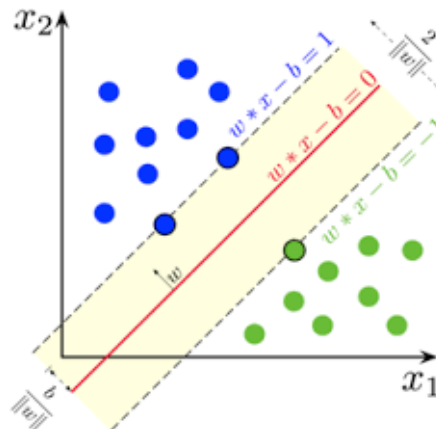


Figure 2.14: SVM functioning

5. Hidden Markov Models (HMMs) are probabilistic models used in machine learning and sequential pattern recognition applications. These models consist of two main components: a hidden state (hidden state) and an emitted observation (emission). Hidden states represent the unobservable or latent information in a system, while emitted observations are the information

2.3 Machine Learning for Activity Recognition

that can be measured or observed. HMMs are often used to model data sequences, such as natural language recognition, handwriting recognition, speech recognition, and more. One of the key features of HMMs is the ability to model the temporal dynamics of sequences, as the hidden states are linked together through probabilistic transitions. These models have been successfully used in a wide range of applications where it is important to handle uncertainty and complex data sequences. In Figure 2.15 there is an example of an HMM scheme. In [Abreu *et al.* \(2019\)](#) they use sensors like accelerometer, gyroscope, magnetometer and microphone to create a new dataset, with 10 complex activities, such as opening a door, brushing teeth and typing on the keyboard. The classifier was based on multiple hidden Markov models, one per activity.

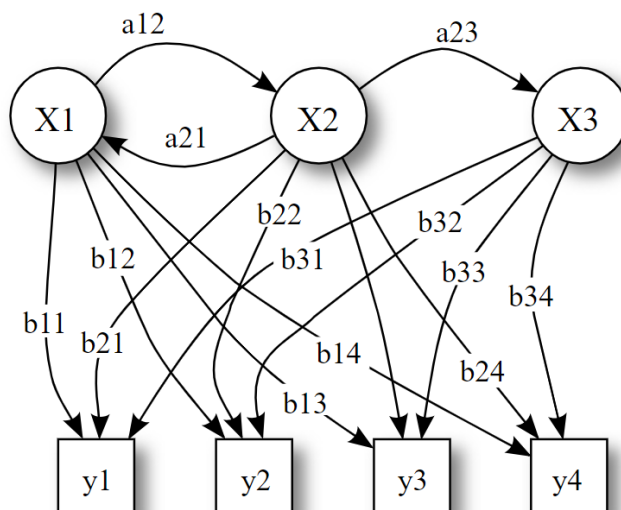


Figure 2.15: HMM scheme

6. Transformers models are a type of neural network architecture that has revolutionized the field of machine learning, particularly in the area of natural language. These models are known for their ability to capture long-term relationships in data sequences through the attention mechanism, which allows different parts of an input to be given weight based on their relative relevance. Transformers are widely used in natural language processing applications, such as machine translation, text recognition and text generation, and have led to state-of-the-art results in many of these areas. In addition to NLP, Transformers models have also been successfully adapted for other applications, such as image processing and time series prediction, demonstrating their versatility and power in the field of machine learning. A

2.3 Machine Learning for Activity Recognition

notable example of a Transformer model is BERT (Bidirectional Encoder Representations from Transformers), which has set new standards in the field of NLP. In [Gavrilyuk *et al.* \(2020\)](#) the object is to recognize individual actions and group activities from videos using Transformers. In [Figure 2.16](#) we can see the scheme of functioning of the approach used in [Gavrilyuk *et al.* \(2020\)](#).

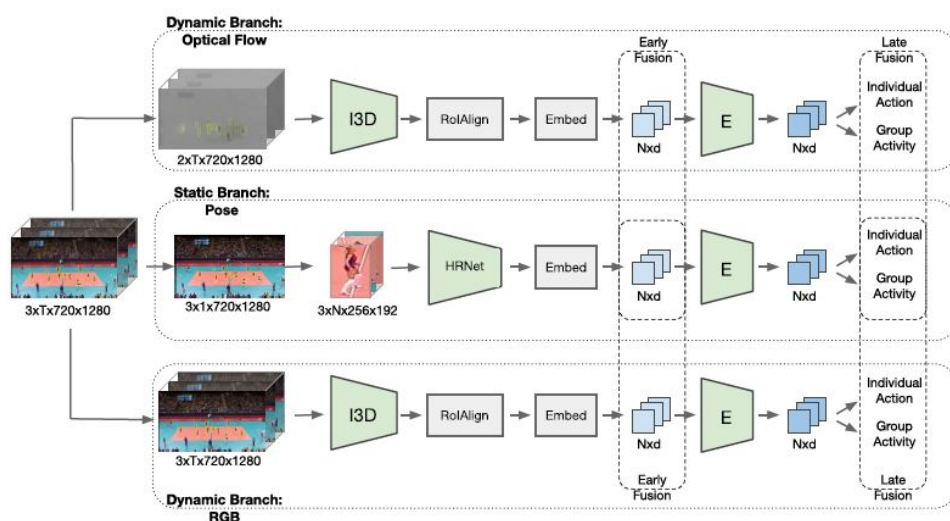


Figure 2.16: Transformers scheme from [Gavrilyuk *et al.* \(2020\)](#)

As previously alluded to, the significance of location in activity recognition cannot be overstated, and the advent of Machine Learning has played a pivotal role in advancing this field, enabling the development of increasingly sophisticated algorithms and systems for precisely determining the position of objects or individuals within enclosed spaces, such as buildings, shopping malls, airports, and factories, some methods are highlighted in [Nessa *et al.* \(2020\)](#) [Zhang *et al.* \(2017\)](#) [Roy & Chowdhury \(2021\)](#).

It is essential to emphasize that indoor tracking faces a unique set of challenges, as traditional GPS technology becomes ineffective in these environments. Thus, alternative technologies come into play, including Passive Infrared Sensors (PIRs), Wi-Fi networks like in [Salamah *et al.* \(2016\)](#), Bluetooth Low Energy (BLE), Radio-Frequency Identification (RFID), and inertial sensors, which serve as data collection tools. Nonetheless, due to the inherent noise and potential inaccuracies associated with such data sources, the incorporation of Machine Learning algorithms becomes imperative to effectively handle uncertainties and deliver highly accurate position estimates.

2.3 Machine Learning for Activity Recognition

Below, we will explore some of the most common and widely utilized approaches for indoor tracking. While some of these methods have been previously discussed, we shall now consider them from a different perspective, particularly in how they synergize with Machine Learning algorithms to enhance their efficacy:

1. **RSSI-based fingerprinting:** In this approach, location data (fingerprints) are collected from various Wi-Fi access points or Bluetooth beacons within the environment like in [Singh *et al.* \(2021\)](#). The collected data include the received signal strength (RSSI) from each access point or beacon. These fingerprints are used to build a reference model that maps RSSI readings to specific locations. Machine Learning algorithms, such as k-Nearest Neighbors (k-NN) or Support Vector Regression (SVR), are then used to estimate the location of an object or device based on the RSSI readings received.
2. **Trilateration:** This method uses distance measurement signals, such as Bluetooth BLE signals or ultrasonic waves, to calculate the position of a device based on the distances detected by beacons or sensing sensors. Machine Learning algorithms in this case are used to improve the accuracy of trilateration by also considering other factors, such as signal attenuation due to obstacles or fluctuations in the environment.
3. **Triangulation :** This method uses angles or directions measured from at least three known landmarks to calculate the position of the object through the intersection of triangles. In this case, Machine Learning algorithms are used to compensate for possible errors and noise in data acquisition and to estimate possible positions even when the object or person is not correctly identified by at least three landmarks.
4. **Sensor Fusion:** Indoor localization can be improved by fusing data from different sensor sources, such as accelerometers, gyroscopes, and magnetometers, in addition to Wi-Fi or BLE connectivity data. The combined use of this data can provide a better understanding of object or user motion, enabling more accurate localization. Machine Learning algorithms, such as Kalman filters or neural networks, can be applied to efficiently integrate and combine information from different sensors.
5. **Map Matching:** This approach is based on the use of maps of the indoor environment to improve location accuracy. Sensor data and RSSI readings are combined with information about the geometry and distribution of access points or beacons in the environment. Machine Learning algorithms can be used to fit the observed data to the map information and provide a more accurate estimate of location.

2.4 Large Language Models

Large Language Models (LLMs) represent a revolution in artificial intelligence and natural language. Developed in recent years, these models have led to significant advances in the ability of machines to understand, generate and interact with human language.

LLMs are artificial intelligence systems that use deep learning techniques to learn from large amounts of text. These models are trained on huge datasets containing billions of words from a wide range of sources, such as books, articles, Web pages and more. This volume of data allows the models to capture the complexity and richness of human language, gaining in-depth knowledge of grammatical structures, semantic relationships and linguistic peculiarities.

One of the first and most famous Large Language Models was the GPT-2 (Generative Pre-trained Transformer 2) developed by OpenAI. Later, it was followed by even larger models such as GPT-3 (Generative Pre-trained Transformer 3), on which, the GPT-3.5 AI Assistant, is based. These LLMs contain hundreds of billions of parameters, enabling them to generate incredibly realistic and consistent text.

LLMs can be used in a variety of contexts and have a wide range of applications. One of the most common uses is automatic text completion, where a model suggests the completion of a sentence or paragraph based on the context. This feature is very useful for speeding up the writing process and reducing grammatical errors.

Other important uses include machine translation, sentiment analysis, creative text generation and more like in [Wang *et al.* \(2022\)](#). LLMs have demonstrated an amazing ability to adapt to different language tasks without having to be reformed or trained specifically for each one. Recently, Large Language Models (LLMs) have been proposed as valuable alternatives for various activities, such as planning [Scalmato *et al.* \(2013\)](#)

However, with the opportunities that these models offer, some concerns also arise. The very size of LLMs requires an enormous amount of computational resources for training, making the process environmentally costly.

To attenuate these concerns, the scientific community and companies are working to develop more energy-efficient models. At the same time, research continues to explore new ways to ensure that LLMs are ethical, transparent and

aligned with human interests.

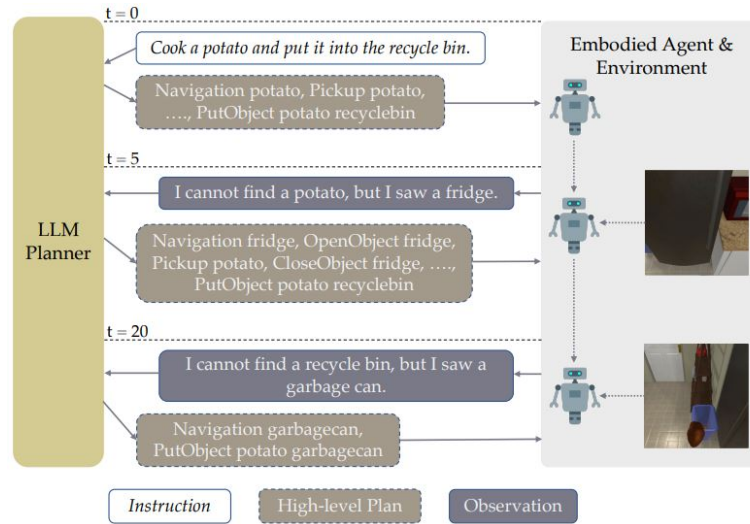


Figure 2.17: Prompt schema of the LLM-Planner

[Song *et al.* \(2023\)](#) proposes a novel method for few-shot planning for embodied agents that can follow natural language instructions to complete complex tasks in a visually perceived environment. The method, LLM-Planner, harnesses the power of large language models (LLMs) to generate and update plans that are grounded in the current environment. The paper also proposes a simple but effective way to enhance LLMs with physical grounding. In Figure 2.17 is visible the way the proposed solution works, especially the study that is done on the prompt of the LLM being considered is important (GPT3). The final optimal prompt includes an intuitive explanation of the task, the list of allowable high-level actions, in-context examples selected by the kNN retriever, the subgoals that have been completed, and the list of objects observed so far in the environment after the task description.

