Universidade Federal de Pernambuco
Centro de Informática
Programa de Pós-graduação em Ciência da Computação

**Gibson Belarmino Nunes Barbosa**

# A MAS-based Architecture for IIoT Safety Applications

Recife
2022

**Gibson Belarmino Nunes Barbosa**

# A MAS-based Architecture for IIoT Safety Applications

Doctoral thesis presented to the Postgraduate Program in Computer Science of the Center of Informatics of the Federal University of Pernambuco, to obtain the title of Doctor in Computer Science.

**Concentration area**: Computer network
**Advisor**: Djamel Sadok
Co-advisor: Luis Ribeiro

Recife
2022

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

B238m Barbosa, Gibson Belarmino Nunes
*A MAS-based Architecture for IIoT Safety Applications* / Gibson Belarmino Nunes Barbosa. – 2022.
160 f.: il., fig., tab.

Orientador: Djamel Sadok.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2022

Inclui referências e apêndices.

1. Redes de computadores. 2. Internet das coisas. I. Sadok, Djamel (orientador). II. Título.

004.6 CDD (23. ed.) UFPE - CCEN 2022-178

**Gibson Belarmino Nunes Barbosa**

**“A MAS-based Architecture for IIoT Safety Applications”**

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação. Área de Concentração: Redes de Computadores e Sistemas Distribuídos.

Aprovado em: 06/09/2022.

**Orientador: Prof. Dr. Djamel Fawzi Hadj Sadok**

# ABSTRACT

The Internet of Things (IoT), and in particular, the concept of an Industrial Internet of Things (IIoT), is one of the key technological pillars of the Fourth Industrial Revolution, also known as Industry 4.0. In this context, an area of considerable interest is safety, whereby multiple intelligent sensors may be permanently connected to a central system to autonomously or semi-autonomously identify safety hazards. Vision systems emerged as popular sensors leveraged in such safety domain as they can simultaneously monitor many different safety concerns. However, the continuous video stream transmission and the increasing number of intelligent devices in IIoT networks introduce additional demand for network resources. There is a risk that the network may fail in the timely handling of video traffic in the case of large scenarios. This work proposes and discusses a reference architecture for the IIoT. It has been based on Multi-Agent Systems and is primarily intended to identify and manage safety risks. The architecture allows the simultaneous insertion of multiple sensors into the system. Input from the different sensors is then dynamically examined and weighed accordingly to estimate the risk level for any given situation. The architecture explores sensor-level intelligence (at the edge layer) to mitigate the network overloading problem. Edge agents quickly assess the risk, deciding whether or not to forward their signals to a local cloud agent for further processing. The cloud agent can then selectively request more information from other distributed edge agents. The architecture is tested in a use case for assessing operators' safety in the assembly of aircraft components and uses intelligent vision systems as safety monitoring devices. In the selected use case, the accuracy of the system and its impact on the network load are evaluated. The results show that the proposed architecture allows and benefits from the distribution of the processing to the edge. It reduces the load in the network by avoiding the transmission in the absence of risks, without losing accuracy, compared to when using centralized continuous and direct transmissions from the distributed sensors.

**Keywords**: multi-agent systems; industrial internet of things; industry 4.0; risk assessment.

## RESUMO

A Internet das Coisas (do inglês *Internet of Things - IoT*), em particular, o conceito de Internet das Coisas Industrial (do inglês *Industrial Internet of Things - IIoT*), é um dos principais pilares tecnológicos da Quarta Revolução Industrial, também conhecida como Indústria 4.0. Neste contexto, uma das áreas de interesse é a segurança contra acidentes, onde vários sensores inteligentes podem estar permanentemente ligados a um sistema central para identificar de forma autônoma ou semi-autônoma os riscos para os humanos. Os sistemas de visão são sensores popularmente usados para identificar riscos, pois podem monitorar simultaneamente diferentes situações perigosas. No entanto, a transmissão contínua de fluxo de vídeo e o número crescente de dispositivos inteligentes nas redes IIoT introduzem pressão adicional na rede. Isso pode levar a uma sobrecarga nos seus recursos. Para lidar com essa situação, este trabalho propõe e discute uma arquitetura de referência para a IIoT baseada em Sistemas Multiagentes, destinada principalmente a identificar riscos de segurança. A arquitetura permite que vários sensores sejam conectados ao sistema. A entrada dos diferentes sensores é então avaliada dinamicamente à medida que a situação de risco evolui. A arquitetura explora a inteligência a nível de sensor (na borda da rede) para mitigar o problema de sobrecarga da comunicação. Os agentes na borda da rede avaliam rapidamente o risco, decidindo se devem ou não encaminhar seus sinais para um agente de nuvem local para processamento adicional. O agente na nuvem pode então solicitar seletivamente mais informações de outros agentes na borda. A arquitetura é testada em um caso de uso para segurança de operadores na montagem de componentes de aeronaves e utiliza sistema de visão inteligente como dispositivos de percepção. No caso de uso selecionado, a precisão do sistema e seu impacto na carga da rede são avaliados. Os resultados mostram que a arquitetura proposta permite a distribuição do processamento até a borda. Ela reduz a carga na rede evitando a transmissão em situações que o trabalhador está seguro, sem perder a precisão, em comparação com uma transmissão direta centralizada feita por câmeras.

**Palavras-chaves**: sistemas multiagentes; internet das coisas industrial; indústria 4.0; avaliação de risco.

# LIST OF FIGURES

# LIST OF TABLES

# CONTENTS

# 1 INTRODUCTION

Industry 4.0 (Rüßmann et al. (2015)) is considered the fourth step among different evolutionary milestones that revolutionized industrial activities. The Internet of Things (IoT) is a key enabling concept of Industry 4.0. It represents a set of technologies that permits sensors, machines, or any device that embeds some processing capability to be connected to a network. IoT is expected to connect a massive number of devices to the Internet. Some studies, such as those reported by Sharma, Shamkuwar e Singh (2019) and Miraz et al. (2015)), point to this number being around tens of billions. The Industrial Internet of Things (IIoT) adheres to this tendency while considering industrial devices, requirements, and processes (Sisinni et al. (2018)). However, connectivity alone does not solve any industrial problem, and emerging IIoT frameworks and architectures must be sufficiently flexible to address several different use cases. One such case is typical concern with safety encountered across industry plants (Wu, Wu e Yuce (2018) and (GNONI et al., 2020)). Multi-sensor environments provide an interesting framework for evaluating safety risks in such complex scenarios. Here, both sensing devices and servers present in the environment can work together to offer a less risk-prone environment by monitoring and predicting possible risks, reacting to them, or alerting nearby persons and safety managers.

In the survey Insurance (2021), the Liberty Mutual Group analyzed the leading causes and costs associated with injuries. Their findings indicate that when considering the ten most expensive injuries, the human influence is decisive in its occurrence, both in the cause of the accident and in the severity of the consequences. Accidents may occur due to different factors, such as taking inappropriate actions, human inexperience, and lack of equipment inspection, among other issues. For instance, injuries may occur when employees do not follow safety recommendations such as using Personal Protective Equipment (PPE). Although a great deal of research and development effort has been invested in order to ensure that machines may operate safely when collaborating with humans (Robla-Gomez et al. (2017) and Lasota et al. (2017)), it is still important to be aware of the human factors, often difficult to control, that can result in accidents.

A prominent case is a human access to high-risk areas, where special care must be taken with the use of protective equipment, knowing how to handle machines and how and when to interact with other workers. Unauthorized access can cause, for example, collision risks. Collisions in industries can result in severe accidents for human beings since a passerby can disturb the worker doing a delicate maintenance activity with heavy machinery.

The report in oshacampus.com (2018) lists some actions that can be taken to avoid accidents or reduce their consequences. One of them is to comprehensively analyze possible risks to workers in the work environment. Identified hazards must then be alerted to with signposts. In addition, a company must provide continuous training and mandate the use of PPE. It is also essential that there is constant monitoring of the work environment and the habits of workers. Since accidents tend to be related to human errors, as previously reported, one must seek approaches that, when possible, do not entrust safety management exclusively to humans. One way to achieve this goal is to consider using a computational system that relies on sensors and actuators present in the environment to assess risk at all times.

Vision systems are ordinary sensors that can be deployed across workplace environments in order to identify safety events. However, these devices require a network with sufficient bandwidth resources, which existing industrial networks may lack as they are not provisioned with this type of demand in mind. As a result, some works such as Chen et al. (2019a), Berardini et al. (2021) and Zhou et al. (2018) suggest strategies that migrate image processing algorithms closer to the cameras located at the edge of the network. This method is more formally known as Edge Computing (Varghese et al. (2016) and Hu et al. (2015)). With that, processing can be distributed to devices at the network's edge, reducing the need to transmit sizeable amounts of raw data. Although, adding a processing unit between the two communicating devices includes the retransmission delay, which should be more efficient than direct transmission. This should be carefully analyzed, especially when using embedded devices with limited processing, such as a Raspberry Pi[1], even though they offer the advantages of being cheaper and less energy-consuming for scaled applications.

In addition, the sensing data coming from different sources must be combined coherently. Despite the geographical distribution of the data sources across a factory, the collected data must be synchronously integrated in order to produce summarized information and results. Conflicting and irrelevant information must be identified and removed in order not to introduce noise into risk detection. There is, therefore, a need for the presence of a well-provisioned computational system to carry out these tasks. At its heart, there is an orchestration module that manages and controls communication among the different components or agents, thus forming a Cyber-Physical System (CPS) (Lee (2008)).

According to Stuart, Peter et al. (2003) Intelligent agents are entities that have the rule of perceiving the environment, processing the perception, and producing a response to achieve some goal. The union of several intelligent agents interacting with each other forms a Multi-Agent System (MAS), which can be applied to solve a broader range of problems than a single agent could. Mainly, it is suitable for avoiding bottlenecks evident in centralized systems by distributing computational resources on the network.

It is clear that a risk detection application must strike a trade-off between the level of accuracy it achieves from listening to an increased number of input devices active at

[1] www.raspberrypi.org

any given moment and the overhead and high cost due to sending potentially redundant information. System cost, as used in this context, includes the consumption of CPU, network, and power resources. In addition to the above concerns with bandwidth and CPU processing, response time may suffer as processing a high number of collected sensing data tends to last longer (REILY; REARDON; ZHANG, 2021). This is particularly the case when considering a multi-camera system (OLAGOKE; IBRAHIM; TEOH, 2020) where the images from different camera perspectives are streamed simultaneously over the network. As more camera frames encapsulated into packets travel across a network, they compete for bandwidth resources. As a result, these packets suffer from additional communication delays, often due to propagation delays and network congestion (SORO; HEINZELMAN, 2009). Optimizing packet transmission performance among the devices and architectural agents, considering bandwidth and latency, are essential design requirements.

This multi-source data transmission can be optimized by taking advantage of the concept of attention. A biological attention mechanism found in humans filters the large amount of information available in the environment for living beings to process. Therefore, it focuses on perceiving and processing only the most relevant or what stands out the most, rejecting noise information that does not contribute to the objective task. Ning et al. (2019) applies this idea to propose a selective sensing framework to reduce the amount of raw data in networks (IoT).

This work raises the following research question: before investing in additional processing and communication resources, would it be possible to reduce their consumption by cleverly orchestrating sensors? The hypothesis is that a reference architecture for security in an industrial environment can facilitate the work of implementing communication devices on the network. It can be thought of to deal with issues encountered in such an environment, such as heterogeneity and the growing number of devices. It can also address the needs of workers in this environment in a way that allows reconfigurability to address specific security issues. In addition, the architecture could handle bipartisan communication seeking to avoid sending unnecessary data to the decision-making process for which it was configured. This all prevents unnecessary traffic from causing a bottleneck that prevents relevant information from being transmitted. For example, a robot on a collision course with a human must receive a command to dodge as quickly as possible. However, if several cameras transmit continuously on the same network, the command to the robot may not arrive in time. To put it more simply, the present work analyzes an approach whereby distributed agents process camera frames locally to obtain risk information at the network's edge. They only transmit the frames through the network for more robust and accurate processing when risks are locally identified or when a higher-level agent requests.