In the end, Large Language Models represent an important evolution in artificial intelligence and natural language. Because of their potential, they can transform the way people interact with digital technologies and open new horizons in language understanding and processing. However, it is essential to address the ethical and technical challenges that arise with their use, ensuring that they are a safe, reliable, and beneficial tool for society.

Chapter 3

Software Architecture

The software architecture was designed and developed based on Flask, a Web framework for Python with a number of features that make it an ideal choice for creating a robust and scalable system. Flask, despite its lightweight nature, provides a powerful framework for managing client-server communications efficiently and effectively.

One of the main qualities of Flask is its modularity. This framework allows code to be broken down into separate modules, each of which can handle a specific aspect of the application. This makes system design and development extremely flexible and makes it easy to adapt the server to the growing needs of users. In addition, Flask offers a wide range of extensions and libraries that simplify the management of common features such as user authentication and data manipulation.

Another key aspect of Flask is its ability to create RESTful APIs in an intuitive way. This is especially important if you want to provide Web services to allow clients to interact with the server in a structured and consistent way. Flask makes defining API endpoints, handling requests and JSON responses, and validating input data a smooth process.

In addition, Flask's lightness does not mean compromising security. The framework offers mechanisms that handle common vulnerabilities ensuring that your system is robust and reliable.

Choosing to base the architecture on Flask offers many advantages, including flexibility, scalability, security, and ease of development.

The final proposed architecture went through multiple ideas and evolutions

before being settled upon. The architecture underwent significant changes from the initial concept to the final design. Originally, a client-server architecture was not considered, but instead, there was a plan to have a single module that could collect data from proprietary clouds and use Machine Learning algorithms or Large Language Models to make predictions with newly obtained data.

To enable the learning of Machine Learning algorithms, it was imperative to develop a database for efficient data storage. Furthermore, a real-time database was also required. It was crucial to have a comprehensive database that incorporated all the data from the manufacturers' cloud to enable the proper functioning of the two reasoning approaches and to maintain the activation history of the sensors. As the manufacturers' clouds only retain the last data sent by the sensor and not the activation history, an internal database was considered necessary.

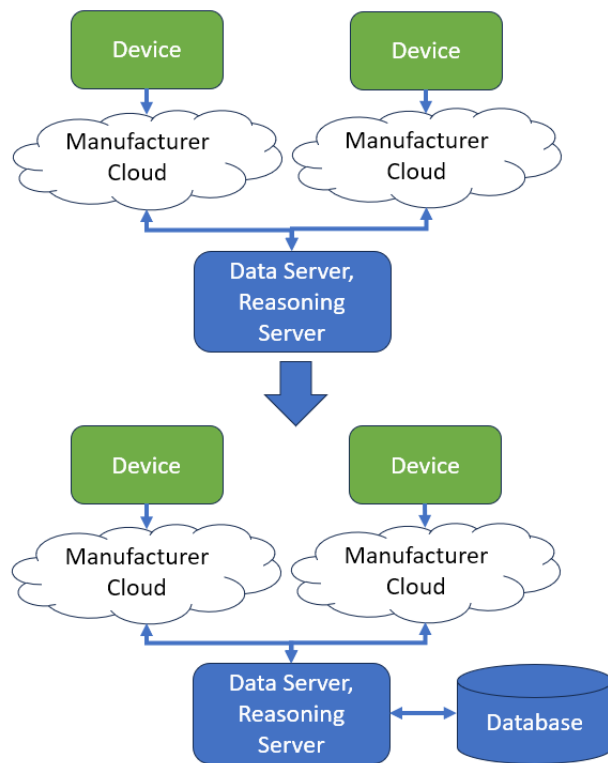


Figure 3.1: First evolution of the architecture.

Upon thorough consideration, it was concluded that dividing the data acquisition and reasoning activities into separate modules would be the most efficient approach. This not only enhances modularity but also facilitates future upgrades

or troubleshooting without interruption to either process. Consequently, two modules were devised: one to collect data from producer clouds and feed it into the database (Data Server), and another to analyze the data with its own set of algorithms (Reasoning Server). This configuration closely resembles the ultimate design.

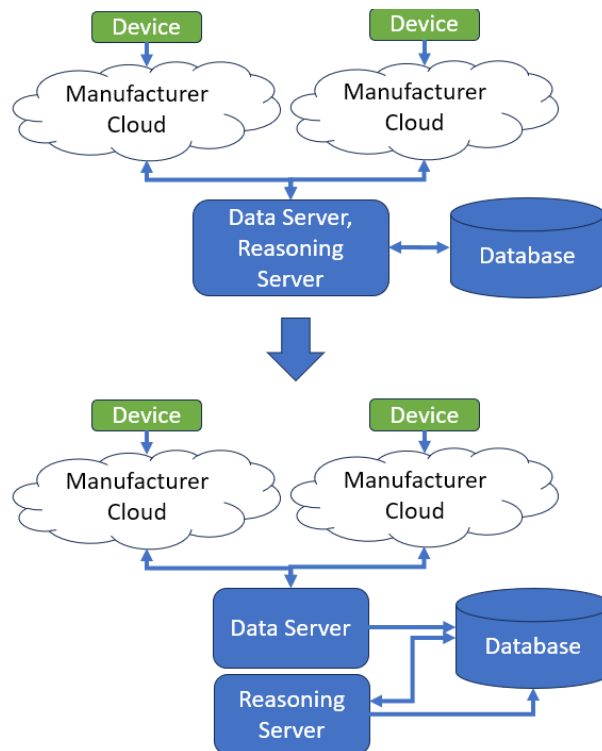


Figure 3.2: Second evolution of the architecture.

Our current plan involves activating both the Data Server and Reasoning Server simultaneously, in order to gather new data and generate fresh predictions. However, this approach poses two challenges. The first is that the data collection module gathers information from all sensors connected to the architecture, even those in different environments, resulting in excessive computational energy consumption. To address this issue, we need to modify the Data Server to differentiate between environments and end data acquisition in specific situations. The second challenge is that the predictions generated by the Reasoning Server are stored in a database that is only accessible by components on the same machine. This limits the usefulness of the predictions for other devices, such as a robot, that may require this data in a different format and cannot directly access the machine.

In order to resolve the aforementioned problems, a client-server architecture has been suggested. The servers are comprised of the data collection (Data Server) and reasoning (Reasoning Server) modules, while clients are utilized to address problems. The Data client is responsible for managing the activation of specific environments to collect data. For Reasoning Client, it requests information from the server and receives prediction data in response. This facilitates a link between the machine and external sources. For instance, a robot with internet connectivity can readily acquire predictions from the Reasoning Server by making a request.

Having thus presented the evolution of the architecture, it is now time to go and see the final architecture in detail. The proposed method features the use of PIR sensors, magnetic door sensors, and cameras of a robot for data acquisition in the environment. On the software side, the architecture includes two main servers, one for data collection (Data Server) and the other for reasoning about the data (Reasoning Server), a Database for data storage, and two clients interacting with the two servers respectively (Data Client and Reasoning Client). The architecture is visible in Figure 3.3.

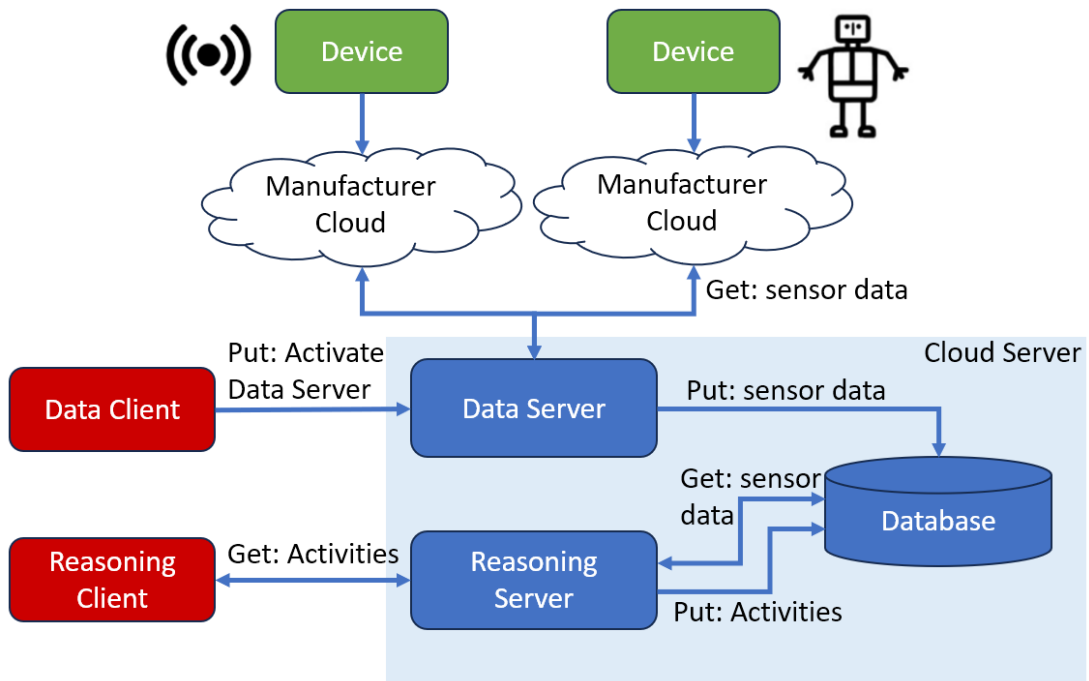


Figure 3.3: Cloud architecture for data acquisition and activity recognition.

Devices: Sensing devices acquire data periodically or upon request, typically uploading data to cloud servers for data collection provided by the manufacturer.

While the architecture we propose is general and can accommodate various types of devices, our current setup includes three types of devices. PIR sensors, when strategically placed at points of interest, provide valuable information about ongoing activities. Magnetic door sensors complement PIR sensors for localization, enhancing the amount of data available for reasoning.

Finally, to avoid the potential perception of intrusiveness in a domestic environment, we have chosen to incorporate a humanoid social robot equipped with onboard cameras. The robot’s cameras, when the robot approaches a person and is welcomed, allow us to capture the relative orientation of the head and the eyes, providing additional information to better understand the activities taking place.

Data Server: As mentioned earlier, a common trend in mobile sensing technology is that wireless sensors periodically upload the data they acquire to a private cloud maintained by the manufacturer that developed them. Users are then provided with apps (or APIs) to query the cloud and access the collected data. Following this approach, the Data Server serves as an access point for gathering data from various manufacturers’ clouds. This server is responsible for authenticating with the manufacturers’ clouds, accessing the relevant data, and transferring it to a centralized database, as shown in Figure 3.3. This aggregation process consolidates all the data into one location, simplifying further analysis and processing. Specifically, after initialization, the Data Server periodically collects data from all proprietary servers and stores them in the database

Reasoning Server: The Reasoning Server processes data acquired from sensors and stored in the database. It collects and organizes data, interprets them to classify human behaviors, and draws conclusions. Two types of reasoning approaches were considered:

- Machine Learning Algorithms (ML). To tackle various prediction and classification challenges, two types of ML algorithms were selected: Random Forest and Support Vector Machine (SVM). The use of these algorithms allowed us to address both situations where binary prediction was required and classification situations where specific classes had to be identified. To this end, we needed to acquire a dataset in the Smart Environment, and then train Random Forest and SVM to learn a function mapping inputs (sensor data) to outputs (relevant activities and locations).
- Large Language Models (LLM). Reasoning based on LLMs ??? represents

a novel approach to data analysis for activity understanding. This approach relies on the concept of representing sensor data in natural language to construct prompts (e.g., “Prompt: the PIR in the kitchen detected somebody, and it is noon. What’s happening?”) and then using LLMs to comprehend and interpret free text. This enables a deep analysis of qualitative data, ultimately allowing the detection of the context in which this data was acquired (“Reply: Perhaps somebody is having lunch”). In this case, there is no need for specific model training for the environment since the model relies on the common-sense reasoning capabilities embedded in language.