This work proposes a reference architecture considering a Multi-Agent System (MAS) structure to decide when to transmit information between low-end Edge devices and high-end Server agents. A distributed agent represents a processing unit located at the Edge computing layer that can process locally or pass the data to a server in a cloud for further heavy-duty and high-accuracy processing. This proposal is thought in order to reduce the potential overload that could be experienced by IIoT networks by avoiding the unnecessary transfer of all data to centralized servers. It also proposes an attention mechanism to select the most critical input source to be processed with the highest priority. A safety-related use case is applied to evaluate the proposed architecture.

## 1.1 OBJECTIVES

Design and build a MAS-based reference architecture to reduce the amount of information streamed in a centralized IIoT network, considering security applications. However, the desired reduction should not significantly impact the initial accuracy of the system. The solution seeks to facilitate the implementation of a safety system that may be tailored in order to apply to different industrial environments. It saves network resources and controls latency by avoiding transmitting sensor information that does not contribute to risk identification. The adopted approach must be flexible, scalable, and capable of incorporating different sensors and agents.

### 1.1.1 Specific Objectives

- Prototype a solution for safety based on the proposed architecture.
- Evaluate the impact of the insertion and removal of edge devices on the video stream.
- Evaluate the impact on decision accuracy and network resource utilization when deploying the proposed architecture in a safety use case.
- Evaluate the impact on network resources as a result of using an Attention module that intelligently selects more relevant input for processing.

### 1.1.2 Contributions

The present research contributes with the following:

- A MAS-based reference architecture for IIoT edge-cloud computing.
- A system to detect collision risks in industrial environments.
- A multi-camera dataset that describes a human interaction activity in the industry.
- Evaluation of the utilization of edge devices to forward camera input video streams across a WiFi network.

As part of the proposed investigation and development, some contributions are seen as important but not essential to the adopted solution. These are related to the formal description of risk in the context of Human-Robot Interaction (HRI). As a result, the text contains Appendix A, which specifies an ontology for such a context, and Appendix B depicts the system applied to HRI. Finally, Appendix C presents an evaluation of a scheduler that adds fuzzy logic to the proposed architecture for selecting object tracking agents. The appendices add the following contributions to the work:

- Definition of a reference architecture for HRI.
- Specification of an ontology for the composition of risk in HRI.
- The design of a system to detect human-robot collisions considering a wireless communication network.
- A fuzzy-based scheduler evaluation applied to object tracking agents.

### 1.1.3 Organization

This work is organized as follows. Chapter 2 highlights important technologies and subjects for developing the project. Chapter 3 presents the works related to this proposal. Chapter 4 describes the proposed architecture. Chapter 5 details the use case, prototype implementation and discusses the experiments and results. Finally, Chapter 6 summarizes the main findings of the work.

# 2 BACKGROUND

This chapter sheds light on the different technologies that this work leverages. They include relevant architectural designs, the world of IIoT, and major human safety considerations encountered in the industry context.

## 2.1 IIOT

When thinking of an industrial environment, the image that comes to mind is one that possibly contains manufacturing machines, Computer Numerical Control (CNC) machines, robots, sensors, and last but not least, workers. Human operators usually use these devices and closely coordinate their activities with them to reach a mutual goal. These devices have digital interfaces and continuously collect and process events, exchange information and share data as part of their tasks and to improve production efficiency. The Industrial Internet of Things (IIoT) leverages the power of distributed processing to improve industrial services. IIoT Devices may be activated or not, based for example, on their current demand. As a result, adaptive energy-saving schemes should be put in place.

The IIoT and the Internet of Things (IoT) share some essential underlying architectural design technologies and concepts. They both rely on the interconnection of objects and information exchange among these as part of implementing their services. Due to their similarities, both architectures leverage technologies such as cloud computing and wireless communications. Some protocols and communication paradigms are also common, like the Message Queue Telemetry Transport (MQTT) standard protocol that implements the publish/subscribe message exchange style. In addition, both support wireless communications and scalability for connecting massive numbers of objects. Despite this, there are differences stretching beyond the mere application area.

The IoT draws from the general notion of connecting devices in *normal* environments with the presence of end-users. Examples of IoT deployment include traditional user-centric spaces such as an office, a home, a city, a farm, a vehicle, etc., where so-called smart devices are used to improve user experience. IIoT, on the other hand, has a new set of requirements that reflect manufacturing and automation services. Here, one is not concerned with user comfort or convenience. Instead, safety and latency constraints become part of the major design requirements.

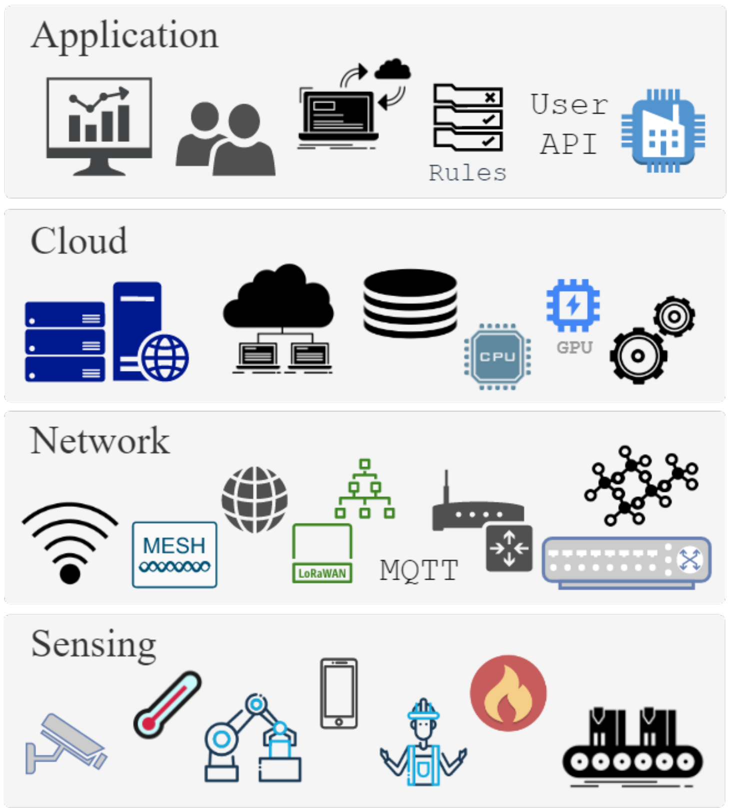
The types of objects used in industries are very different from those encountered in IoT networks. IIoT adopts a class of devices known for their reliability and ruggedness to withstand the harsh conditions typically encountered in manufacturing. In addition, IIoT devices such as Programmable Logic Controllers (PLC) and Robots require a timely and deterministic operation. The presence of heavy machines and robots demands precise process control in order to ensure specific safety requirements for nearby workers and an environment with reduced risks. The real-time requirements for IIoT networks are often much more demanding than those of IoT. Typical transmission delays for automation networks are in the range of a few milliseconds only. Higher delays or the loss of essential commands issued to manufacturing machines as part of process control can desynchronize manufacturing operations or prevent an emergency alert from being raised in time. Compared with IoT, the IIoT network must operate in a more reliable and deterministic manner. Failing to do so would result in financial losses and harmful safety consequences to humans.

The IIoT spans several specialized applications. It can be applied to improve the operation of industrial networks by leveraging wireless technologies such as the IEC 62591 WirelessHART standard, developed by International Electrotechnical Commission (IEC) based on the existing and widely used Highway Addressable Remote Transducer Protocol (HART) (Std (2009))as well as the ISA100.11a (Raptis, Passarella e Conti (2020)). The latter is an alternative wireless networking standard for industrial automation developed by the International Society of Automation (ISA). Also, IIoT is tailored to offer industrial services that benefit those who control the factory. New services may be specified. For example, machine operating time can be rented out during idle periods. In addition, it benefits those who hire this service, as it prevents them from setting up their own complete production system when there is limited demand for a product, for example. As a network infrastructure, IIoT may define a management plane that implements monitoring services. These can be used to build applications that detect and assess the state of the components of an industrial plant at all times, which can be used by collecting and analyzing sensor data. As a result, machine learning technologies are considered important tools to process this data and build models that can identify patterns and predict potential problems in the plant and its surroundings.

Many recent works have proposed general models for representing IoT or IIoT systems infrastructure. They include the contributions in Younan et al. (2020), Khan et al. (2020), Lee, Bae e Kim (2017) and Boyes et al. (2018). Although there are some differences among the proposed IIoT models, these mostly center around the taxonomy or the scale of the modules they support. In other words, there are some noticeable common underlying structural components among existing IIoT infrastructure models. Thus, a high-level view of IIoT architectures can be summarized by considering the following modules or layers: Sensing, Network, Cloud, and Application. They are illustrated in Fig. 1 and presented next.

*Sensing layer* represents the physical end devices connected to the network. The infrastructure must scale in order to support a large number of devices. This is a crucial design requirement as existing and emerging infrastructures tend to grow with the advancement of technology, the expansion of industrial plants, the introduction of new products, and the

Figure 1 – IIoT Architecture.



Source: The author.

increased complexity of automation processes. Devices typically specialize in attending a given class of applications and their requirements. They implement different information communication interfaces; for example, some use USB cable connections, while others use Wi-Fi, WirelessHART, or ISA100.11a communication. In addition, IIoT end devices differ in terms of their capabilities. They may have a wide range of processing capabilities and power consumption features and may collect and transmit data at different rates. This is clearly a heterogeneous layer due to the wide-ranging requirements among the manufacturing processes, the adopted business models, regional regulatory law, and many other factors that influence our choice of devices and their operation mode. The subsequent heterogeneity of the collected data dictates the need for IIoT systems to implement mechanisms capable of coherently merging such data. Time synchronization of collected data is essential for its analysis. As such, the underlying networks must support real-time and isochronous communications, bounded delay, and be capable of scheduling transmission events.

*Network layer* is concerned with the connectivity of devices and the specification of upper-layer protocols to support the information flow in the particular context of process automation. The supported protocols must be capable of interworking in order to support end-to-end information exchange. Interworking among protocols can be a daunting task, as it requires managing devices with different protocols at different OSI abstraction layers, ranging from the physical to the application layers. In addition, the choice of communication protocols and network technologies is highly influenced by energy consumption requirements. There is a need for IIoT networks to be efficient enough to optimize the network infrastructure. Manufacturing plants and industries are encouraged to reduce energy consumption across all systems. Quality of Service is also an important design feature, as the data must have a guarantee of delivery, avoiding losses or delays. Finally, meeting data security requirements such as confidentiality, integrity, and availability is essential.

*Cloud layer* expands the constrained capabilities of devices, providing more processing and storage power. The cloud infrastructure is used to handle more demanding jobs, which previously could not be processed due to the limitation of devices at the edge. Therefore, it can support the analysis of the collected data sent at the sensor layer using advanced techniques such as machine learning methods. In addition, the cloud layer is also seen as a platform service provider for the application layer, one that abstracts the underlying infrastructure and sensing services.

*Application Layer* hosts the upper layer services and applications targeted by the industry. To achieve the goals, the application layer supports applications that consume services from the cloud layer, which can be used, for example, to display collected data and analyze tendencies through an application programming interface (API). A typical web-based API, using the Representational State Transfer (REST) architectural style, often allows applications to communicate and access each other's services.

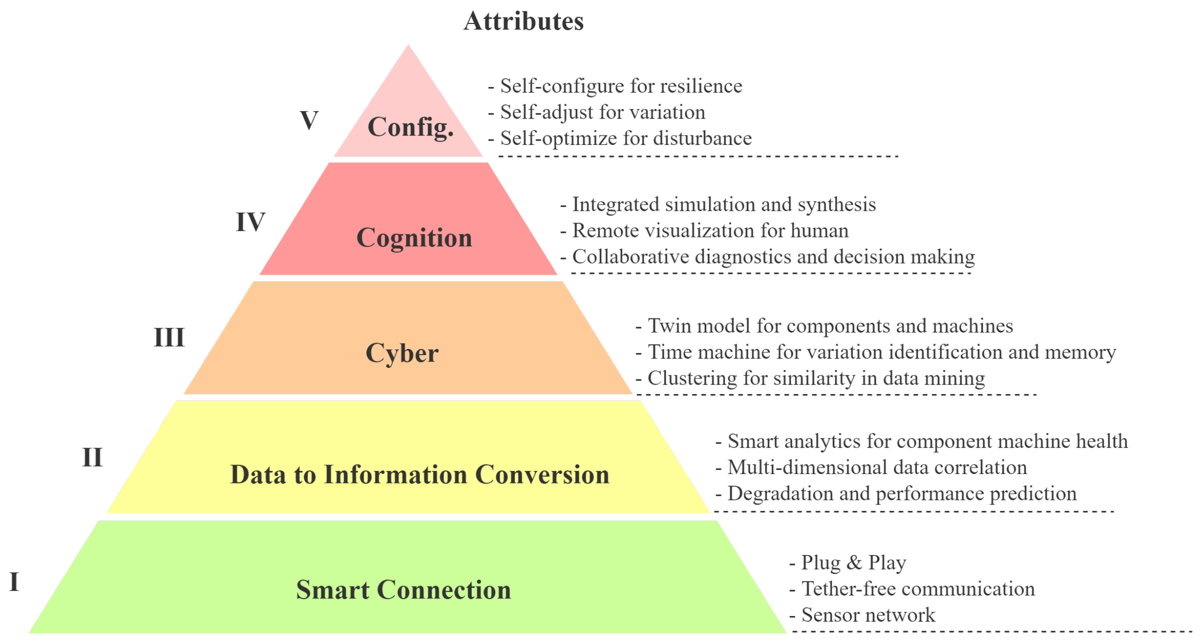
## 2.2 REFERENCE ARCHITECTURES

As reported by Fraile et al. (2019), a reference model is a specification of system architectures that use a unified framework and language. It allows systems produced from such a model to interoperate using a unified understanding. Therefore, they are structures that unify different modules to achieve a well-defined objective. These modules can be methods, services, patterns, applications, or a set of solutions that can interact with each other.

Reference architectures should specify and develop an easy-to-reproduce document to be reused in different implementations for projects in diverse application domains. This document reports the requirements for such an implementation and justifies using the best technologies for the implementation context. Templates can be adopted to bring previous developers' experiences to benefit future projects. Templates often contain information representing best practices and answer architecture-related questions that may arise. In other words, they contribute to avoiding the misunderstanding of architectural specifications, development errors, and delays.

Cyber-Physical Systems (CPS) reference architectures were proposed to deal with some IIoT limitations such as security, connectivity, standardization, and interoperability. The 5C (Lee, Bagheri e Kao (2015)) is the most popular and exploited architecture for such domain (Pivoto et al. (2021)). It considers five layers structured as a pyramid. The first layer, at the base of the pyramid, is the *Smart Connection layer* that represents the physical devices connected to the network, which are responsible for the data acquisition and its transmission. The second layer is the *Data to Information Conversion layer*. This layer converts the data structure of devices or their information models into meaningful information for the system. The third layer is known as the *Cybernetic layer*. It offers device abstraction in the cybernetic or computing infrastructure allowing devices to represent and communicate information in an abstract mode, i.e., in a device and protocol-independent manner. The fourth layer is the *Cognition layer* where the application explores the extracted information for analysis, simulation, decision making, fault detection, diagnosis, and user visualization. Last but not least, there is the *Configuration layer* at the top of the pyramid. It allows machines and hosts to self-configure, self-adjust, and self-optimize without a human interface. Such automation, known as self-organization, eases the configuration of the IIoT control processes and the underlying infrastructure, speeds up recovery from software and hardware failures, and service reconfiguration to meet the performance requirements of new applications.

Figure 2 – 5C Architecture.


Attributes
V
Config.
- Self-configure for resilience
- Self-adjust for variation
- Self-optimize for disturbance
IV
Cognition
- Integrated simulation and synthesis
- Remote visualization for human
- Collaborative diagnostics and decision making
III
Cyber
- Twin model for components and machines
- Time machine for variation identification and memory
- Clustering for similarity in data mining
II
Data to Information Conversion
- Smart analytics for component machine health
- Multi-dimensional data correlation
- Degradation and performance prediction
I
Smart Connection
- Plug & Play
- Tether-free communication
- Sensor network

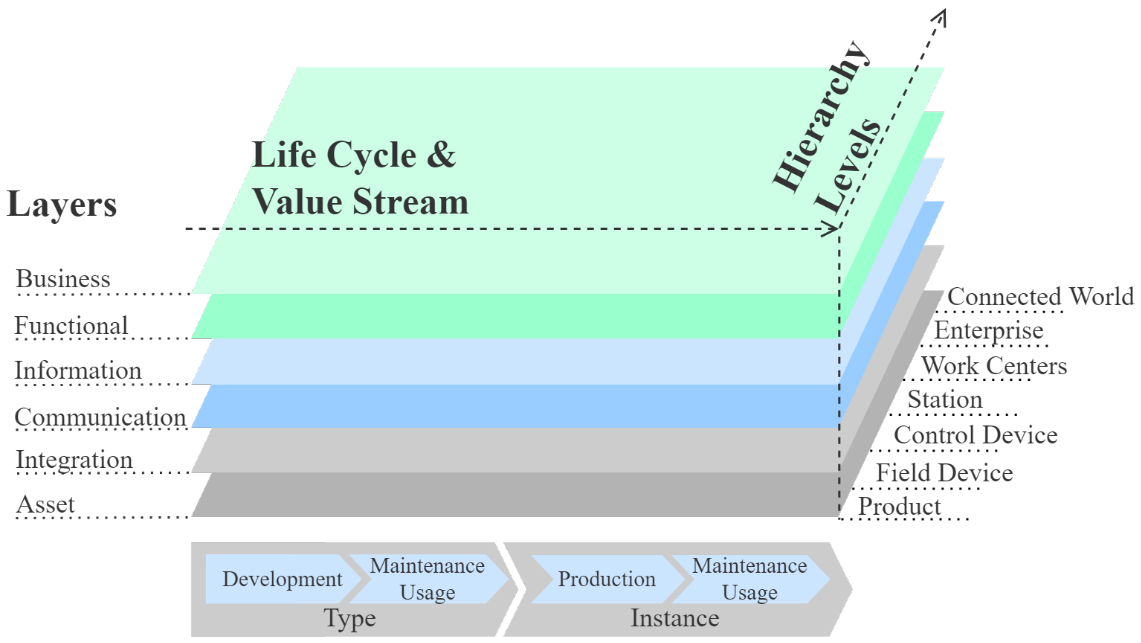
Source: Adapted from Lee, Bagheri e Kao (2015).

Two reference architectures, namely, Industrial Internet Reference Architecture (IIRA) and Reference Model for Industry 4.0 (RAMI 4.0), were developed to increase interoperability between applications in the IIoT. From the point of view presented in Lin et al.

(2017), the new IIoT architectures represent the union of Information Technology (IT) and Operational Technology (OT) to IIoT, where IT brings agility and speed, flexibility, cost reduction, business insight, and security. On the other hand, the OT guarantees efficiency, utilization, consistency, continuity, and safety.

The RAMI 4.0 (Adolphs et al. (2015) and Heidel (2019)) can be examined according to a three-dimensional view, where each axis represents the coordinates of its applicability, as shown in Fig. 3. The first axis illustrates the *Hierarchy Levels*. It represents the functional levels of factories. The idea is to adopt Industry 4.0 concepts as part of the RAMI 4.0 project. This specified axis has different levels: connected world, company, work centers, station, control device, field device, and product. The second coordinate is that of the *Life Cycle & Value Stream* directed to the needs of the product and its life cycle, such as its classification, development, and maintenance. The life cycle represents a period that begins early with production and ends with the disassembly or disposal of manufactured products.

Figure 3 – Reference Model for Industry 4.0 (RAMI 4.0).



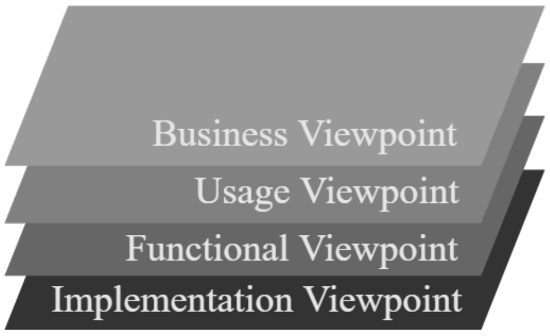
Source: Adapted from Heidel (2019).

The third coordinate consists of the architectural *Layers*, where this layering concept represents the integration with CPS systems. First, there is the *Business Layer* that considers the business lifecycle definitions such as rules to be followed and regulatory and legal conditions. The *Functional layer* represents where applications execute. It, therefore, supports the services used to meet the demands of the business layer. Secondly, the *Information layer* deals with the services responsible for managing the data, such as data processing, maintenance, persistence, provisioning, integration, and integrity. This layer represents the processing and transformation of data to support *Functional layer* requests.

The *Communication Layer* provides a homogeneous data format for communication and an interface for accessing the offered services by other parties. The *Integration Layer* is concerned with the connection between digital and physical machines. It reports events to devices. Also, it includes the documentation of assets, software, firmware, and Human-Machine Interfaces (HMI). Finally, the *Assets Layer* contains physical elements of the environment that are integrated into the complete system and whose information flows through the layers of the architecture.

The other reference architecture, namely, the IIRA (Lin et al. (2015)), was proposed by the Industry IoT Consortium Architecture Task Group. It is specified departing from some concerns that the proponents consider necessary and several common points set to achieve application objectives. These concerns are grouped into four points of view: *Business Point of View*, *Usage Point of View*, *Functional Point of View* and *Implementation Point of View*. Fig. 4 illustrates the viewpoints in the IIRA.

Figure 4 – The Industrial Internet Reference Architecture Viewpoints.



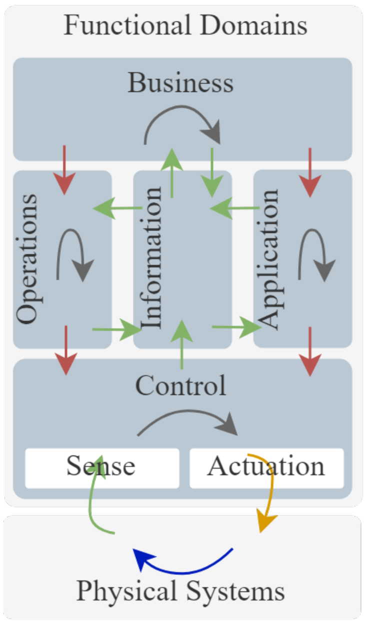
Source: Adapted from Lin et al. (2015).

The *Business Viewpoint* represents the relationship between the IIoT system and the business. It defines important parameters such as development objectives linked to the definition of stakeholders, business vision, regulatory context, and mapping of the objectives. The *Usage Viewpoint* refers to how humans and system components can operate to meet business definitions. Therefore, it considers users, systems, and tasks, as well as their roles and activities.