The Reasoning Server periodically acquires data from the database, reasons upon them, and stores results in the database.

Database: The database is essential for real-time data storage, and its structure allows for a general view of the various environments, where “environments” are defined, for example, as different houses or workplaces. The database includes:

- a table listing all the environments and their statuses;
- tables for each environment summarizing all the sensors present in each of them;
- tables for each sensor that store the history of their activations;

The database is managed using SQLite.

Data Client: The Data Client interacts with the Data Server. When a user or system intends to initiate data acquisition from specific environments or sensors, the Data Client sends a request to the server, which server responds by providing the collection of relevant data.

Reasoning Client: The Reasoning Client is responsible for interacting with the Reasoning Server. A user or a system can send requests to this client: the client forwards the request to the Reasoning Server, which processes the data in real-time using the models mentioned before and returns results.

Data Flow: In Figure 3.4, it is possible to observe how data travels within the architecture.

1. The sensors send data to the manufacturer’s cloud.
2. The data server requests data from all sensors to the manufacturer’s cloud

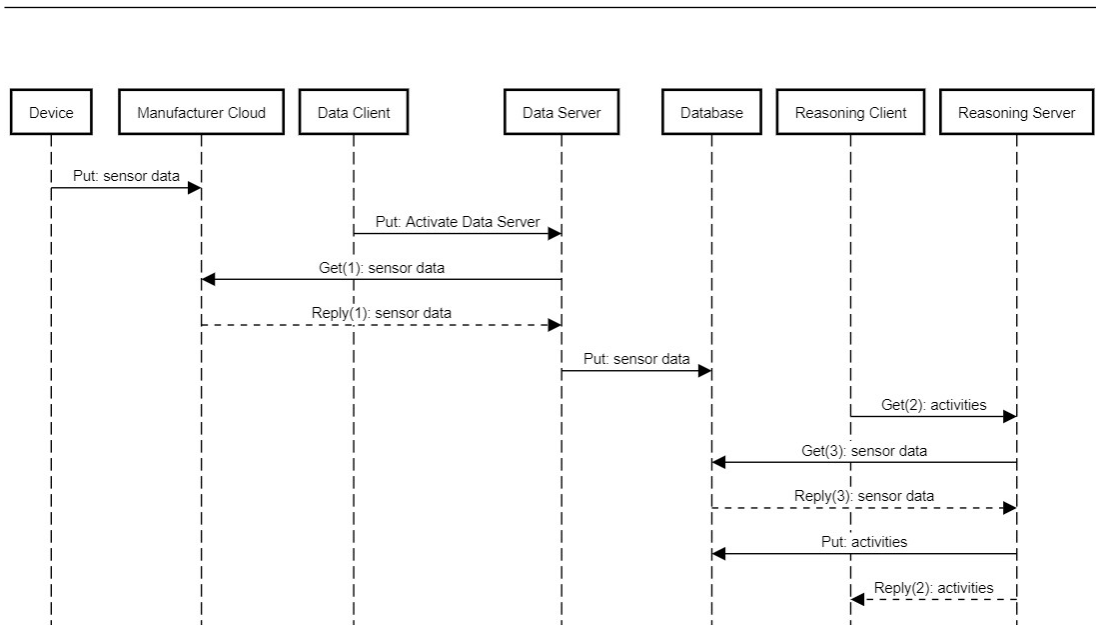


Figure 3.4: Sequence diagram showing data exchanges between architecture's components.

3. The manufacturer's cloud responds by sending the requested data
4. The Data Server acquires the data and writes them to the Database
5. The Reasoning Client requests the predictions from the Reasoning Server
6. The Reasoning Server reads the data from the database
7. The Reasoning Server processes the data and sends the predictions made to the Reasoning Client and stores them in the Database

In addition, a note should be made about the type of data that is exchanged within the architecture.

- the Data Server takes individual values from the Manufacturer's clouds, one for each sensor, and then the same values go and insert them individually into the appropriate database tables
- the Data Server takes individual values from the Manufacturer's clouds, one for each sensor, and then the same values go and insert them individually into the appropriate database tables
- the Reasoning Server sends to the Reasoning Client and Database an array of seven elements containing the predictions of all activities

This software architecture is designed to enable the collection, analysis, and prediction of data from distributed sensors in various environments. Using a combination of servers and clients, along with machine learning algorithms, the system provides an effective way to obtain valuable information from large amounts of data. The architecture is scalable, as new sensors and environments can be easily integrated into the system, and machine learning models can be adapted and improved over time to provide increasingly accurate results.

3.1 Sensors

The described architecture requires a variety of sensors to collect data from complex environments. In particular, Passive Infrared (PIR) sensors, Magnetic Door sensors and cameras are used to monitor human presence and activities within environments. These sensors represent an important part in the analysis of human interaction with the surrounding environment

PIR Sensors: PIR sensors detect infrared radiation emitted by the human body and other heat sources in the surrounding environment. These sensors are very sensitive to changes in temperature, and as a result, they detect human presence in the surroundings with great accuracy. In the described architecture, PIR sensors are strategically distributed to cover different areas within the environment. This arrangement makes it possible to obtain information about human presence and activity in various segments of the monitored environment. Specifically, the PIR sensors used are from Anvek and are part of the family of sensors managed by TUYA which provides the cloud services for sensor data acquisition and management. These sensors have a detection distance of 12 meters and an angle of 110°.

Magnetic Door Sensors: The integration of magnetic door sensors with PIR sensors is a step forward in architecture optimization. These sensors monitor the open/closed status of doors and windows in the environment. The use of magnetic sensors provides a richer context for PIR sensor readings. For example, when a door is opened or closed, the data collected by the PIR sensors can be associated with a specific area of the environment, thus providing greater accuracy in locating human activities. The magnetic door sensors used are Wi-Fi DW2s from SONOFF which are managed through eWeLink servers that enable data acquisition and management.

Cameras: The introduction of cameras for the acquisition of eye and head orientation data represents a significant step forward in enriching the understanding of human interaction with the environment. These cameras use eye and head motion tracking technologies to accurately detect a person’s gaze direction and head position. These data are important for understanding how people move and interact with the environment, enabling additional levels of detail in analyses of human activities. In this case, an HD wireless mini camera from Peecla is used, which connects to the network and sends images to the manufacturer’s cloud. Unlike the other types of sensors, for the camera, there is no API to use to get the images but just connect to a specific IP address to get live streaming of the images that the camera is receiving. From the streaming of images, six values are obtained.

- Distance of the person from the room and thus from the robot
- head yaw
- head pitch
- head roll
- gaze yaw
- gaze pitch

In particular, the five angles obtained are important in trying to interpret what the person is doing based on the context surrounding the person.

In the end utilizing PIR sensors, magnetic door sensors, and cameras to collect eye and head orientation data is a highly effective means of sensing and monitoring. With these diverse sensors at hand, valuable insights can be gained on human presence, location, and activities within a given environment. By combining these technologies, a thorough and accurate understanding of the dynamics and behaviors within monitored spaces can be achieved.

3.2 Data Acquisition

The data acquisition phase represents an important part. The selected sensors, all of which operate on battery power or are connected to power, introduce a flexible and scalable approach to data collection, minimizing the use of complicated cabling and enabling easy installation in various settings. The key element that

makes this data acquisition possible is the connection to the Internet, which provides the bridge between the sensors and their respective manufacturers' clouds.

Data Acquisition Modes: The sensors operate autonomously, constantly monitoring their surroundings for significant changes and activity. With their battery- or grid-powered capabilities, the sensors are able to operate continuously without interruption. Once changes in data, such as human presence or other specific metrics, are detected, the sensors begin the data transmission process.

Internet Connectivity and Data Transmission: Internet connectivity is the hub that enables sensors to communicate with the manufacturer's cloud. The sensors, after sensing significant data, transmit this information to remote servers via the Internet. This step is important to ensure that the data are updated in real-time in the respective manufacturers' clouds. Regular and timely sending of data is a key element in obtaining up-to-date information about the monitored environment.

Sensor State Storage in Manufacturers' Clouds: Manufacturers' clouds play an important role in storing data from sensors. Whenever a sensor detects a change, it sends data to the manufacturer's cloud, which in turn stores the latest sensor status. This approach ensures that data are always available and accessible, enabling continuous, real-time monitoring of the environment.

Role of the Data Server in Acquisition and Storage: When the Data server is activated by its client, it begins the process of collecting data from the manufacturers' clouds.

The Data Server runs at the following Base URL

Listing 3.1: Data Server URL

```
1 BASE = "http://localhost:8000/"
```

When the Data Client attempts to connect to the Data Server, it includes a payload in the request with the fields "house" and "desiredstatus". The "house" field indicates which environment the Data Server should refer to, while "desiredstatus" indicates whether the status should be online or offline. In other words, it determines whether the data collection of sensors in that environment should be activated or deactivated. The example below shows, if "house1" is the environment, how data collection for sensors in that environment can be activated.

Listing 3.2: Payload example

```
1 payload = {"house": "house1", "desiredstatus": "online"  
            "}"
```

The server, using the appropriate authorization, accesses the data stored in the individual manufacturers' clouds. The collected data is then transferred to the server and stored in a local database. This database managed with SQLite, does not just store the latest status of individual sensors. Instead, it maintains a complete historical track of each sensor's activations over time. This feature provides a historical and detailed picture of the activities and changes detected in the monitored environment. Specifically, this server leverages two APIs, the "Tuya IoT Development Platform" for PIR sensors and "ewelinkAPI" for magnetic door sensors. For both there was a need to create login credentials for the two clouds, so you have access to an APIkey that allows access to the cloud and the use of certain specific requests.

This part of the process is identified by the first four transitions of the Sequence Diagram in Figure 3.4 while as far as the architecture in Figure 3.3 is concerned, the parts involved are those present at the top of the figure i.e. Device, Manufacturer's Cloud and Data Server.

In conclusion, data acquisition in this architecture represents a process that takes advantage of Internet connectivity and the ability of sensors to operate autonomously. This combination provides ample visibility into the monitored environment and allows for up-to-date and historical data to be at hand. Accurate management and storage of data in the local database are critical to ensure that information is always accessible and ready for further analysis and evaluation.

3.2.1 Cloud Architecture

The cloud architecture that acts as a bridge between the sensor manufacturers' clouds and the Data Server is an important element in the infrastructure. This is an adaptable component plays the essential role of obtaining data from the various manufacturers' clouds and delivering it in a consistent and structured manner to the central server.

Access to Manufacturers' Clouds: To acquire data from manufacturers' clouds, the cloud architecture follows a well-defined access process. Each producer requires a custom access method, which means that for each cloud, the

architecture must perform a specific sequence of steps. Initially, to gain access, an authentication process must be performed through a login. Once logged in, an access token is generated that enables the architecture to retrieve data from the manufacturer's cloud. Another important aspect is the indication of the continental cloud to which one refers. This is critical because data may be hosted in different clouds based on geographic location. The cloud architecture must ensure that requests are routed to the correct cloud, ensuring compliance with privacy regulations and data location.

Producer Cloud Structure: Due to the variety in the structure of producer clouds, the cloud architecture must be able to accommodate different models. Some manufacturers may organize data into well-defined categories, while others may require more complex searches to access specific sensor data. In addition, the structure of the data itself may vary from cloud to cloud, requiring different data processing. To obtain specific information about various sensors, the cloud architecture must generate different queries. Depending on the information desired and the structure of the cloud, the requests can vary greatly. This requires careful planning and dynamic processing of requests by the cloud architecture, which must be able to understand and interpret the data structures and requests of the various clouds.

Below you can see specifically the two requests that are made for the two APIs used where "DEVICE_ID" contains the identification code of each sensor.

Listing 3.3: TuyaAPI device status request

```
1 openapi.get("/v1.0/iot-03/devices/{}/status".format(↵
    DEVICE_ID))
```

Listing 3.4: ewelinkAPI device status request

```
1 connection.getDevicePowerState('DEVICE_ID')
```

In particular, these two requests are used to access a specific part of the respective clouds, which is the part that contains all the data about the individual device and contains all the information about it including the status. As it is possible to see from the name of both functions, they are "GET" requests. This is important because some areas of the cloud allow not only to fetch data with the "GET" function but also to make changes through "POST" functions. With the type of function then you identify the type of data access you want to do.