The *Functional Viewpoint* is where the IIoT system is actually designed. This point of view is closer to the implementation of the system architecture. There are five domains that fall within the *Functional Viewpoint*, which are illustrated in Fig. 5. There is first the *Control Domain* which deals directly with the sensor and actuators, controlling reading, writing, and communication in an abstract form to such devices. The *Operations Domain* manages the control of predictions, optimization, monitoring and diagnosis, deployment, and management services. The *Information Domain* represents functions aimed at data manipulation, such as data collection, analysis, modeling, and analysis. The *Application Domain* is related to the user's needs. For example, it can present data to users or allow them or other applications to configure the system using an Application Programming

Interface (API). The *Business Domain* reflects the inclusion of the business process in the system to operate considering the end-to-end operations of the IIoT system.

Figure 5 – Functional View of the Industrial Internet Reference Architecture.



Source: Adapted from Lin et al. (2015).

The last viewpoint of the IIRA is the *Implementation Viewpoint*. This one deals with the practical resources needed to maintain the system up and running, such as the technologies, infrastructure, components, and processes necessary to make it operational.

## 2.3 COGNITIVE ARCHITECTURE AND ATTENTION MECHANISM

The author Vernon (2014) defines cognition as the process by which an autonomous system perceives the environment, learns from experience, anticipates future events, acts to achieve goals, and adapts to changing circumstances. These characteristics are easily perceived in humans, which motivates understanding cognition. Studying how the human mind works allow advances in therapeutic treatments and methods to replicate cognition in machines. (GIBAUT, 2018) says that cognitive agents are intelligent agents endowed with artificial minds created from cognitive models.

Cognitive architectures are another specific class of reference architectures. Several scientific areas such as psychology, biology, and philosophy have been studied to extract concepts that support cognitive architecture development. Based on the definition proposed by Sun (2004), cognitive architectures can be defined as a formalization of an organized modular structure, where each module has specific functionalities to deal with different components of cognition. In addition, there is an interaction between the modules that allow for achieving the objectives of the architecture. Each cognitive architecture has specific goals. These can be the design of a simulated model of the human mind or the controlling of robot parts.

One of the main characteristics of such cognitive architectures is the attention mechanism. For Kaplan e Hafner (2006), attention is the process by which an agent focuses on some feature of the environment as opposed to a relative exclusion of other features. Attention can be divided into two classes. Passive attention is when an event that stands out (such as a loud noise) attracts an agent's attention. The second type is active attention, which is when an agent purposely engages in an activity set as its primary goal while still being able to choose how to interact with the environment. A case in point is the driving activity.

An agent with attention control can dynamically optimize processing and reduce the delay experienced as the result of information exchange. One way to achieve this fact is to apply an attention system that selects or filters the information that will be transmitted for further processing by other modules. This filtering usually considers various factors, including, for example, the usefulness of exchanged information. In other words, some of the architectural components should be able to evaluate how helpful information can be and only submit the one it thinks is still relevant for further processing. As a result, processing, storage, and communication resources may be saved, leading to a faster response to system changes. The result is an unattended or autonomous behavior that continually changes the system configuration to save resources. This is especially useful when considering a time-constrained distributed system. This is the case of the application scenario adopted in this work, where communication is a determining factor for reaction time when a person is at risk.

## 2.4 SAFETY

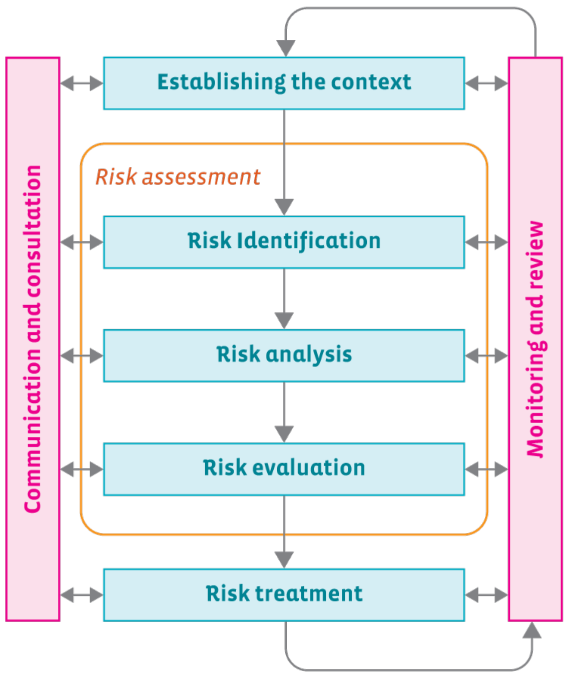
People are prone to making mistakes and failures when performing tasks, which can be caused by their inexperience, recklessness, or negligence. Even very experienced employees may commit mistakes. For example, they may forget or neglect to use necessary Personal Protective Equipment (PPE) when there is no constant supervision. Thus, one must recognize the usefulness and essential role of computer systems that seek to identify problems and automatically reduce the occurrence of human errors.

When discussing the topic of safety, it is relevant to report on the Morgan Stanley Automation Worldwide Industrial Automation Survey (Morgan... (2015)). It considers the opinion of 200 automation executives surveyed to assess the most critical factors that led to the adoption of IIoT in their factories. The main raised concern was to "Improve operational efficiency" with 47% of the votes. Additionally, worker safety (Enhancing worker safety) was cited by 14% of respondents. This demonstrates that safety measures are still not the priority for now. This can be worrying and dangerous, as damage to physical integrity can be irreversible and may result in serious financial expenses.

Experts worldwide propose different ideas to decrease the possibility of accidents in a hazardous environment. When they agree on the best behavior to deal with some situation, this can be put together as a standard that can be followed in the industries as a recommendation or government regulatory enforcement. Therefore, some patterns are discussed in the following paragraphs.

The Risk Management Process must follow several steps to ensure that all accident events and conditions are properly observed. Fig. 6 illustrates the process. In an overview, the ISO 31000:2009 standard (ISO (2009) and Purdy (2010)) state that a risk assessment process must define a context in which risk management is applied, in addition to the constant need to maintain communication and consultation with experts and users.

Figure 6 – Risk Management Process.



Source: Adapted from Purdy (2010).

Risk Assessment is divided into three stages. Understandably, it starts with Risk Identification, which focuses on knowing how, when, and where hazards can occur. Then there is the Risk Analysis, which aims to define the consequences and probability of risk occurrence. Finally, there is the Risk Evaluation, where a decision must be made about the severity of the risk and its priority over other risks. After the Risk Assessment phase, there is a step called Risk Treatment. It refers to a stage that selects countermeasures to end or lessen the consequences of a risk. Most of the time, it results in selecting the least harmful consequences. The ISO 31000:2009 recommends that the evaluation improvement process be continuous. Thus, the assessment and review of recorded risks and possible new risks must be reviewed periodically.

The work in Smith e Brooks (2013) addresses the specifics of risk management with a focus on safety. The authors make it clear that all risk management systems, including those for safety, must be compatible with the structure proposed by the ISO 31000:2009 recommendations.

The Hazard Rating Number (HRN) methodology was presented by Kirton (1995) and ABNT (2013). HRN is a strategy following the steps inserted in the risk management process. Hence, it is suitable for risk identification, analysis, and evaluation. This methodology uses a scoring system generated from the probability of an accident occurring, the frequency of exposure, the degree of possible damage, and the number of people exposed. This methodology uses a scoring system generated from the Likelihood of Occurrence (LO), the Frequency of Exposure (FE), the Degree of Possible Harm (DPH), and the Number of People (NP) at risk. There is a score for each one of these parameters. The result is the HRN index, which is merely the product of these parameters, as shown in the equation 2.1.

$$HRN = LO \times FE \times DPH \times NP \tag{2.1}$$

The ISO Specification 12100:2010 (ISO (2010)) considers safety in the design of machines to ensure they are safe when operating in an industry. Industrial robots are observed in the ISO 10218-1:2011 (ISO (2011a)) and ISO 10218-2:2011 (ISO (2011b)) standards. The ISO 10218-1:2011 focuses on safety in the design and development of robots, while the ISO 10218-2:2011 offers the requirements to reduce or eliminate risks when installing, operating, maintaining, or repairing robots. As a complement to the protocols ISO 10218-1:2011 and ISO 10218-2:2011 standards, there is also the ISO/TS 15066:2016 (ISO (2016)) recommendation. This standard offers guidelines for workplace safety and collaborative robotic systems in industries. ISO/TS 15066:2016 focuses on robots sharing the same space and tasks with workers.

The work in Robinson (2019) raises the challenges encountered when linking the Industrial Internet of Things (IIoT) to functional safety. A machine offers functional safety when an automatic protection system triggers faults that are dangerous for humans. The international standard IEC 61508 (BELL, 2006) proposes functional safety measures for safety-related electrical, electronic and programmable systems. Certified sensors based on such standards are important when implementing safety systems since the measured data have an assured confidence level. Although it is necessary to encourage the use of this type of sensor, the high costs and the difficulty of its integration into established systems can sometimes make its adoption difficult. The research in Gall (2008) offers an interesting discussion on the use of the standard, including the problems and benefits related to the certification of equipment and systems.

## 2.5 DISCUSSION

This chapter presented different concepts that can be used to build a system that promotes safety in the work environment.

Industries increasingly include the IIoT. Therefore, proposing a reference architecture to deal with industries brings the need to evaluate the characteristics of such technology. This was done in the present chapter, mainly portraying its evolution when comparing the IoT and the analysis of structural components. This is all simplified and organizes the architectural proposal of this work according to what is already established.

The same idea applies to reference architectures for industries, which, most of the time, as demonstrated earlier in this chapter, apply to a wide range of applications, and their complexity and consequent learning curve for application is a deterrent for workers to apply it in some specific contexts. However, the organization provided is essential for replication and inspiration in proposing derived architectures. Attention mechanisms are interesting ideas to be applied in conjunction with the IIoT scenario. Such mechanisms can benefit the quality of the network by focusing on the relevant flows in the network, while reducing the prioritization of less important ones at that moment.

The quality of the network is a factor that can impact the worker's safety, as relevant information to avoid accidents may not arrive or arrive late at the destination. Different factors can contribute to the instability of industrial networks, such as the increased number of devices and the different types of devices. There is also the evolution of technologies, which imposes the need for more bandwidth, as seen when using cameras, as special sensors, with increasingly higher resolution. Some devices that demand more resources (such as cameras) can be a barrier to the communication of others, as they overload the network, causing delays and data loss. Thus, the idea of an architecture that can use IIoT concepts to offer a system that reduces data transmission can be essential in some IIoT contexts.

# 3 RELATED WORK

This chapter presents research with contributions related to the present work. The discussion focuses on safety systems for industry, multi-agent vision systems, architectures that support attention mechanisms, and several existing proposals that fall in the context of IIoT.

## 3.1 SAFETY SYSTEMS FOR INDUSTRY

This section provides an overview of works that propose different systems with support for intelligent approaches designed to ensure some level of safety when used in an industrial environment.

The first among these is a strategy used in anomaly identification in historical data. The work in Kudryavtsev, Yemelin e Yemelina (2018) seeks to prevent accidents at hazardous industrial sites where prevention is achieved by developing a system that processes information from industrial safety monitoring sensors. The authors develop a method that considers the processing of statistics for historical data to extract information and identify the level of acceptable reported events. Consequently, preventive measures can be taken based on the result of the statistical analysis. The proposal was applied to risk management at hazardous industrial sites in the Republic of Kazakhstan. According to the authors, its implementation made it possible to quickly and objectively determine the state of risk and possible accidents in industrial facilities, therefore preventing the occurrence of emergencies.

The research in Hashimoto et al. (2020) proposes an accident prevention system that detects anomalies using video information streamed from security cameras. The authors use identification based on unsupervised learning algorithms to find data that differ from the normal distribution. More specifically, this work uses deep learning techniques to manipulate time series data for anomaly detection. Such strategies can identify the behavior of factory employees classified as being unlike their *normal* activities they commonly perform on the factory floor.

The study in Askarpour et al. (2019) proposes a methodology to evaluate human behavior, resulting in a formal model of operators' behavior. This model is then used to understand dangerous situations arising from human error. Engineers can use it before implementing a system as an initial step that helps improve designs based on operator behavior.

The authors in Teimourikia e Fugini (2017) propose a real-time safety management approach for use in smart work environments. The objective of the published article is to present a methodology for adopting the ISO 31000:2009 standard in real-time, based on the MAPE-K cycle (*Monitor-Analyze-Plan-Execute plus Knowledge*) (Iglesia e Weyns (2015)). MAPE is an approach that aims to deal with dynamic self-adaptive systems based on monitoring, analysis, planning, and execution steps. MAPE-K, on the other hand, is formed when knowledge representation is added to the previous steps. To achieve their goal, the authors characterize the knowledge of the environment using an ontology. The work presents a use case that shows how the ontology can be instantiated to create an abstract model and how users can define restrictions based on an organization's safety protocol. They list some crucial factors such as risks, consequences, and prevention strategies that should be adopted. The work of Teimourikia, Fugini e Raibulet (2017) complements the previous one (Teimourikia e Fugini (2017)) by proposing a solution that monitors dangerous events and analyzes the risk and its consequences to plan preventive strategies in real-time. Their approach is based on Bow-tie risk analysis mapped to a Bayesian network (Khakzad, Khan e Amyotte (2013)). It is used to control worker access to machines across workspaces.

Another work that uses ontology and Bayesian networks is proposed by Aziz, Ahmed e Khan (2019). It is an ontology-based hazard identifier that standardizes information, facilitates computational processing, improves knowledge sharing between humans and robots, and establishes information that other industries can reuse. Based on this ontology, the reasoning is performed using multi-entity Bayesian networks (Laskey (2008)) and produces quantitative risk estimation results. To achieve this, the research identifies the likely hazards considering the mutual influence between the different hazards and the operational and environmental conditions.

A risk management system based on a quantitative model for risk modeling is proposed by Fugini, Raibulet e Ubezio (2012). The system perceives the environment, identifies and analyzes the risks, and then generates a map highlighting areas with high-risk levels. In addition, the work proposes corrective measures for some risky situations.

Some reviewed related works offer strategies that deal with the risk in environments with the presence of actuator devices. Robots are an example of actuators that can cause serious harm to workers. For instance, the research presented in Inam et al. (2018) develops a method for risk assessment in the context of human-robot collaboration. The work describes the application scenario, showing the collaboration situations, where the dangers can occur, and how to deal with them. The system generates a graph that semantically represents the scenario and is used for risk assessment.

The idea of Speed and Separation Monitoring (SSM) reported in the ISO/TS 15066:2016 standard (ISO (2016)) seeks to maintain the distance between operators and robots as a way of ensuring safety during the operation of tasks. For instance, SSM can be applied to adapt robot speed according to human distance. The work published in Byner, Matthias e Ding (2019) uses this idea to propose a dynamic human-robot separation, in which the safety distance between the person and the robot may vary according to the relative speed between them. Another option is to actively modify a robot's path, depending on the operator's location, to maximize separation. Also, it is possible to change the robot's pre-programmed tasks dynamically. The authors consider that it is necessary to obtain reliable information about the position of humans, although they argue that in the period in which the article was produced, there were no sensors that could do this with the level of precision that the problem requires.

The work in Zanchettin et al. (2019) seeks to provide safety through a collision-free strategy based on collision avoidance. The objective is to avoid compromising production so that factory and machine operation is not interrupted in the event of an accident. This is achieved by generating new waypoints that a robot must follow; the intention is that the new path is optimal and requires the shortest possible time to avoid stops or deceleration in the presence of an obstacle. The solution uses a depth camera that obtains information from the environment to infer the robot's best decision. The strategy is validated using the UR5 robotic arm.

The authors of Magrini et al. (2020) present a framework that aims to ensure the safety of humans in a robotic cell where there can be human-robot interaction. The structure is based on the continuous identification of the human-robot distance through cameras with depth sensors, changing the robot's speed according to the proximity of any obstacle. It also uses non-contact collaborative operations based on identifying operator gestures mapped to commands for the robot.

Despite the benefits that the works presented bring, the proposed security systems for industry presented are specific solutions for their use cases, which means that they do not present flexibility for application in other contexts. In addition, they do not worry about scalability or integrating other sensors or CPS communication programs. For example, when one of the proposals addresses human-robot interaction, the solution only applies to controlling that single robot. An exciting proposal that this thesis seeks to make available is to have a reference architecture that is generalized enough to applications of safety systems in the industry. Therefore, the software made available by different authors could be executed in a pre-established structure. Thus, issues of information exchange, data structure adequacy, and sematic equalization to send commands to devices would not concern developers. This would allow for the seamless incorporation of software and devices into the risk prevention strategy.

## 3.2 MULTI AGENT VISION SYSTEMS

The information received by an application or decision agent may be interpreted as one of three things: either it is redundant information already known by the agent, it can be partially new, or totally new. Both possibilities contribute to the final diagnosis, albeit at different levels depending on the level of redundancy. As a baseline, redundant information often confirms and emphasizes the accuracy of events and data already received. There are cases where such redundancy can benefit applications, increasing trust and reliability ((BOSSE; ROY; GRENIER, 1996)). For example, cameras monitoring the same scene can transmit more perspectives simultaneously, thus eliminating the presence of blind spots and allowing the construction of a more comprehensive view. The study in (LI et al., 2022) adopts this approach in a smart city context. Similarly, the contributions of (RODRIGUES et al., 2021a) and (RODRIGUES et al., 2021b) consider a risk monitoring application in a Human-Robot Interaction (HRI) scenario.

The article in (ELMENREICH, 2002) relates the concept of sensor fusion, which suggests combining sensory data to extract better information than when interpreting the data individually. Intuitively, the more devices an app connects to, the more relevant control decisions it can reach. Several existing works have proposed efficient ways to collect, gather and process data in environments with many perception devices. Most of them mainly focus on camera selection approaches.

Some proposals consider a multi-robot vision approach, such as the works by Reily, Reardon e Zhang (2021) and Reily e Zhang (2020). Both resort to merging the perception of the multisensor environment to recognize a target object. They operate by allocating weights to the robot's point of view and selecting an optimal combination, that is, the views that offer the most informative observations. They use regularized optimization to identify the variety of views that best represent and recognize objects. Its applications seek to reduce the cognitive load that could be suffered by human operators, who need to view different videos at the same time. As a result, views judged to be less relevant are ignored and not shown to operators.

Other works consider the problem of choosing the best view in a multicamera network, where a single view is selected for processing. This reduces computational costs, as there is no need to process images from multiple cameras. This type of selection is applied in the context of recognizing human action. For instance, the work of Spurlock e Souvenir (2016) evaluates view change as a Markov decision process and applies reinforcement learning to support view selection. The study in Rudoy e Zelnik-Manor (2012) uses spatio-temporal shapes (like human silhouettes at different times) to select views where user actions are easier to recognize. The authors adopt a learning approach to carry out such recognition. A related research in Daniyal, Taj e Cavallaro (2010) adopted a policy that considers the frames and characteristics of the objects to make the multicamera selection. Classification is then made based on characteristics such as the number of objects, their size, and location. The work uses a Dynamic Bayesian Network (Murphy (2002)) responsible for choosing the best current view based on a previous one.

Camera selection in the context of target tracking for a multi-camera system remains challenging. This problem represents choosing the cameras with the best tracking of the object at a specific time as the object travels along a path. This selection is highly dynamic, as the camera with the most accurate point of view varies with time, while the object to be tracked moves through different locations. To deal with this problem, Morioka et al. (2017) developed a camera selection method using a fuzzy automaton based on the previously selected camera (previous camera state) and the object tracking level in each available camera. The tracking level is the position measurement error in the monitoring area. Their approach also uses a handover protocol responsible for passing control between selected cameras without considering centralized processing.

A scenario for person tracking using multiple cameras is suggested by (TESSENS et al., 2014). It deals with overlapping camera information. The authors claim that better tracking can be achieved by combining detection data from different camera nodes, basing their decision on only a subset of the total number of cameras, even under challenging circumstances of occlusions and limited fields of view. Its selection framework introduces a unifying approach that integrates different quality measures of vision into a criterion based on generalized information theory.

The hand-off is the process of switching to the camera that has the attention for processing at the current time. For example, this occurs when an object leaves the field of view of one camera and is seen in another. This control aspect has attracted the interest of many researchers. The work presented by Li e Bhanu (2011) suggests using various tracking algorithms based on different features. The authors merge information to obtain a better result; hence camera choice is made with more confidence. Such an approach outperforms a single tracker and presents a low computational overhead. Daniyal's video production study (DANIYAL; CAVALLARO, 2011) classifies image content from multiple cameras using a multivariate Gaussian distribution model. It seeks to reduce overhead when switching among cameras, maximizing the view of relevant information over time.

Unlike the presented works, our method has its evolution in its generalization of the application context and could be applied as an independent framework to run the previous approaches. We decouple the essential characteristics for camera selection and generalize them as agents. The adopted strategy incorporates the main functionalities, the sensors, the processing, an ensemble, and a attention mechanism. The used sensors can, though not necessarily, be cameras. Also, the processing step generalizes the use of different algorithms for each agent.

## 3.3 ARCHITECTURES WITH ATTENTION MECHANISMS

In this section, reference architectures that have an attention mechanism are presented. In this context, the Cognitive Model with Attention (CMA) proposed by Chen et al. (2019b) intends to control autonomous cars. The integrated attention mechanism supports the subsequent planning and control system of a vehicle's actions. The attention engine uses the current state collected by vehicular sensors, the previous states, and the data obtained in the internal learning process to select the most expressive information for the planning stage. The attention mechanism is based on Recurrent Neural Networks (RNNs) of the

Long Short-Term Memory (LSTM) type, which allows the sequential processing of data with long-term temporal dependence. Thus, the attention mechanism can work on the data received in a temporal sequence and learn from human behavior during prolonged driving.

The attention mechanism described by Ramirez-Pedraza et al. (2020) is part of an architecture under development, which was not named in the work. It assists in the obstacle detection process and aids autonomous navigation. The idea is that an agent endowed with such an attention mechanism can find a way out of a maze on its own. The planning and decision-making modules receive information filtered by the care system and generate responses based on the agent's current state. The attention module contains data referring to the relevant characteristics for detecting stimuli that help accomplish the current task. These stimuli are emphasized while irrelevant information is suppressed. Significant features become a map of relevance. Then a bump map is made using the information received from the visual input. Both maps are combined to relate common points that meet the task objective. This is used to guide the agent to the destination location.

CogVis (Deshpande et al. (2017)) is a cognitive architecture designed to control attention. It seeks to perform Change Detection, a process to detect visual changes in scenes. For this, similarities and dissimilarities in images obtained at subsequent times are evaluated. The attention engine not only directs the focus to the information requested by the knowledge base but also suggests places in the field of view that are drawing attention so that they can be processed further. It also has a sequential processor that works in continuous cycles and a parallel processor that continuously interprets, groups, and labels image objects.

The work Gao et al. (2019) specifies an architecture that focuses on the perception of the animated activities of agents (animals or people) in a scene, such as a walk or run action. It considers mechanisms of both attention and pre-attention. The capacity of the attention system is indicated by the number of hypotheses it can choose from. A hypothesis represents an interaction with another agent, such as chasing, fleeing, or helping. The function of attention is to process the chosen hypothesis and store the result in working memory. It does not detect a fixed movement pattern but infers it from previous experiments. Each focus of attention can randomly change its hypothesis based on the current internal state and the observation obtained by the agent. There can be multiple foci in the same hypothesis, which improves the accuracy of perception. Pre-attention processes possible focuses of attention in parallel. This occurs by detecting the direction of movement from the heat emanating from the objective. Each possible focus of attention has an associated probability; the greater the probability, the greater the chance of being selected by the attention system.