Benefits of Flexible Cloud Architecture: The flexible and adaptable approach of the cloud architecture allows new sensor manufacturers and new clouds to be integrated with relative ease. This flexibility is a key advantage, allowing the entire system to remain aligned with evolving technology and changing industry needs.

The cloud architecture provides an essential link between the various sensor manufacturer clouds and the central server. Its ability to adapt to the various access methodologies and structures of the manufacturers' clouds is crucial to ensure accurate and timely data collection. This cloud architecture is designed to provide a dynamic and reliable solution for data acquisition in the overall architecture.

3.2.2 Database Structure

The memory of the described software architecture resides in a highly organized and modular data management system, implemented through a specified database structure. This structure, consisting of several interconnected tables, is designed to enable the efficient collection, organization, and storage of vast amounts of data from sensors distributed in various environments. This modular approach provides flexibility and scalability, allowing monitored environments to be added or modified easily.

Table of Environments: The database structure begins with a table of environments, which serves as the starting point for configuration and activation of acquisition data. In this table, each monitored environment is listed along with an indicator of whether or not data acquisition is active in that particular environment. This approach allows careful control over data acquisition, enabling acquisition to be turned on or off as needed.

Environment-Specific Tables: For each environment listed in the environments table, there is a dedicated table that collects specific details related to that particular environment. These tables contain detailed information about the sensors that are part of that environment, allowing for a highly organized structure. This separation of tables by environments makes the system highly modular and adaptable to future expansion or modification.

Individual Sensor Tables: The database contains highly detailed information in the individual sensor tables. Each sensor has its own table, which stores the complete history of its activations and collected data. These tables can include activation time stamps, data types, values, and more. The database's de-

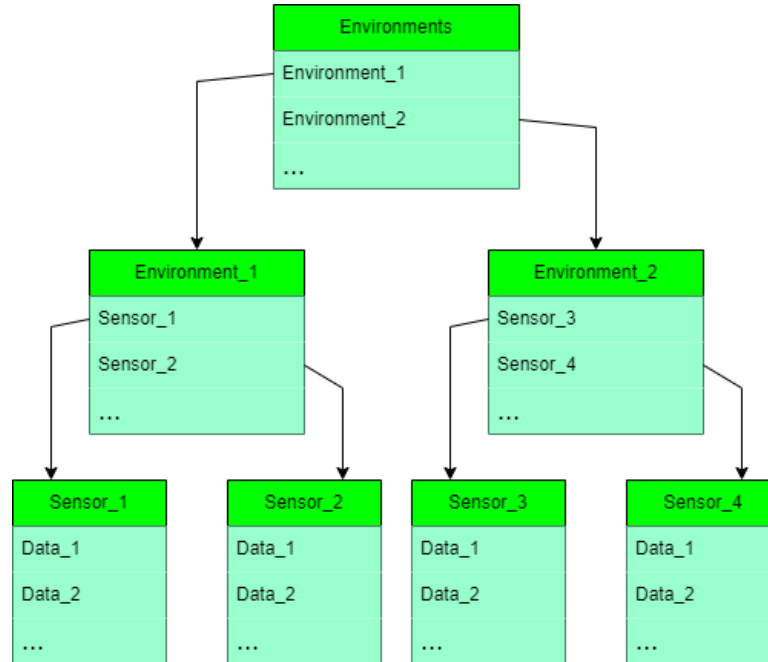


Figure 3.5: Database structure.

tailed organization in separate tables makes it easy to search for further analysis.

Advantages of Modular Structure: The modular approach to database structure offers several advantages. First, it allows simplified and focused data management for each environment and sensor. In addition, this structure facilitates the management of changes and expansions. If new environments are added, simply create new specific tables. Similarly, the addition of new sensors involves the creation of new entries in the corresponding tables.

The database structure plays a central role in maintaining consistency, organization, and accessibility of sensor data in the architecture. Its modular design allows for seamless adaptation and expansion while ensuring the data remains efficient and comprehensible. By categorizing tables based on environment and sensor, a robust foundation is established for monitoring, analyzing, and leveraging the collected data.

3.3 Reasoning Server

The Reasoning Server is an intelligent computational platform that processes data acquired from sensors to extract meaning and value. It is equipped with analytical capabilities, such as machine learning algorithms and large language models, which enable it to recognize hidden patterns, trends, and connections within the data. This server is the link between the vast amount of data collected and the ability to provide relevant and useful information to end users.

The role of the Reasoning Server in converting data processing into refined knowledge is crucial for analysis and reasoning. It is a critical component in transforming raw data into valuable insights by gathering, organizing, and utilizing it to create predictive models, interpreting human behaviors, and drawing significant conclusions to facilitate continuous improvement.

The Reasoning Server has two different features that can be accessed depending on which server route is taken. It can be run on either the network or locally at a specific IP address and port number. These two elements constitute the BASE URL. The Reasoning Server is located at the following BASE address.

Listing 3.5: BASE URL

```
1 BASE = "http://localhost:5000/"
```

This address corresponds to the main functionality of the Reasoning Server, the real reasoning that makes the predictions for the seven chosen activities.

For the ML approach, for example, here are the individual functions that allow obtaining the predictions. In order we have predictions for the activities "Somebody at the desk", "Somebody outside the office", "Watching the computer screen", "Somebody moving", and "Ongoing meeting".

Listing 3.6: BASE URL service

```
1 y_pred_pos = pos_prediction()
2 y_pred_office = office_prediction()
3 y_pred_gaze = gaze_prediction()
4 y_pred_path = path_prediction()
5 y_pred_meeting = meeting_prediction()
```

Instead for the LLM approaches there will be the prompt and the sequence of queries that will be presented later.

The Reasoning Server also has a second feature that is identified by the following URL

Listing 3.7: BASE+/robot URL

```
1 URL = BASE + "/robot"
```

When the Reasoning Client connects to this URL, it can access the latest predictions directly from the Database. The addition to the BASE URL is named "robot" to allow for the possibility of one or more robots that can gather prediction information and potentially take action based on it.

Listing 3.8: BASE+/robot URL service

```
1 con = sqlite3.connect(DB)
2 con.row_factory = dict_factory
3 curs = con.cursor()
4 query = 'SELECT * FROM Predictions;'
5 data = curs.execute(query).fetchall()
6 lenght = len(data)
7 last_data = data[lenght - 1:]
8 return jsonify(last_data)
```

To provide this service, the Reasoning Server connects to the Database and retrieves data from the Prediction table. This table contains all the predictions made in chronological order. After that, the Reasoning Server extracts the latest prediction for all activities and sends it to the Reasoning Client.

Within the Reasoning server, two types of reasoning were considered to maximize the analysis of collected data:

1. Machine Learning Algorithm (ML).
2. Large Language Models (LLM).

3.3.1 Activity Detection with ML

Integrating machine learning algorithms into the Reasoning Server is an important step in maximizing the value of sensor data. By utilizing this technique, hidden patterns, trends, and connections can be detected, leading to a deeper understanding of ongoing activities and events in the monitored environment.

This approach is useful in a study environment that includes a study room, office, corridor, and bathroom, where the analysis can focus on significant activities and events occurring in these locations.

Definition of Activities and Events: The initial phase of this process required a thorough analysis to establish the activities and events that needed to be recognized and monitored. This involved defining events such as the presence of people at different desks, the presence of people outside the office, the identification of meetings taking place in designated areas, and the understanding of activities being performed based on people’s eye and head movements. Regarding movement paths within the environment, it was a key priority to define which rooms were visitable and which paths were possible. This analysis of the physical layout of the environment made it possible to accurately determine the different possible routes and transit areas, enabling accurate modeling of possible routes.

Creation of Detailed Datasets: To feed the machine learning algorithms, a large dataset had to be created. For each activity and route considered, detailed information was collected from sensors. This dataset captured a variety of possible situations and scenarios, allowing the algorithms to learn from a wide range of contexts. It is important to underline that the datasets used in the learning phase are stored separately in the "test.db" file and not in the "sensors.db" file which stores the real-time data. The "test.db" file is accessed in the same way as the "sensors.db" file.

Choice of Algorithms Random Forest and SVM: To address the different prediction and classification challenges, two types of machine learning algorithms were selected: Random Forest and Support Vector Machine (SVM). The use of these algorithms allowed addressing both situations where binary prediction was required (Random Forest) and classification situations where specific classes had to be identified (SVM). Through learning from the dataset, machine learning algorithms extracted patterns and relevant information from historical data. They learned to recognize connections between inputs (sensor data) and outputs (relevant activities and events), enabling advanced predictive and classificatory capabilities.

We briefly remind that ML aims to find a function $f_{\Theta} : X \rightarrow Y$, where X and Y are respectively the input and the output space, and Θ is the set of parameters of the function f such that the function well approximates $P(Y|X)$. Since the real distribution of $X \times Y$ is not known, the function f_{Θ} must be learned using information about the input and the output space X and Y derived from the sampling of these spaces, which is called the dataset $D_n = \{(X_1, Y_1), \dots, (X_n, Y_n)\}$, where

n is the number of samples. Model Selection consists of splitting the dataset into three parts: the Learning Set L , the Validation Set V , and the Test Set T . For each configuration of its hyperparameters, the function f , with its parameters, is learned through L and then validated on V . The model is represented by the configuration of parameters and hyperparameters that give the best results, according to a loss function. In the end, the model can be tested on T : the model is repeatedly fed a window of data of the designated size, taken individually from the T set, drawing conclusions about the model's accuracy. The data are divided in 80% for learning and validation, 20% for testing. This procedure needs to be carried out for each of the activities considered.

In our particular scenario utilizing the SVC algorithm (a variant of SVM employed to detect the "Somebody moving" activity), grids were used thanks to which is possible to determine which parameters need to be optimized. In this case, the parameter "C" is being optimized by testing various values within a set range on the grid.

The parameter "C" in the Support Vector Machine (SVM) algorithm is a hyperparameter that controls the trade-off between maximizing the width of the decision "margin" and minimizing the classification error on the training set. In other words, it affects the ability of the SVM model to balance the correctness of classification on the training data and its ability to generalize well to new data.

Low values of C: When C is small, the SVM model is more tolerant to errors on the training set and tries to maximize the width of the decision margin, even at the cost of misclassifying some training points. This leads to SVM models with greater generalization capacity, but can sometimes result in wider margins and more conservative decisions.

High values of C: When C is large, the SVM model tries to minimize the classification error on the training set, which means it will try to correctly classify as many training points as possible. This can lead to SVM models with tighter margins and could lead to a higher risk of overfitting, i.e., the model may overfit the training data and have a worse generalization ability.

When conducting cross-validation, it is important to take into account the number of dataset portions being analyzed. This factor influences both the accuracy of model performance estimates and the speed of the process of finding the best parameters. The chosen value determines how many times the function f with the parameter set Θ will be evaluated on different portions of the database. Opting for a higher value can result in more reliable model performance estimates,

but it also means that the model will need to be trained and assessed multiple times, which requires more computational time.

Modularity of the code: Considering that each activity has a different learning phase, the learning process in the code is made modular. There is a function for each activity that returns the model obtained from the learning process. This allows specific changes without affecting all activities and enables new activities to be added or removed based on the environment in a simpler way.

In the same way, when the Reasoning Client makes a request, the prediction phase begins and for each activity there is a specific function that accesses the database, retrieves the necessary data for the specific activity, and provides it the corresponding model.

Benefits of the Machine Learning-Based Approach: The machine learning algorithm-based approach has made it possible to overcome the limitations of traditional analysis and capture small and complex details in the collected data. These algorithms are able to adapt to changes and improve their performance as they gain experience on new data, thus contributing to increasing accuracy in task analysis.

In order to extract insights from sensor data, the utilization of machine learning algorithms is essential. Our Reasoning Server use powerful algorithms such as Random Forest and SVM to identify patterns and generate comprehensive datasets. This enables careful analysis and interpretation of activities within the observed environment, delivering valuable information for informed decision-making and ongoing optimization. Ultimately, this technique increases the level of understanding derived from the data.