The ARCADIA (Adaptive, Reflective Cognition in an Attention-Driven) architecture, reported by Bridewell e Bello (2015), represents attention as a process that seeks to balance cognition and perceptions. It is used to perceive and track objects. The attention mechanism considers processing cycles where only one item is in focus in each cycle, and the one highlighted directs the behavior of the components in perception. Attention selection initially considers changes in preexisting objects, then whether there are new objects, based on data from the perception stage. Finally, new goals are proposed, which are selected at random if there is any doubt regarding which element to focus on.

The contribution described in Rea, Metta e Bartolozzi (2013) considers an attention system aimed at a humanoid robot, the iCub[1]. Such a system aims to select the region of interest with low latency, allowing the robot to determine where to fix its gaze quickly. This is achieved by processing events that indicate variations in the environment. These events are generated at a low level by the vision sensors. Once events are identified, visual information is sent to modules that extract information and generate feature maps. These maps are then unified into a single salience map, which is transmitted to the module that will direct attention to the region of greatest prominence.

The LIDA architecture (Franklin et al. (2014)) has an attention module responsible for bringing the content to consciousness. The attention system of this architecture is described in the work of Faghihi, McCall e Franklin (2012). This architecture implements the theory of the Global Workspace (BAARS, 1997), which considers that human cognition is implemented by a kind of microprocessor with specific applications. To enjoy global attention, these processors must compete or form coalitions, which is when processes come together to perform a specific cognitive task. In the adopted architecture, these microprocessors are called *codelets*. Among them, those directed to attention are called attentional *codelets*. Attentional *codelets* have the role of directing the content of a coalition to consciousness. The LIDA architecture also considers attention-oriented learning, which can happen in two ways. In the first one, called selective learning, the attention *codelets* that brought the winning coalition to consciousness are positively reinforced, thus having higher priority in the future. In the second approach, instructional learning, a new attention *codelet* is created for specific content to which attention can be directed.

The ASMO architecture (Lane, Gobet e Smith (2009)) was developed for use on two platforms, a teddy bear robot for human interaction and the Nao robot [2] applied to the soccer competition of robots[3]. In this architecture, the robot's cognition is composed of different processes that perform a single activity, which can be an internal skill or an action. Each process has an attention value that determines its processing priority; the higher the value, the higher its priority. The values are used to mediate the processes' access to robot resources. A process can change its attention value based on predefined

[1] http://www.icub.org/
[2] www.softbankrobotics.com/us/nao
[3] RoboCup Standard Platform League - All participants must use identical robots, in this case, the robot Nao. They are autonomous and must communicate with each other.

algorithms. However, it cannot change that of other processes, except in the case of the supervisor process. The latter is responsible for evaluating the priority of processes considering the agent's intentions and motivations and conflicting objectives between processes.

Table 2 gathers the works that were presented in this section. The works were coded according to the Table 1. It presents the purpose of applying each of the cognitive architectures and the role of the attention mechanism in the architecture. Each scenario of application of cognitive architectures and attention mechanisms has its specificities. Our work considers the context of risk prevention in the interaction between humans and robots in industry.

Table 1 – List with codes of works with attention mechanisms in cognitive architectures.

| Code | Work |
|---|---|
| A1 | Ramirez-Pedraza et al. (2020) |
| A2 | Chen et al. (2019b) |
| A3 | Deshpande et al. (2017) |
| A4 | Gao et al. (2019) |
| A5 | Bridewell e Bello (2015) |
| A6 | Rea, Metta e Bartolozzi (2013) |
| A7 | Faghihi, McCall e Franklin (2012) |
| A8 | Lane, Gobet e Smith (2009) |

Source: The author.

Table 2 – List with codes of the works with its architecture application and attention mechanism. The description of the work codes are in Table 1.

| Code | Architectures application | Attention mechanism |
|---|---|---|
| A1 | Obstacle detection for autonomous navigation | Emphasizes stimuli relevant to the current task |
| A2 | Car automation | Vehicle control, planning and actions. |
| A3 | Detection of visual changes in scenes | Directs attention to information that stands out |
| A4 | Perception of animated activities in scenes | Process prominent animated interactions in the image |
| A5 | Object perception and tracking | Directs perceptual focus |
| A6 | Humanoid robot control | Guides eye movement to lock in focus |
| A7 | Action selection and learning | Choose content to add to awareness |
| A8 | Robot soccer and human interaction | Sets the processor access priority |
| This | MAS-based for IIoT Safety | Relevance of distributed resources |

Source: The author.

## 3.4 IIOT

Human safety using IIoT networks is a relatively unexplored domain. In this area, much of the research is focused on the mining industry, such as the works of Zhou et al. (2017), Peng et al. (2019) and Prasad e Pillai (2018). The work in (ZHOU et al., 2017) specifically considers underground coal mines to relate the benefits and challenges of using the IIoT. It monitors and analyzes past and real-time atmospheric and terrestrial stability information. The system is applied in a use case that utilizes light and fan control to alert workers and ventilate the space in case of an emergency in the coal mine. The adopted IIoT network includes IoT sensors that consist of a micro-controller unit with WiFi communication and message exchange with the server based on the MQTT publish and subscribe protocols.

The study conducted in Peng et al. (2019) proposes a method to detect anomalies in sensor data and identify hazards for early safety alerts. The authors consider algorithms that process data received from one or several sources. The IIoT network considers two different network topologies. The first one is a star topology connecting wireless sensors to base stations. The second network configuration uses a chain topology connecting the base stations with the server by wired media (fiber optic). Similarly, the work in Prasad e Pillai (2018) also focuses on networking for mining industries. Its authors propose an IIoT network framework that allows sensor networks to communicate with a cloud server that hosts a monitoring unit. They consider using Arduino modules as processing units, and communication is based on Xbee modules that exchange information with a *nodemcu* gateway. The last node forwards the messages to an AWS (Amazon Web Service) cloud server using the MQTT protocol.

Different frameworks are also proposed for the IIoT context. The suggested framework in Reegu et al. (2020) is intended to provide efficient and resilient communication in critical disaster circumstances. The premise is that a network malfunction caused by intruder attacks or common device failure issues can cause accidents for people. It seeks to detect abnormal events on time, locate the event site and workforce, generate alerts for workers and emergency service providers, and guide humans to safe locations. Its authors consider a four-layer IIoT network: data acquisition, host computing system, cloud computing, and application.

The framework proposed by Mayer et al. (2017) uses ontology for automated reasoning and decision making. The user, who does not need to be an ontology expert, can insert safety actions into the processing workflow, using the new knowledge in decisions to mitigate possible dangers. The system is demonstrated in an app to advise on the best protective equipment in a risky situation. As a use case, the work presents a system that recommends the best protective equipment for use in risk situations.

The proposal of Sun et al. (2019) is a framework applied to Occupational Safety Health (OSH). It checks the air quality to prevent long-term health problems for workers. Its authors consider a Raspberry Pi as their network gateway that processes service requests. The framework discloses the information collected by the sensors in the network to different requesting services.

The approach proposed by Molka-Danielsen, Engelseth e Wang (2018) leverages big data analysis and data visualization techniques. Data is collected using automated Wireless Sensor Network (WSN) monitoring tools. The initial idea is to apply this system in supply chain and operations management for automated decisions. The system proposes recommendations for humans passing through that environment to lead them to the safest place and indicate how long workers could safely remain in that space.

The safety system can also be attached to the worker as a wearable device. Examples of works that embrace this idea include Hayek et al. (2017), Telawi, Hayek e Börcsök (2017) and Campero-Jurado et al. (2020). In the work from Hayek et al. (2017), the authors consider a System on Chip (SoC) to detect human movement patterns in the industry to avoid hazards like collision and recommend better routes for robots across the scene. The solution uses a centralized server connected to a WiFi network. The work demonstrates that movement patterns can be detected, including walking, running, or moving to the right or left. The study in Telawi, Hayek e Börcsök (2017) also creates an SoC aimed at industrial safety. The objective is to ensure secure communication in wireless networks. The system seeks to improve the response time and amount of processed data. It also aims to reduce repeated messages and data loss and prevents unwanted messages, wrong messages, data corruption, and delays.

Campero-Jurado et al. (2020) presents the development of a smart helmet to prevent risks to workers. It has many built-in sensors to infer hazards and notify people. Communication uses a wireless IIoT network to alert outside people of a risky situation. The work uses machine learning approaches such as Naive Bayes Classifier (NBC), Neural Networks (NN), and Convolutional Neural Networks (CNN).

Similar to Campero-Jurado et al. (2020), Kumar et al. (2020) also uses computational intelligence to automate safety assessment. The work is then evaluated in an industrial welding setting. Its authors apply CNN to identify the safety equipment that workers are wearing, such the helmet and gloves. A reasoning process decides whether they should shut down critical machines in a risky situation. For this, image processing algorithms of the data perceived by the cameras in the environment are executed in real-time.

The work proposed by Kwon e Kim (2019) describes a model based on association rules to predict accidents at construction sites. Its authors consider various types of accidents and establish a link between the type and the cause of the accident. Accidents can be caused by oxygen concentration, hydrogen sulfide (H2S) concentration, humidity, flame size, building vibration, and temperature. The types of accidents are related to fire, suffocation, collapse, and heat stroke. With the adopted solution, the authors claim that it is possible to identify accidents and alert workers. They consider an IIoT network with sensors that send data to a gateway that ultimately relays the information to a big data analytics server over the internet.

Some works put forward systems that use IoT devices to transmit frames on the network. For example, Filteau, Lee e Jung (2018) proposes a Raspberry Pi-based video streaming service that can stream to various mobile devices. This system streams to Android apps using a JSON structure to transport images. The system successfully supports the transmission of video streams to different devices, but the device's processing power affects the transfer time and Quality of Service (QoS). The work of Khan et al. (2021) applies edge devices to analyze real-time traffic flow data in an intelligent transport system. It is used to evaluate heterogeneous and congested traffic conditions.

The Mez system (George et al. (2020)) employs an on-demand video frame transmission from IoT camera nodes to an edge server. The application on the edge server can specify the upper limit of network latency and the lower limit of precision that the application can tolerate for transmitting the frames. Latency and accuracy limits are reached by modifying video frames, analyzing quality parameters such as resolution, color space, blurring or downsizing, with the possibility of removing artifacts and differentiating frames. The SlugCam architecture proposed by Abas, Obraczka e Miller (2018) uses camera agents that transfer information based on pre-implemented computer vision algorithms. It is a solar-powered camera agent that communicates over a wireless network. The user defines the algorithms to extract the necessary information from the images. Work in Zubal', Lojka e Zolotová (2016) proposes an algorithm to detect when a human is in a thermally hazardous area using thermal vision systems that are included in an IoT gateway and made available to other applications for real-time safety tracking.

The work in Xhafa, Kilic e Krause (2020) analyzed the use of throttling to meet real-time requirements when processing the IoT stream on an edge device. It evaluates network parameters using a Raspberry Pi device in the edge layer. Authors show that these devices cannot support processing an increasing number of streams, resulting in crashes due to overheating. A different system, proposed by Uma e Eswari (2021), aims to avoid car accidents. It uses multiple sensors connected to an edge device (Raspberry Pi) that monitor different parameters such as blinking, head nodding, yawning, alcohol consumption, and gas leakage. The data is transferred to a cloud, where machine learning algorithms are applied, and the driver's safety status is assessed.

There are many contact points between these previous works and the work discussed in this paper. However, the present work seeks to evaluate the impact of local processing in network load and discusses an architecture that can potentially scale to a large number of sensors that are selectively enabled over a network to improve the detection accuracy of events of interest. The architecture is also not restricted to specific use cases and offers a modular and re-configurable structure that allows users to quickly customize a system to change conditions or the addition and removal of new sensors and actuation devices.