3.3.2 Activity Detection with LLM

Reasoning based on large language models, such as using ChatGPT 3.5 Turbo, in the context of the Data Server represents an important step in data analysis and context interpretation. This approach relies on the use of high-powered language models to understand and interpret free text, enabling deep analysis of qualitative data, written interactions, and associated metadata. In the context of an office environment with labs, offices, hallways, and bathrooms, the use of ChatGPT 3.5 Turbo offers a unique opportunity to extract knowledge from textual data collected by sensors.

Role of Large Language Models: Large language models, such as ChatGPT 3.5 Turbo, are trained on a large amount of text and possess the ability to generate consistent and relevant responses to a variety of questions and topics. These models are able to understand context, recognize entities and relationships, and generate texts that reflect deep knowledge. In the context of the architecture described, the use of a large language model opens up new possibilities for understanding human interactions, interpreting textual data, and drawing meaningful insights.

Question and Answer Modulation: The ChatGPT 3.5 Turbo API allows questions and answers to be modulated according to the specific needs of the analysis. Users can flexibly formulate questions, asking the model to extract specific information from the collected textual data. This approach offers considerable flexibility in guiding the model toward extracting relevant knowledge, allowing customization of the analysis.

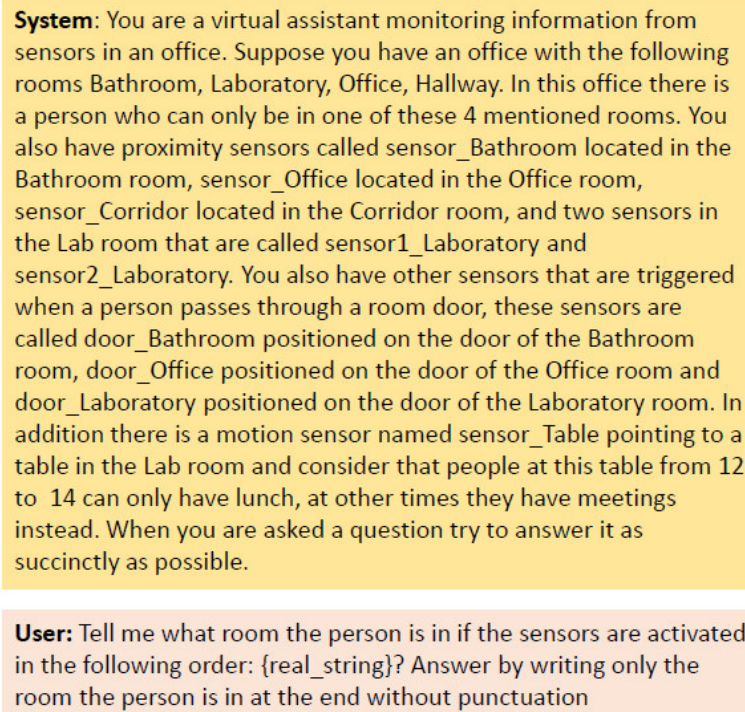
Textually Interpreted Activities and Events: With the power of large language models, it is possible to interpret free text associated with different activities and events. For example, descriptions of meeting or meeting activities can be analyzed, identifying participants, topics discussed, and outcomes achieved. In addition, data collected by sensors on people's gaze and head can be correlated with texts to better understand the activities performed and the attention paid to certain topics.

Structuring of Text Data: Large language models allow textual data to be structured in a consistent and organized manner. For example, detailed accounts of events, activities, and human interactions can be created by summarizing data into understandable and useful formats for end users.

Integration with Quantitative Analysis: The qualitative analysis provided by large language models integrates with quantitative analysis based on machine learning algorithms. While machine learning algorithms recognize patterns and relationships in numerical data, large language models allow textual data to be interpreted and contextualized, thus enriching the overall analysis.

Advantages of the Large Language Models Approach: Large language models can enhance the analysis of textual data, providing a more comprehensive understanding of activities and events taking place in a monitored environment. This approach includes human interpretation, which can capture shades and details that may not be detected by machine learning algorithms alone.

The approach we implement using LLMs retains the entire architecture described so far by simply replacing the ML-based Reasoning Server with another LLM-based Reasoning Server. The sensors used for each activity, and the data windows used are the same as the ML approach summarized in Table 4.1. However, since LLMs introduce a different level of interaction based on natural language, there is a need for preprocessing the acquired samples so that we know what values to include in the prompt that will be submitted to the language model.



System: You are a virtual assistant monitoring information from sensors in an office. Suppose you have an office with the following rooms Bathroom, Laboratory, Office, Hallway. In this office there is a person who can only be in one of these 4 mentioned rooms. You also have proximity sensors called sensor_Bathroom located in the Bathroom room, sensor_Office located in the Office room, sensor_Corridor located in the Corridor room, and two sensors in the Lab room that are called sensor1_Laboratory and sensor2_Laboratory. You also have other sensors that are triggered when a person passes through a room door, these sensors are called door_Bathroom positioned on the door of the Bathroom room, door_Office positioned on the door of the Office room and door_Laboratory positioned on the door of the Laboratory room. In addition there is a motion sensor named sensor_Table pointing to a table in the Lab room and consider that people at this table from 12 to 14 can only have lunch, at other times they have meetings instead. When you are asked a question try to answer it as succinctly as possible.

User: Tell me what room the person is in if the sensors are activated in the following order: {real_string}? Answer by writing only the room the person is in at the end without punctuation

Figure 3.6: Structure of the prompt: the system field is constant for all activities, while the user field varies based on the specific activities and sensor readings.

In this study, we used ChatGPT 3.5 Turbo as a 'reasoner,' capable of answering questions based on natural language understanding. To obtain answers consistent with our requirements, it is necessary to construct the prompt provided to the OpenAI API interface as accurately as possible. The prompt consists of three fields: 'system,' 'user,' and 'assistant.' The 'system' field is highly important as it provides guidelines for the model's responses and, most importantly, furnishes a description of the environment upon which it must reason. The 'user' and 'assistant' fields are used for the user's question and the model's answer in

the previous iteration, respectively. While the 'system' field remains consistent throughout the process, the user's questions vary depending on the task to be analyzed, resulting in varying model responses as required.

In a preliminary phase, which could ideally correspond to validation in ML, the responses provided by ChatGPT 3.5 Turbo were manually analyzed to assess their consistency. When the answers were consistent, the prompt wording was retained, and the prompt was considered appropriate for that activity and the corresponding sensors. If the answers were not consistent, we returned to the previous step to modify the wording of the prompt submitted to the LLM. The final structure of the prompt can be observed in Figure 3.6.

Please note that the variable part {real_string} should be filled with a sequence of sensor readings, automatically composed from actual sensor data. For example, if we set {real_string} to "sensor_bathroom, bathroom_door, sensor_corridor, office_door, sensor_office," the response would simply be "office." If we submitted a different user prompt, such as "Tell me the path of the person if the sensors are activated in the following order: sensor_bathroom, bathroom_door, sensor_corridor, office_door," we can expect a more detailed reply, like this: "The person's path is as follows: sensor_Bathroom (in the Bathroom); door_Bathroom (leaving the Bathroom); sensor_Corridor (in the Hallway); door_Office (entering the Office)."

In conclusion, reasoning based on large language models is a good solution for the described architecture. Through the use of ChatGPT 3.5 Turbo and question-and-answer modulation with the prompt, it is possible to interpret textual data, extract meaningful knowledge, and enrich the overall analysis. This approach provides a deep perspective on the monitored environment, allowing meaningful insights to be captured from human interactions and qualitative data.

Chapter 4

Experiments

4.1 Methodology

Experiments were designed to capture a wide range of activities and interactions in a university environment. The methodology involved strategically placing sensors to capture meaningful details, creating representative datasets, and conducting data acquisition and analysis.

4.1.1 Strategic Arrangement of Sensors

We consider a university environment with an office, a study room (featuring desks where students study or have meetings), and a bathroom, as shown in Figure 4.1.

- Two PIR sensors were placed to cover three study desks in the study room. Specifically, PIR1 monitored desks 1 and 3, while PIR2 covered desks 1 and 2.
- PIR3 sensor was placed to detect any movement around a meeting table, allowing for the detection of people’s activities and meetings.
- PIR4 sensor was placed inside the corridor, in the direction of the office door, to detect the passage of people through that area.
- PIR5 sensor was installed inside the office itself.
- PIR6 sensor was placed inside the bathroom.
- Three magnetic door sensors were placed on the study room door (magnetic sensor1), office door (magnetic sensor2), and bathroom door (magnetic sensor3), providing information on door opening and closing.

- A robot with onboard cameras is positioned in strategic positions depending on the activities to be recognized. In this experiment, the robot does not move autonomously to keep the experiment simpler.

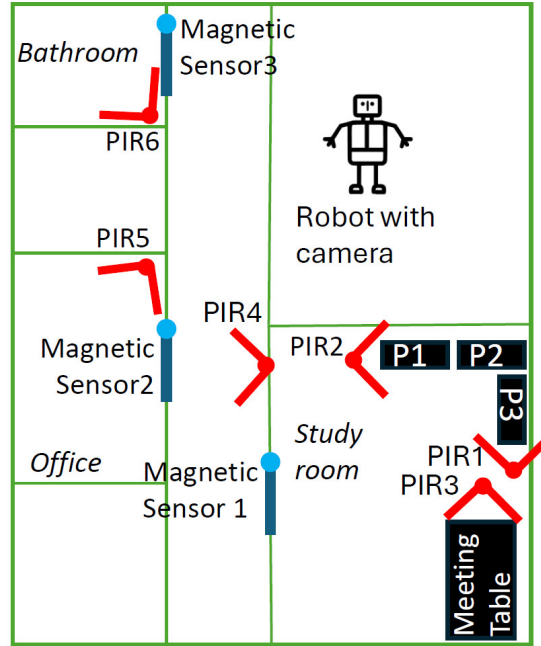


Figure 4.1: Smart Environment considered for experiments. In the image, the red dots represent the PIR sensors with their visibility range. Blue dots represent the magnetic door sensors. The robot is manually placed in different positions. In black we see the desks and the meeting table.

4.1.2 Activities and Creation of Datasets

The following activities were considered, necessitating the creation of representative datasets for training ML algorithms and testing both ML and LLM algorithms.

- Somebody at study desks. Data were collected for empty desks, individually occupied desks, and different combinations of occupancy.
- Ongoing meeting. Data were recorded during simulated meetings as well as during lunch breaks to distinguish between meetings and free time.
- Somebody outside the office. Data were acquired during the quick passage of people and their static presence, such as while waiting for somebody.

- Somebody moving. Data were acquired about the sequence of sensor activations while moving between rooms. This activity is mainly related to localization in the environment and the ability to know topologically where a person is. Since the localization that is done is topological, in Figures ?? it is possible to see the topological map of the environment represented in Figures 4.1.
- Watching the computer screen. The robot’s camera was used to observe students sitting at desks in the study room, both while they were studying and during other activities, with the ultimate goal of encouraging students to work if they were spending time on social networks or their smartphones.

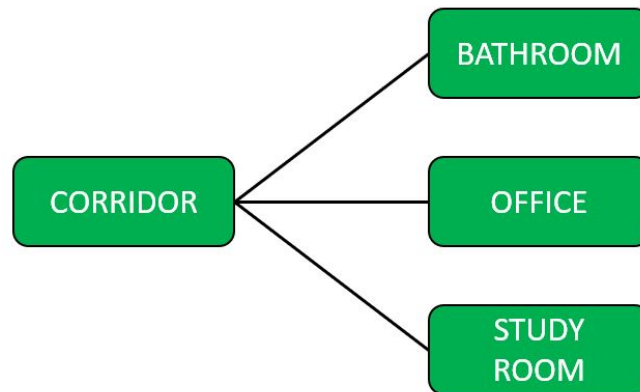


Figure 4.2: Environment Topological map

Data collection was conducted to capture realistic and representative situations of the selected activities within the office environment. Data were recorded and documented during working hours, with every aspect accurately annotated by a human observer for proper interpretation.