## 3.5 DISCUSSION

When developing a solution for IIoT, the works list some difficulties. One of them is related to information security, privacy, and integrity, as reported by Zhou et al. (2017) and Robinson Robinson (2019). This is relevant since the transmitted information must reach the receiver without loss to allow timely processing. It is also vital that malicious people cannot invade the network or make some attacks such as DDoS. Robinson also lists the need to reduce reaction time as challenging since reaction time needs to be as short as possible in a risky situation. The work in Zhou et al. (2017) highlights the difficulties of implementing network interoperability and data analysis. In Campero-Jurado et al. (2020) the authors encountered problems related to the limitation of the processing of embedded systems and the accuracy that can impact the production and availability of the service.

Some studies, such as Prasad e Pillai (2018), Hayek et al. (2017) and (HAYEK et al., 2017) consider one of the possible evolutions of their work is to apply computational intelligence to sensor data to predict risks in industries with sufficient time ahead to react or recommend safety locations for people to move towards them.

In Hayek et al. (2017), Kumar et al. (2020) and Telawi, Hayek e Börcsök (2017), the authors point to the need to reduce the size of the embedded system, as well as to know the requirements of software restrictions to run on the limited IoT devices and to communicate through their networks. The authors in Telawi, Hayek e Börcsök (2017) also seek to provide wireless safety communication and improve the message exchange protocol to handle sensors on moving objects such as robots and vehicles. Other works such as Campero-Jurado et al. (2020) and Prasad e Pillai (2018) plan to integrate their solution with cloud, fog, or edge computing.

Some of these intended future evolution plans found in the reported research are considered in this work proposal. Among these is the processing on edge devices with constrained performance and consistent communication with high-end servers. A second goal is the adoption of a multilayer edge-fog-cloud infrastructure and the concern to avoid overloading the communication network.

The present work seeks to evaluate the impact of local processing on the network load and discusses an architecture that can potentially scale to a large number of selectively enabled sensors in a network to improve the detection accuracy of events of interest. Note that the developed architecture is not restricted to specific use cases and offers a modular, reconfigurable structure that allows users to quickly customize a system to change its operating conditions or add and remove new sensors and actuation devices.

# 4 THE PROPOSED ARCHITECTURE

This chapter specifies the reference architecture designed as part of this thesis work. Initially, it discusses the project requirements. Next, it gives an overview of the proposal, followed by the modules that comprise the proposed architecture. The present chapter finalizes with a presentation of the steps adopted for the implementation of this architecture.

## 4.1 REQUIREMENTS

The need for an efficient approach to dealing with the interaction between devices in the industry drives the design of this architecture. Such devices exchange information using the network; hence it must provide mechanisms to interact with CPS devices, reduce and balance the load on the IIoT. As the research's primary use case is the industry's worker safety, ideally, there must be no delays when triggering an alert or issuing a command to a machine. The more available time an application has to react to a possible accident situation, the better it is for it.

One way to avoid delays and maintain a sustainable network flow is to reduce bottlenecks. This can, for example, be achieved by preventing bandwidth-intensive devices, such as cameras, from transmitting all the time. These devices consume considerable communication bandwidth resources even with the data they stream may have limited or no contribution towards risk detection and making a final decision.

It is the view of this work that the proposed architecture must support a mechanism by which it prioritizes information sources according to the relevancy of their data and their contribution to the diagnosis of a risk situation that may currently be occurring. In other words, there is a need for an intelligent mechanism that selects at any time among the agents (processing video streams) in an attempt to both reduce network traffic and speed up the detection of safety risks.

Note that such optimization cannot be made to the detriment of the accuracy of the diagnosis. As a result, unlike previous architectures that centralize the risk identification, the currently proposed one distributes intelligent processing among a number of agents and adaptively orchestrates their operation. The difference between the outputs when applying the distributed architecture and the traditional centralized solution should not be significant to maintain accuracy. A fast system is considered useless when it mainly returns incorrect. These concerns are particularly heightened in the present context of industry physical safety applications. It is important for the adopted solutions not to ignore risk alerts, act upon them in a timely manner and avoid having false positives.

When working in an IIoT CPS environment, different devices (software, machines, sensors, or actuators) need to work together to achieve an intended goal. Therefore, a scalable system must be considered capable of incorporating devices and software with different processing capabilities, intensities, data structures, and applications. Such a system must harmonize the agents' semantics in the heterogeneous environment to ensure that its modules can work together.

The proposed reference architecture must allow distributed agents to be configured independently of the framework's programming language. Reconfigurability and scalability are essential system characteristics. It must be flexible so that it cannot restrict the support for different application scenarios nor deny users to adapt and tailor a given solution to their needs.

In short, we can relate the following list of requirements:

- **R1 (CPS interaction):** the architecture must deal with both software and devices in the environment. It must handle the conversion of physical processing frequency, data structure and commands to cyber information. Especially those sensors and software used to identify industry hazards such as cameras, presence and smoke sensors.
- **R2 (Semantic equalization):** data from different sources, such as software and heterogeneous devices, must be combined in a unified semantics and frequency response to allow decision making considering the available variables.
- **R3 (Human and machine understandable vocabulary):** the communication language between the architecture modules must follow a language the human can understand to maintain and reconfigure, and the machine can interpret to execute the modules.
- **R4 (Reconfigurability):** reconfigurable framework allows users to quickly customize the safety system's behavior for the online application.
- **R5 (Flexibility):** represents the ability to support the application in environments with different network infrastructures and machines without needing changes in architecture.
- **R6 (Scalability):** represents the support for adding more devices, which may have different communication forms, data structures, and processing capacities.
- **R7 (Attention mechanism):** the architecture must include a method to prioritize the most informational distributed agents for multiperspective decision-making while reducing the priority of the others.
- **R8 (Decision-making):** the agents that execute the architecture implementation in different devices in the network infrastructure must make local decisions considering the rules made available by the system users.

- **R9 (Network performance):** the proposed architecture must avoid network traffic when there is no relevant data for the decision that the system seeks to make.

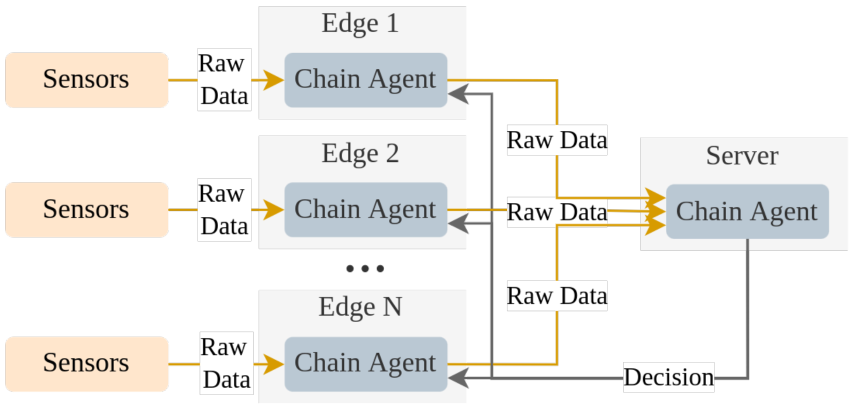
- **R10 (Accuracy):** the accuracy of the system that applies the architecture should not have a significant impact when compared to not using it.

As a result, the proposed architecture mainly consists of a set of intelligent agents that cooperate towards achieving a specific task such as safety in an IIoT environment. This multi-agent system continuously extracts and combines knowledge from its agents. Structuring the proposed reference architecture as a multi-agent system has its advantages. It ensures system scalability, avoiding a single point of failure as in the case of centralized systems. Furthermore, system organization offers more clarity, given its modularity, as reported by REIS (2003).

## 4.2 OVERVIEW

Fig. 7 presents an overview of the adopted architecture. It assumes that the used sensors have their computing infrastructure or can be connected and associated with an edge computing device. The architecture also assumes the availability of an underlying network infrastructure that provides computing resources with high computational capacity, such as in a conventional cloud/edge configuration.

Figure 7 – High-level visualization of the Architecture for multiple edge devices.



Source: The author.

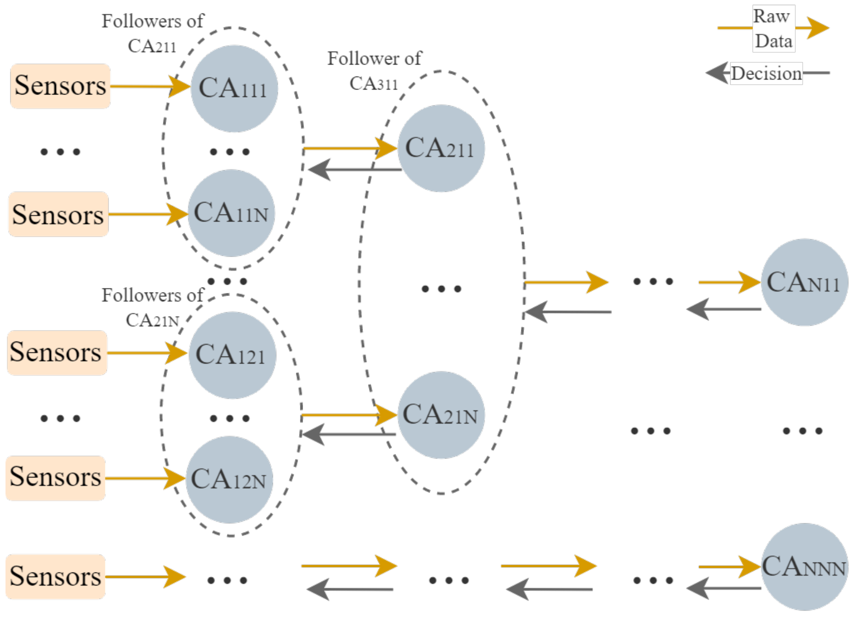
The Chain agent in Fig. 7 is the main active entity in the architecture. Chain agents can be located at the edge or cloud levels and are responsible for processing relevant system information. Edge chain agents pre-process and evaluate raw data collected directly from one or several sensors and decide whether said data must be forwarded up the chain for a more accurate evaluation. At a later stage, the Cloud chain agent will perform more thorough data analysis and make a decision regarding the current state of the system being monitored. Such decisions at a later stage may include: acting upon the system, asking additional edge agents for data, and informing edge agents that decisions have been taken and their data is no longer needed, among others.

The general idea here is that the edge agents will act as the system's first responders providing a continuous, local and quick evaluation of a developing system situation. Therefore, the edge layer acts mainly as an adaptive filter limiting potentially unnecessary network traffic.

Although the primary use case adopted in this thesis, later detailed, explores risk identification in an industrial environment, the architecture applies to many other scenarios where network utilization needs to be balanced in the presence of sensors requiring high network bandwidth. In this context, chain agents, their processing, and actions are configured through description files and use specific interaction interfaces, enabling the creation of more complex decision chains.

Holonic aggregation (MELLA, 2009) of chain agents is, therefore, possible as depicted in Fig. 8 providing additional flexibility possibly needed in other application domains. Coordination between agents follows a leader-follower structure. CA is the abbreviation for Chain Agent. Followers send raw data only to their leader (yellow arrows in Fig. 8), and leaders send decisions to all their followers gray arrows in Fig. 8). Followers have only one leader; however, leaders can have one or more followers. In $CA_{xyz}$, the $x$ variable represents the agent level in the chain. The level is the proximity of the agent to the sensors in the environment, where $x = 1$ is the lowest agent level. $x = 0$ represents the sensors level. An agent that is the leader of another agent is one level above this agent. Therefore, the leader receives raw data from the follower at a lower level and returns the decisions to that agent. In this case, the agent at level $x = N$ receives data directly from agents at level $x = N - 1$. $y$ is a number to represent the group of all agents at the same level $x$ that follow the same leader at the level $x + 1$. $z$ is the individual agent number in the group $y$ and level $x$. The decisions sent by the leaders are prioritized and subscribe to the agent's decision. Thus, agents' decisions at higher levels are propagated to lower-level agents.