Table 4.1 presents the differences just mentioned between the various activities, including: the classification output (binary/not-binary), the algorithms used for classification, the sensors used, the size of the dataset computed as number of data windows \times data window (i.e., where the data window determines how many replicas of sensor data in the recent history are considered for classification).

The acquired data were subsequently utilized to train, validate, and test the ML algorithms and test LLMs (which do not require training).

Table 4.1: Activity, Sensors and Data

Activity, outputs, and algorithms	<i>Sensor and size</i>	<i>Dataset</i>
Somebody at study desk 1 {T,F} Random Forest and GPT3.5	PIR1, PIR2	1500×15
Somebody at study desk 2 {T,F} Random Forest and GPT3.5	PIR1, PIR2	1500×15
Somebody at study desk 3 {T,F} Random Forest and GPT3.5	PIR1, PIR2	1500×15
Somebody outside the office {T,F} Random Forest and GPT3.5	PIR4	500×15
Watching the computer screen {T,F} Random Forest	Robot’s camera	10700× 1
Somebody moving { $p_1, p_2, p_3, p_4, p_5, p_6$ } SVM and GPT3.5	PIR1 PIR2 PIR3 PIR4 PIR5 PIR6 magnetic sensor1 magnetic sensor2 magnetic sensor3	105×40
Ongoing meeting {T,F} Random Forest and GPT3.5	PIR3	820 × 15

4.1.3 Database Access

Access to the database is critical to the proper functioning of the architecture, a wrong way of accessing it would first compromise the veracity of the data in it and then the quality of the predictions obtained from the two approaches used. The database is managed through SQLite, which means having a file that in the specific case is "sensors.db." To access this file, one must connect to it and use specific queries to access the database to insert data or retrieve it.

Following the flow of data in Figure 3.4 the first to access the Database is the Data Server, which writes data into it. The storage of the data in the database is done using specific commands inserted into the queries handled by SQLite:

Listing 4.1: SQLite formalism to insert data in the database

```

1  conn = sqlite3.connect(DB)
2  sql = f"INSERT INTO {tab}(time,data) VALUES(?,?)"
3  insert_data = (obj, data)
4  cur = conn.cursor()
5  cur.execute(sql, insert_data)

```

In this code block, there are three variables being used. Firstly, "conn" is a variable that represents the database connection. Secondly, "cur" is a cursor that is used to navigate through the database and retrieve specific data. Finally, "sql" is a query that specifies the table and columns to be used for inserting data. The INSERT INTO command is used to insert data into the specified table and columns. The data to be inserted is represented using the VALUE command and two question marks. These question marks are later replaced with actual data using the "insert_data" variable. The "insert_data" variable contains the current time and status of the sensor. It is important to note that the number of values in "insert_data" should match the number of question marks in the query. Finally, the "cur.execute" command is used to execute the query with the data from the "insert_data" variable.

The Reasoning Server is responsible for accessing the database to retrieve data. It uses a similar structure to that used by the Data Server to write data to the database. However, the query changes to retrieve the data from the database.

Listing 4.2: Reasoning Server database access

```

1  "SELECT time,data FROM tab"

```

In this case, we are not using the "INSERT INTO" and "VALUES" commands. Instead, we are using the "SELECT" and "FROM" commands to indicate the columns and table from which we need to retrieve the required data.

4.1.4 ML

In the preceding chapter, we discussed the essential stages of learning and validation for ML algorithms. They allow us to identify the optimal set of parameters Θ for the function f , which can effectively approximate $P(Y|X)$. In particular, grids were also discussed for the optimization of parameters such as the "C" parameter within the SVC algorithm. The grid used and the process for arriving at the final model is presented below.

Listing 4.3: ewelinkAPI device status request

```

1  grid = {'C': np.logspace(-4, 3, 5),
2         'kernel': ['linear']}
3  MS = GridSearchCV(estimator=SVC(),
4                   param_grid=grid,
5                   scoring='balanced_accuracy',
6                   cv=10,
7                   verbose=3)
8  H = MS.fit(X_train, y_train)
9  M = SVC(C=H.best_params_['C'],
10         kernel=H.best_params_['kernel'])

```

It is possible to see that in the "grid" variable there is the parameter "C" and is also indicated the range of values to test for it. The "GridSearchCV" function is then used to perform cross-validation and determine the best values within the grid.

Below in Table 4.2 are summarized the values of "C" tested and the respective score values obtained considering "balanced accuracy" as the scoring parameter.

Table 4.2: C parameter optimization

C parameter	Score
0.0001	0.143-0.286
0.005623413251903491	0.857-1
0.31622776601683794	1
17.78279410038923	1
1000.0	1

As can be seen in Table 4.2 there are three values that have maximum scores and present the same score for all the dataset sections used in the cross-validation, this gives consistency to the model with those specific parameters. As can be seen from the code previously presented, the final model is created by then going to choose those that are considered the best parameters, in this case, the model is created with the value of $C=0.31622776601683794$ being the lowest value of C that presents the maximum score.

The previous chapter also discussed the importance of the number of dataset portions analyzed during cross-validation. This information is denoted by the "cv" parameter in the "GridSearchCV" function. In the current scenario, the value of "cv" is set to 10, indicating that each C value is tested on 10 different dataset portions for cross-validation. The integer 10 is a common value for the parameter "cv" but it can be set differently depending on the context and available resources.

4.1.5 LLM

In the previous chapter, the use of LLMs in Reasoning Server was discussed, with particular emphasis on the importance of the prompt and how it should be created in order to optimize performance. In the creation of the prompt also the sequence of questions that are asked by the user is very important, in fact whenever a question is asked, it is added to the prompt along with the given answer, the assistant will go and give an answer also contextualized by the questions that he/she has previously answered. This is a very important quality for a model created specifically for dialogue, but in our specific case it can create some problems since, during the various tests, it was noticed that taking into account also the previously given answers, the most recent answers were not completely objective but even created confusion by giving wrong answers. To avoid this, the "pop" function was used to remove the last interaction between user and assistant from the prompt. This avoids that the assistant's answer from becoming conditioned by previous questions and thus allows for a more objective answer for that specific question.

In the previous chapter, an example was provided of a user's query and the assistant's response for the activity "Somebody moving." The field "system" used was also shown in Figure 3.6. Now, will be presented queries and their expected responses for the other activity.

Starting with the activity "Somebody at desks" we have the following expected question and answer which will be the same for all is three desks considered, obviously going to interchange P1, P2 and P3 which refer to desk_, desk.2 and desk.3 respectively.

Listing 4.4: "Somebody at study desks" user question and response

```
1 sensor1_Laboratory is {real_string1},↵  
   sensor2_Laboratory is {real_string2}.P3 is busy?
```

2 Yes , P3 **is** busy ./No , P3 **is not** busy .

When someone is at the desk, the first response is given, the second response is given when no one is present.

The prompt for the "someone outside the office" activity is structured in the following manner. In this case it can be seen that it was asked to answer only with yes or no. This was done to prevent the assistant from giving an unnecessarily long answer.

Listing 4.5: "Somebody outside the office" user question and response

```
1        Can you tell me if there are people waiting in the ←
           corridor knowing that the sensor called ←
           sensor_Corridor is {real_string}? answer yes or no.
2        yes/no
```

The prompt for the "Watching the computer screen" activity is now presented. This activity is exclusive to LLMs since it cannot evaluate the values obtained from analyzing camera images, and its response is always Don't know. The variable part {real_string} is filled with the values of the head yaw, head pitch, head roll, gaze yaw and gaze pitch.

Listing 4.6: "Watching the computer screen" user question and response

```
1        Consider that usually if a person is looking more or ←
           less in front of himself is watching the computer ←
           screen, if not he is doing something else, would ←
           you be able to tell me whether a person who is in ←
           the study room is watching the computer screen or ←
           doing something else considering you have the data ←
           of head yaw, head pitch, head roll, gaze yaw and ←
           gaze pitch which respectively are: {real_string}. ←
           Answer me with Watching, Do not Watching or Do not ←
           know.
2        Do not know
```

The last activity is an "Ongoing meeting" which takes into account both the value of the PIR3 sensor and the current time. Initially, a single question was created with both values in it, but this often caused confusion in the response of the assistant. To make it easier to understand, the user is asked two simpler questions instead.

Listing 4.7: "Watching the computer screen" user question and response

```

1   if the sensor sensor_Table is {real_string} tell me if↔
      there are people at the table.
2   yes/no
3   if there are people at the table and they are the {now↔
      } tell me if there is an ongoing meeting?
4   yes/no

```

It's important to note that the second question is dependent on the first one. If the answer to the first question is yes, then the second will be asked. However, if the answer is no, the second question is unnecessary and won't be asked.

4.1.6 Online Data Acquisition and Analysis

After completing the offline testing phase of the algorithm, we proceeded to an 'online' simulation of its operation. During this phase, the trained models were applied to data samples taken from the datasets, and the entire process was executed as if it were in real-time, mirroring the actual operation of the architecture illustrated in Figure 3.3 and simulating the office environment shown in Figure 3.4. This simulation allowed us to evaluate how the models would perform in a scenario closely resembling real-world conditions. We expect similar results, with possible differences due to random components in LLM responses.

Finally, we conducted real-time testing with new data (10 data windows for each activity), acquired and processed in real-time from people interacting with the Smart Environment. We evaluated both accuracy and execution times.

4.2 Results

This section presents the results obtained from the experiments conducted and mentioned in the previous section.

4.2.1 ML Accuracy

As shown in Table 4.3, the accuracy values of ML algorithms consistently reach high levels, which is expected given the simple scenario we are considering. Notably, the lowest accuracy is associated with the occupation of study desk number 1, yielding an accuracy of 0.76, indicating that 76 percent of the samples were correctly recognized. The results also include instances with a maximum accuracy of 1.0, particularly in detecting somebody in front of the office and recognizing

people moving along a path. It is important to note that as expected, the data from the online simulation of the whole architecture are identical to offline testing since the same data were used.

Table 4.3: ML Accuracy

Activity	<i>Offline testing</i>	<i>Online simulation</i>
Somebody at study desk 1	0.76	0.76
Somebody at study desk 2	0.89	0.89
Somebody at study desk 3	0.95	0.95
Somebody outside the office	1	1
Watching the computer screen	0.99	0.99
Somebody moving	1	1
Ongoing meeting	0.87	0.87

In order to evaluate the effectiveness of our predictions, have been carefully created seven confusion matrices, one for each activity. These matrices provide an in-depth overview of the precision of our predictions, highlighting the true positives, false positives, true negatives, and false negatives. Through a comprehensive analysis of these matrices, we were able to pinpoint the sensitivity, specificity, and accuracy of our model for each activity. Armed with these insights, we were able to identify opportunities for improving our model. In the Tables 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 4.10 are presented the confusion matrices for ML approach.

Table 4.4: Study desk 1 confusion matrix

Somebody at study desk 1	<i>Positive</i>	<i>Negative</i>
Positive	133	21
Negative	51	95

It is also important to take into account the averages of overall learning time and learning times for each activity. In Table 4.11 is possible to see all the learning times. These times depend on the dimension of the respective datasets, so by the number of processed data, and depend also on the type of data. As can be seen, the learning time of the "Watching the computer screen" activity is the highest because the input data are floats, unlike the other tasks that have binary data as input.

Table 4.5: Study desk 2 confusion matrix

Somebody at study desk 2	<i>Positive</i>	<i>Negative</i>
Positive	136	18
Negative	14	132

Table 4.6: Study desk 3 confusion matrix

Somebody at study desk 3	<i>Positive</i>	<i>Negative</i>
Positive	145	9
Negative	5	141

Table 4.7: Somebody outside the office confusion matrix

Somebody outside the office	<i>Positive</i>	<i>Negative</i>
Positive	53	0
Negative	0	48

Table 4.8: Watching the computer screen confusion matrix

Watching the computer screen	<i>Positive</i>	<i>Negative</i>
Positive	1083	3
Negative	4	1055

Table 4.9: Somebody moving confusion matrix

Somebody moving	<i>Path1</i>	<i>Path2</i>	<i>Path3</i>	<i>Path4</i>	<i>Path5</i>	<i>Path6</i>	<i>No Path</i>
Path1	4	0	0	0	0	0	0
Path2	0	4	0	0	0	0	0
Path3	0	0	2	0	0	0	0
Path4	0	0	0	3	0	0	0
Path5	0	0	0	0	3	0	0
Path6	0	0	0	0	0	2	0
No Path	0	0	0	0	0	0	3

Table 4.10: Ongoing meeting confusion matrix

Ongoing meeting	<i>Positive</i>	<i>Negative</i>
Positive	66	16
Negative	4	78

Table 4.11: ML: Learning time evaluation in seconds

Activity	<i>ML learning times</i>
Somebody at study desks	6.024517
Somebody outside the office	1.213123
Watching the computer screen	32.895582
Somebody moving	0.223945
Ongoing meeting	1.973264
Total learning time	42.331436

4.2.2 LLM Accuracy

Please keep in mind that, for sensor data processing, the Reasoning Server retrieves N data samples from pertinent sensors (the data window size is specified in Table 4.1) and generates a vector that represents the sequence of sensor activations. This sequence is then used to construct a string that populates the prompt to be sent to the Large Language Model (LLM).

The results presented in Table 4.12 are generally very good. However, it is worth noting that the model is not suitable for all activities: for example, it is unable to provide outcomes for eye and head orientation data, which are used for detecting somebody watching the computer screen. This limitation arises from the difficulty of expressing eye and head direction in natural language, highlighting the constraint of this approach, which is applicable only when sensor readings can be easily described in natural language. For the other predictions, we once again observe that for some of them, we achieve 1.0 accuracy due to the simplicity of the scenario. It is important to emphasize that, unlike the previous case, the data collected during the online phase are not identical to those obtained in the offline phase. This discrepancy arises due to the fact that LLMs exhibit behavior that is not always consistent, thereby introducing unpredictable variations in classification outputs.

Table 4.12: LLM Accuracy

Activity	<i>Offline testing</i>	<i>Online simulation</i>
Somebody at study desk 1	1	1
Somebody at study desk 2	0.97	0.97
Somebody at study desk 3	0.77	0.76
Somebody outside the office	0.98	0.99
Watching the computer screen	n.a.	n.a.
Somebody moving	1	1
Ongoing meeting	0.89	0.78

Confusion matrices are also presented for the LLM approach as for the ML approach in the Tables 4.13, 4.14, 4.15, 4.16, 4.17, 4.18. However, in this case the matrix related to the "Watching the computer screen" activity is missing as the model is unable to provide either a positive or negative answer.

Table 4.13: Study desk 1 confusion matrix

Somebody at study desk 1	<i>Positive</i>	<i>Negative</i>
Positive	49	0
Negative	0	34

Table 4.14: Study desk 2 confusion matrix

Somebody at study desk 2	<i>Positive</i>	<i>Negative</i>
Positive	192	11
Negative	0	289

4.2.3 Real-time test

Table 4.19 presents the results obtained during the real-time behavior testing of the architecture for both classification approaches. As observed in 4.19, the accuracy achieved in these tests is occasionally lower. However, it is essential to note that the real-time testing dataset contained a significantly smaller number of samples, so these results should be interpreted with caution. During real-time

Table 4.15: Study desk 3 confusion matrix

Somebody at study desk 3	<i>Positive</i>	<i>Negative</i>
Positive	14	42
Negative	0	128

Table 4.16: Somebody outside the office confusion matrix

Somebody outside the office	<i>Positive</i>	<i>Negative</i>
Positive	248	3
Negative	3	248

Table 4.17: Somebody moving confusion matrix

Somebody moving	<i>Path1</i>	<i>Path2</i>	<i>Path3</i>	<i>Path4</i>	<i>Path5</i>	<i>Path6</i>	<i>No Path</i>
Path1	4	0	0	0	0	0	0
Path2	0	4	0	0	0	0	0
Path3	0	0	2	0	0	0	0
Path4	0	0	0	3	0	0	0
Path5	0	0	0	0	3	0	0
Path6	0	0	0	0	0	2	0
No Path	0	0	0	0	0	0	3

Table 4.18: Ongoing meeting confusion matrix

Ongoing meeting	<i>Positive</i>	<i>Negative</i>
Positive	44	12
Negative	0	57

tests, it is important to take into consideration the temporal shifting window of data for each activity. This window adjusts to include the last N data for each sensor in the database. The value of N and the size of each window can be found in Table 4.1. By considering the second value in the "Dataset" column for the respective tasks, can be obtained the appropriate values. If the value of N is greater, the delay before observing the correct prediction increases. This is because the N -size window needs to be filled with data related to the event before it can be recognized accurately.

Major differences emerge in terms of execution time: ML algorithms take approximately 0.3 sec on average on a Intel(R) Core(TM) i7-7700HQ CPU@2.80 GHz to recognize all 7 activities, while LLMs, owing to GPT 3.5 response time, require approximately 13 sec for all 7 activities. In addition, in Table 4.21 it is possible to observe individual task execution times.

Table 4.19: ML vs LLM: Real-time evaluation

Activity	<i>ML</i>	<i>LLM</i>
Somebody at study desk 1	0.70	1
Somebody at study desk 2	0.80	0.90
Somebody at study desk 3	0.90	0.70
Somebody outside the office	1	0.90
Watching the computer screen	0.90	n.a.
Somebody moving	0.90	0.95
Ongoing meeting	0.80	0.75

Table 4.20: ML vs LLM: Execution time evaluation in seconds

Activity	<i>ML</i>	<i>LLM</i>
Somebody at study desks	0.180	6.535
Somebody outside the office	0.050	1.850
Watching the computer screen	0.064	0.680
Somebody moving	0.006	2.057
Ongoing meeting	0.002	2.109
Total execution time	0.302	13.231

4.2.4 Communication times

In analyzing the performance of this architecture, it is important to give proper emphasis to the timing of communication between the various components and to understand how often certain routines are executed.

Based on the Sequence diagram shown in 3.4, the Data Server obtains new data from the Manufacturer Clouds and updates the database every second. On average, it takes 0.061724 seconds for the Data Server to communicate with the Manufacturer Clouds. When writing data to the database, the Data Server takes an average of 0.011295 seconds to complete the process. The Reasoning Server reads and processes the data with a cadence of approximately 2.446922 seconds for the ML approach and 15.417970 seconds for the LLM approach. In addition, the Reasoning Server takes approximately 2.043962 seconds to respond to the Reasoning Client connecting to the "BASE+/robot" URL. Additionally, the Reasoning Server takes an average of 0.00804 seconds to read from the Database and 0.015392 seconds to write to it. Comparing the write times of the Data Server and Reasoning Server, it can be seen that they are very similar, and the read time of the Reasoning Server is quite low. It can be observed that the Reasoning Client is quicker when connected to the route "/robot" compared to the basic route "/", due to the latter having to include the time taken for reasoning about the data. As indicated in Table 4.21, on average, the reasoning time for ML and LLM is about 0.302 and 13.231 respectively. On the other hand, for the former, there are no execution times required, only the time taken to retrieve the data from the Database and send it to the Reasoning Client. It is notable that when the reasoning times of both approaches are added to the time taken to receive the data at the "/robot" route, the total time periods are similar to the periods calculated for the two approaches mentioned above.

Table 4.21: ML vs LLM: Execution time evaluation in seconds

Communication	<i>Time (s)</i>
Data Server - Manufacturer Cloud	0.06172
Data Server - Database (writing)	0.01129
Reasoning Server period (ML)	2.44692
Reasoning Server period (LLM)	15.41797
Reasoning Server - Database (reading)	0.00804
Reasoning Server - Database (writing)	0.01539
Reasoning Server - Reasoning Client	2.043962

4.3 Discussion

Starting with the choice of architecture, after establishing the final structure, it was crucial to figure out what type of sensors to use and which activities to recognize. The choice was made taking into account both the difficulty in recognizing certain activities and taking into account the possibility of using a very versatile type of sensors that would allow to keep costs low but have a very good performance for acquiring data on various activities.

As explained in the previous sections, the choice fell on the use of PIR sensors, magnetic door sensors, and cameras. These sensors in particular PIR and magnetic door sensors, are cheap and above all very versatile in that the data they provide us in certain areas of use are very important and easily interpreted. The choice of activities was obviously made so that we could best evaluate the goodness of the activity using the chosen sensors. Most of the activities are based on a concept of localization in the environment such as "Somebody at the study desk", "Somebody outside the office" and "Somebody moving", activities in which, however, the presence of people in certain areas not only gives us an indication of localization in the environment, but also allows us to understand what people are doing in certain areas set up for a certain purpose. Then we have the "Ongoing meeting" activity which also makes use of a clock to keep track of the time and best interpret the possibility of the event in question. Finally, we have the "Watching the computer screen" activity for which a camera mounted on a robot is used, which by moving around a specific area is able to obtain information about the chosen activity through the students' faces.

After choosing the sensors and activities it was important to move on to choosing the environment to figure out which areas might be of interest and how the sensors could be placed. Once this was done, once the approaches and algorithms to be used to analyze the data obtained from the sensors were chosen, everything was ready to move on to the data collection phase for the machine learning algorithms and then to the testing phase.

As it was possible to observe from the tables proposed above, the results obtained are very good both from a point of view of accuracy which has very good values, especially in some activities. Upon a detailed analysis of the results, significant differences become evident: LLMs, aside from their intuitiveness as they allow expressing sensor readings and events in natural language, exhibit comparable performance to ML in activity recognition during simple scenarios, such as recognizing a meeting. However, in more complex activity recognition tasks involving head and eye gaze orientation data, LLMs fail to provide viable

solutions. Another important consideration also needs to be made about the execution times which are very different between the two approaches, is important to note that for the Somebody at study desk activity, for the three desks, we go from a time of 0.180 seconds to one of 6.535 seconds about 36 times higher, a difference that increases considering the total time which for the LLM approach is about 44 times larger than for the ML algorithm approach. For the LLMs, however, there was no need to create a dataset to create the models used as for the ML algorithms, which in the preparation phase took a lot of time because of the huge amount of data needed. In general, a parallel could be drawn between these two approaches with compiled and interpreted programming languages. For compiled programming languages there is the script compilation phase that can take some extra time, however, then you get an executable that is very fast in terms of execution time, just like the approach of using ML algorithm. For interpreted programming languages, there is no compilation phase, but each time the code is executed it is reinterpreted each time raising the execution time, just like the approach with LLMs.

Chapter 5

Conclusions

In this thesis, we proposed a cloud architecture for activity recognition and localization within a Smart Environment. Subsequently, we evaluated two methods: one based on standard ML approaches (Random Forest and SVM) and another using Large Language Models (LLMs) to process data acquired from various sensors distributed in the Smart Environment.

The results are very good for both approaches with some predictions that can almost be said to be deterministic, with an accuracy of 1, thus a 100% correct prediction. Taking into consideration the results obtained from the ML algorithms, the lower values are due to the fact that in some situations, some samples were very similar to each other despite indicating different situations, this makes it so that when the model goes to evaluate those specific samples it can have more doubts and thus make mistakes more easily. Considering instead the use of LLM as we saw from the results, there was one activity in particular for which the model failed to give any kind of response. This was probably due to not being able to associate a specific head and gaze movement with a specific activity or behavior.

In general, it can be said that both approaches have distinct advantages and disadvantages. ML algorithms are notably effective and versatile but require a substantial amount of data for training. On the other hand, Large Language Models prove to be valuable alternatives in certain specific predictions, offering high accuracy in straightforward cases, and they have the advantage of not requiring any training.

This type of architecture also, as it is designed, is very modular and is easily improved or even modified in its main modules. In fact, in order to test the architecture with the two proposed approaches, it was enough to simply replace part

of the "Reasoning Server" module without having to touch the other modules at all. The same applies to the sensors to be used which can be added or changed very easily by simply going to add the new sensors in the right database table. This modularity also makes the architecture very versatile in its applications in that it can be used in home and work environments, but also in medical settings such as hospitals and nursing homes, by adding and change the type of activities that one wants to classify. For example, one could monitor Activities of Daily Living (ADLs) and Instrumental ADLs (IADLs) to keep track of the functional status of older people. This could also be done just by using the sensors used in this thesis by placing the PIR sensors and magnetic door sensors in the correct way. This would go a long way in obtaining important information for clinicians but most importantly, it would not infringe on the privacy of people living in a given environment. Privacy discourse is very important when it comes to smart environments as often there is a risk of obtaining certain data or images that from the privacy point of view could create problems.

The proposed architecture obviously is not perfect, but it has some limitations for example the fact that it is not deterministic and not have a 100% success rate, it may be limited for some specific uses where a very pronounced deterministic factor is required. In addition, the use of cameras with regard to privacy discourse must be very limited and cannot be used for any kind of activity, while with regard to PIR sensors and magnetic door sensors, although they are very versatile, they limit the range of activities that can be carried out especially when one wants to go into the specifics of an activity. Another limitation of the architecture as it is designed is the recognition of movement by different people in the same environment. Currently, if two people were moving at the same time, the architecture would not be able to distinguish which person is making one path and which person is making another path and would also risk confusing the paths.

From these limitations, one can take cues for future work and improvements, first and foremost that of trying to make the final predictions more and more deterministic, but above all going to include other types of sensors to increase the amount of data on which it is possible to work and consequently expand more and more the number of classifiable activities. For example, the addition of new sensors could be very useful in improving the localization of multiple people in the environment, perhaps using a fingerprint system that would allow the identification of the specific person, perhaps using RFID sensors or even identifying them through Wi-Fi or Bluetooth technology. Another improvement could be improving the delay between an event and its accurate prediction is crucial. One way to achieve this is by reducing the window width for each task. However, it's important to have a balance and not reduce it too much, as this can

result in the loss of essential features that machine learning algorithms require for proper classification. In addition, another future work that could become a real added value for this architecture is to improve the use of a robot, which in the work done is mentioned only for the use of the camera for acquiring head and gaze orientation data, but in the future, it could be much more central to the dynamics of the environment and for example in the interpretation of the results obtained by the Reasoning Server. For example, robot navigation in the environment could be implemented to help people in their daily activities.

References

- ABREU, M., BARANDAS, M., LEONARDO, R. & GAMBOA, H. (2019). Detailed human activity recognition based on multiple hmm. 171–178, cited By 5. [24](#)
- AGATE, V., FERRARO, P. & GAGLIO, S. (2019). A cognitive architecture for ambient intelligence systems. vol. 2418, 52 – 58, cited by: 5. [vii](#), [6](#)
- ALONSO, R., GARCÍA, O., ZATO, C., GIL, O. & DE LA PRIETA, F. (2010). Intelligent agents and wireless sensor networks: A healthcare telemonitoring system. *Advances in Intelligent Systems and Computing*, **71**, 429–436, cited By 9. [5](#)
- BARONTI, P., BARSOCCHI, P., CHESSA, S., MAVILIA, F. & PALUMBO, F. (2018). Indoor bluetooth low energy dataset for localization, tracking, occupancy, and social interaction. *Sensors (Switzerland)*, **18**, cited By 48. [17](#)
- BAVAFA, M. & NAVIDI, N. (2010). Towards a reference middleware architecture for ambient intelligence systems. 98–102, cited By 5. [5](#)
- BIAU, G. (2012). Analysis of a random forests model. *Journal of Machine Learning Research*, **13**, 1063–1095, cited By 904. [20](#)
- BREIMAN, L. (2001). Random forests. *Machine Learning*, **45**, 5–32. [20](#)
- CAI, Y., GENOVESE, A., PIURI, V., SCOTTI, F. & SIEGEL, M. (2019). Iot-based architectures for sensing and local data processing in ambient intelligence: Research and industrial trends. vol. 2019-May, cited By 18. [5](#)
- CHE, F., AHMED, Q., LAZARIDIS, P., SUREEPHONG, P. & ALADE, T. (2023). Indoor positioning system (ips) using ultra-wide bandwidth (uwb)—for industrial internet of things (iiot). *Sensors*, **23**, cited By 1. [vii](#), [16](#), [17](#)
- CUBO, J., NIETO, A. & PIMENTEL, E. (2014). A cloud-based internet of things platform for ambient assisted living. *Sensors (Switzerland)*, **14**, 14070 – 14105, cited by: 108; All Open Access, Gold Open Access, Green Open Access. [7](#)

- FRAILE, J., BAJO, J. & CORCHADO, J. (2008). Hybrid multi-agent architecture (hoca) applied to the control and supervision of patients in their homes. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, **5290 LNAI**, 193–202, cited By 1. [5](#)
- GARCÍA-PATERNA, P., MARTÍNEZ-SALA, A. & SÁNCHEZ-AARNOUTSE, J. (2021). Empirical study of a room-level localization system based on bluetooth low energy beacons. *Sensors*, **21**, cited By 6. [vii](#), [18](#)
- GAVRILYUK, K., SANFORD, R., JAVAN, M. & SNOEK, C. (2020). Actor-transformers for group activity recognition. 836–845, cited By 98. [vii](#), [25](#)
- ILÇI, V., GÜLAL, V., ALKAN, R. & ÇİZMECI, H. (2015). Trilateration technique for wifi-based indoor localization. [vii](#), [13](#)
- JAIN, V., GUPTA, G., GUPTA, M., SHARMA, D.K. & GHOSH, U. (2023). Ambient intelligence-based multimodal human action recognition for autonomous systems. *ISA Transactions*, **132**, 94 – 108, cited by: 3; All Open Access, Bronze Open Access. [20](#)
- KARTAKIS, S., SAKKALIS, V., TOURLAKIS, P., ZACHARIOUDAKIS, G. & STEPHANIDIS, C. (2012). Enhancing health care delivery through ambient intelligence applications. *Sensors (Switzerland)*, **12**, 11435–11450, cited By 23. [6](#)
- LAI, K.C., KU, B.H. & WEN, C.Y. (2018). Using cooperative pir sensing for human indoor localization. 1–5, cited By 14. [15](#)
- MALATRAS, A., ASGARI, A. & BAUGÉ, T. (2008). Web enabled wireless sensor networks for facilities management. *IEEE Systems Journal*, **2**, 500–512, cited By 50. [6](#)
- MARIN-PERIANU, M., MERATNIA, N., HAVINGA, P., DE SOUZA, L., MÜLLER, J., SPIESS, P., HALLER, S., RIEDEL, T., DECKER, C. & STROMBERG, G. (2007). Decentralized enterprise systems: A multiplatform wireless sensor network approach. *IEEE Wireless Communications*, **14**, 57–65, cited By 61. [6](#)
- MONTANHA, A., POLIDORIO, A., DOMINGUEZ-MAYO, F. & ESCALONA, M. (2019). 2d triangulation of signals source by pole-polar geometric models. *Sensors (Switzerland)*, **19**, cited By 5. [vii](#), [14](#)

- MURAD, A. & PYUN, J.Y. (2017). Deep recurrent neural networks for human activity recognition. *Sensors (Switzerland)*, **17**, cited By 301. [22](#)
- NAWAL, Y., OUSSALAH, M., FERGANI, B. & FLEURY, A. (2023). New incremental svm algorithms for human activity recognition in smart homes. *Journal of Ambient Intelligence and Humanized Computing*, **14**, 13433–13450, cited By 3. [23](#)
- NESSA, A., ADHIKARI, B., HUSSAIN, F. & FERNANDO, X.N. (2020). A survey of machine learning for indoor positioning. *IEEE Access*, **8**, 214945 – 214965, cited by: 97; All Open Access, Gold Open Access, Green Open Access. [25](#)
- NI, L., LIU, Y., LAU, Y. & PATIL, A. (2003). Landmarc: Indoor location sensing using active rfid. 407–415, cited By 1000. [16](#)
- NURWULAN, N. & SELAMAJ, G. (2020). Random forest for human daily activity recognition. vol. 1655, cited By 8. [21](#)
- PIENAAR, S. & MALEKIAN, R. (2019). Human activity recognition using lstm-rnn deep neural network architecture. Cited By 62. [22](#)
- ROY, P. & CHOWDHURY, C. (2021). A survey of machine learning techniques for indoor localization and navigation systems. *Journal of Intelligent and Robotic Systems: Theory and Applications*, **101**, cited By 67. [25](#)
- ROY, P. & CHOWDHURY, C. (2022). A survey on ubiquitous wifi-based indoor localization system for smartphone users from implementation perspectives. *CCF Transactions on Pervasive Computing and Interaction*, **4**, 298–318, cited By 12. [18](#)
- SALAMAH, A., TAMAZIN, M., SHARKAS, M. & KHEDR, M. (2016). An enhanced wifi indoor localization system based on machine learning. Cited By 99. [25](#)
- SANCHEZ-PI, N. & MOLINA, J. (2010). A multi-agent approach for provisioning of e-services in u-commerce environments. *Internet Research*, **20**, 276–295, cited By 13. [5](#)
- SANPECHUDA, T. & KOVAVISARUCH, L. (2008). A review of rfid localization: Applications and techniques. vol. 2, 769–772, cited By 183. [15](#)
- SCALMATO, A., SGORBISSA, A. & ZACCARIA, R. (2013). Describing and recognizing patterns of events in smart environments with description logic. *IEEE Transactions on Cybernetics*, **43**, 1882–1897, cited By 17. [27](#)

- SINGH, N., CHOE, S. & PUNMIYA, R. (2021). Machine learning based indoor localization using wi-fi rssi fingerprints: An overview. *IEEE Access*, **9**, 127150–127174, cited By 49. [26](#)
- SONG, C.H., WU, J., WASHINGTON, C., SADLER, B.M., CHAO, W.L. & SU, Y. (2023). Llm-planner: Few-shot grounded planning for embodied agents with large language models. [28](#)
- TAPIA, D., ALONSO, R., DE PAZ, J. & CORCHADO, J. (2009a). Introducing a distributed architecture for heterogeneous wireless sensor networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, **5518 LNCS**, 116–123, cited By 30. [6](#)
- TAPIA, D., BAJO, J. & CORCHADO, J. (2009b). Distributing functionalities in a soa-based multi-agent architecture. *Advances in Intelligent and Soft Computing*, **55**, 20–29, cited By 16. [5](#)
- TAPIA, D., FRAILE, J., RODRÍGUEZ, S., ALONSO, R. & CORCHADO, J. (2013). Integrating hardware agents into an enhanced multi-agent architecture for ambient intelligence systems. *Information Sciences*, **222**, 47–65, cited By 83. [5](#)
- THILAKARATHNE, H., NIBALI, A., HE, Z. & MORGAN, S. (2022). Pose is all you need: the pose only group activity recognition system (pogars). *Machine Vision and Applications*, **33**, cited By 0. [21](#)
- WANG, P., GUO, B., WANG, Z. & YU, Z. (2022). Shopsense:customer localization in multi-person scenario with passive rfid tags. *IEEE Transactions on Mobile Computing*, **21**, 1812–1828, cited By 8. [27](#)
- WEI, S.E., RAMAKRISHNA, V., KANADE, T. & SHEIKH, Y. (2016). Convolutional pose machines. vol. 2016-December, 4724–4732, cited By 2059. [22](#)
- WU, C.M., CHEN, X.Y., WEN, C.Y. & SETHARES, W. (2021). Cooperative networked pir detection system for indoor human localization. *Sensors*, **21**, cited By 7. [vii](#), [15](#), [16](#)
- XU, H., WU, M., LI, P., ZHU, F. & WANG, R. (2018a). An rfid indoor positioning algorithm based on support vector regression. *Sensors (Switzerland)*, **18**, cited By 68. [vii](#), [16](#), [17](#)
- XU, L., YANG, W., CAO, Y. & LI, Q. (2018b). Human activity recognition based on random forests. 548–553, cited By 36. [21](#)

REFERENCES

- ZHANG, L., LI, Y., GU, Y. & YANG, W. (2017). An efficient machine learning approach for indoor localization. *China Communications*, **14**, 141 – 150, cited by: 26. [25](#)
- ZHU, J., LI, Q., CAO, R., SUN, K., LIU, T., GARIBALDI, J., LI, Q., LIU, B. & QIU, G. (2019). Indoor topological localization using a visual landmark sequence. *Remote Sensing*, **11**, cited By 14. [vii](#), [12](#)