Figure 8 – Architecture multilevel chaining.

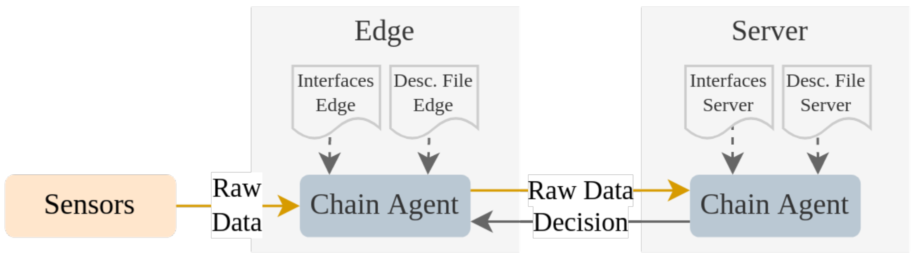


Source: The author.

## 4.3 ARCHITECTURE MODULES

Fig. 9 offers a high-level visualization of the chain agent and shows its associated parameter configuration files necessary for its execution, the description file, and the interfaces. The internal architecture of chain agents is depicted in Fig. 10 and is composed of the following modules: Interfaces, Rx Window, Ensemble, Decision, and Attention. The description file configures the agent's behavior and particularly its modules.

Figure 9 – High-level visualization of the architecture.

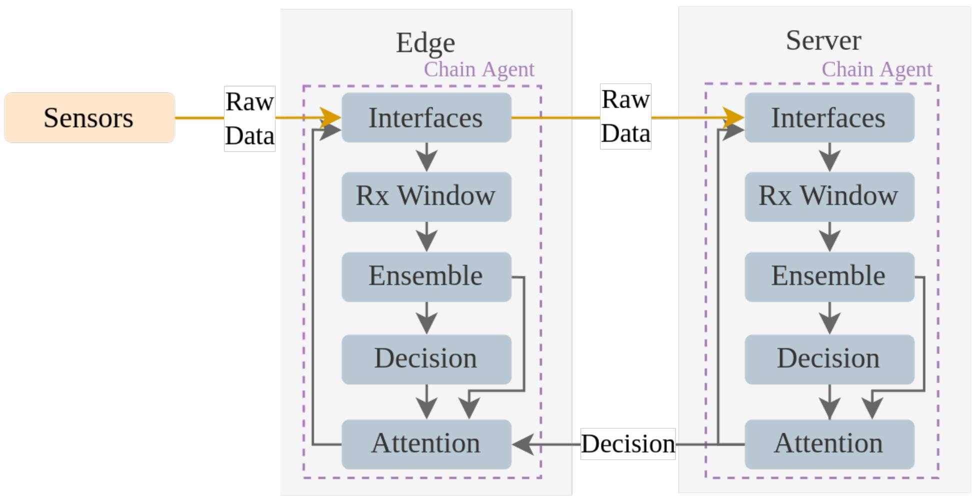


Source: The author.

The chain agent controls the execution flow of the internal modules in Fig. 10. The arrow represents a memory variable passing the output from one module to another. The interfaces deal directly with sensors, requiring different interfaces for different sensors. This means that the chain agent interacts through one or more interfaces.

A simple scenario is used in a didactic manner to ease the understanding of the role and interactions among the architectural modules. The considered IIoT scenario is straightforward. It consists of a switch and a lamp. It works as follows: when the switch power

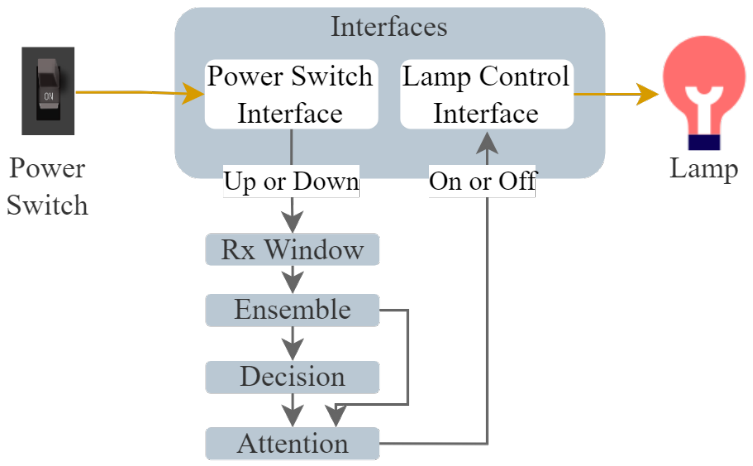
Figure 10 – Reference architecture internal modules interaction.



Source: The author.

button is up, the lamp needs to light on, but the lamp needs to be off if the power button is set down. Fig. 11 illustrates the architecture applied to the example.

Figure 11 – The architecture applied to the power switch and lamp example.



Source: The author.

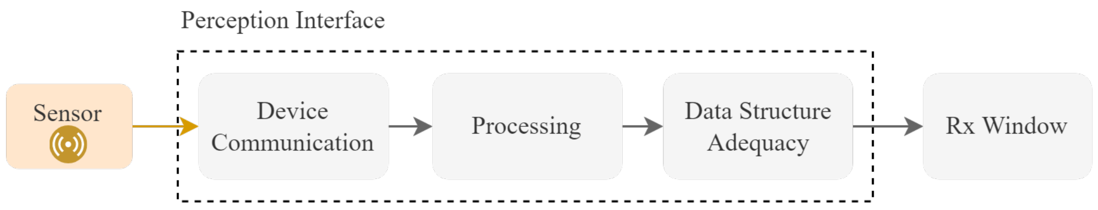
### 4.3.1 Interfaces

A well-designed IIoT system must be capable of processing and understanding the information that devices and sensors in the environment report (Requeriment R1). Also, computers must be able to represent this knowledge internally to produce a response to the stimuli received from the monitored IIoT setup. Knowledge representation formalizes the information so that other agents semantically understand what is happening around them. The system can later use the acquired knowledge to make relevant decisions causing to react to external events in a smart manner. According to Brachman e Levesque (2004), knowledge representation is the field of study concerned with using formal symbols to represent a collection of propositions believed by some putative agent.

*Interfaces* are software integration mechanisms that allow the seamless inclusion of new sensors and actuators in the system at hand, complying with requirement 1. The integration code for the new devices must implement a specific software interface defined by the user. This allows you to integrate new software without architectural changes, which meets the R5 requirement. Through such an interface, the raw data from the sensor becomes harmonized with the chain agents' information processing and execution semantics (requirement R2). In the case of actuators, system actuation information is converted to a device's native input commands. Interfaces must be created only once for each new sensor or actuator before insertion into the architecture. The interface module then channels the data internally to other modules, forwards it further up the chain, or actuates on the system. Even for the same device, interfaces may expose/make available different variables of interest for the system. This favors scalability (requirement R6). Observe that one interface can handle different sensors or actuators but may produce one single output to the agent. For example, a single interface is implemented to receive data from two power switches; the interface must send the agent a piece of single information choosing whether it is active or inactive in a certain period. Even if one is up and the other is down, a choice has to be made by the interface. Therefore from the agent's point of view, there is only one sensor. For the agent to treat the two sensors differently, each one must have its interface.

It is possible to classify interfaces into the two classes *perception* or *action*. *Perception interfaces* receive data from sensors, process it, translate it to the agent's semantics representation, and transmit it to such agent, as illustrated in Fig. 12. On the other hand, the *action interfaces* make reverse processes. They first receive and interpret data from the agent, process it, and optionally talk to an effector in the environment. An effector is seen as an object that causes change to its environment by acting upon it. Fig. 13 illustrates the action process. Note that the adopted architecture allows a single interface to act both as a *perception* and *action interface*. In such a case, it would receive and send data to the Chain Agent.

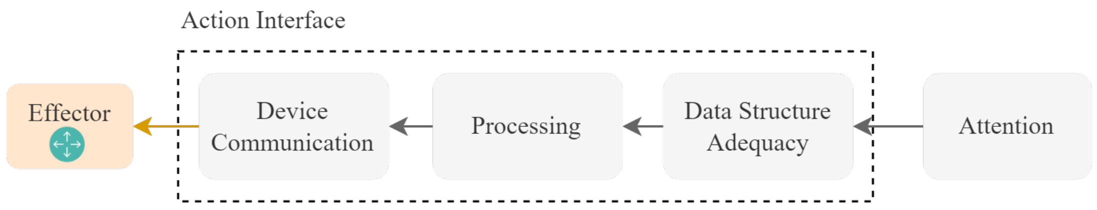
Figure 12 – Perception interface.



Source: The author.

The description of the processing steps present in Figs. 12 and 13 follows:

Figure 13 – Action interface.



Source: The author.

- *Device communication:* a service that talks directly with sensor or effectors, implementing library and functions responsible for the reading data or sending commands.
- *Processing:* a service that receives the data already read from the sensor and processes it to produce relevant information for the agent. To achieve this, it may use machine learning methods or some inference specific to the user application.
- *Data Structure Adequacy:* converts data to the format that the agent understands (at the perception interface) or interprets the agent's data (at the action interface).

At a more concrete level, interfaces are scripts generally implemented using the same programming language as the one used to implement the architecture. Therefore, they are part of the initial execution of the agent, operating seamlessly. The *Interfaces* module links two queues to each new interface, which are used for exchanging messages between the interfaces and the chain agent. One queue is for use by the interfaces to receive data from the agent, and another allows it to send data to the chain agent.

The interfaces forward perception messages from the sensors to the Rx Window module (described later in Section 4.3.2) of the chain agent, be it the edge device or server, as shown in Fig. 10. In this way, the output of the interfaces must follow the structure of Listing 1, representing the semantics that the chain agent understands for receiving messages. Such messages must be placed in the txQueue. The *id* is a string representing the identification of the perception interface. The chain agent uses the *id* to differentiate among interfaces. In particular, it is used by the *Ensemble module*, which is responsible for merging the information from different interfaces. The *Ensemble module* is discussed in more details in Section 4.3.3). When there are two perceptions with the same *id*, the *Ensemble module* considers both as a single module. To be considered independently, they must be defined with different *ids*. The second property of the Listing 1 is the *perceptionvar*, which is also a string that indicates a label representing the values produced by the interface. The *value* property is the measurement obtained by the interface. This value follows a nominal variable, whose interpretation is limited to the semantics the system understands to apply the rules, as defined in the description file for perception (Section 4.3.6). The architecture does not propose calculations using numeric data, which excludes the use of ordinal variables and types such as int and float.

Listing 1 – JSON Schema for the interface perception output.

```
{
    "type": "object",
    "properties": {
        "id"            : "string",
        "perceptionvar": "string",
        "value"         : "string"
    }
}
```

The Listing 2 shows the format of messages sent from the agent to the interfaces, using rxQueue. The *actionvar* is the variable type of the *value* field, and the *value* is the result sent to the interfaces. Different interfaces can use the same action variable from the agent. Action interfaces do not need an *id* parameter because the interface is responsible for further processing or connecting to different effectors based on the *actionvar*. The possible values in the fields *perceptionvar*, *actionvar* and *value* are restricted to what is defined in the description file (Section 4.3.6).

Listing 2 – JSON Schema for the interface action input.

```
{
    "type": "object",
    "properties": {
        "actionvar": "string",
        "value"    : "string"
    }
}
```

In the case of the previously introduced power switch and light example, *powerSwitch-Interface* is the name of the perception interface. In the *Device Communication* stage, the module reads the electrical impulse from the power switch. It converts the impulse to a number between zero and one, representing the position of the switch. This value is then forwarded to the *Processing* stage, which implements a simple threshold-based test. The key is considered "on" for numbers greater than 0.5 and "off" otherwise. Finally, *Data Structure Adequacy* stage converts the Processing response to the agent's semantic structure. The example message is in Listing 3, which follows the JSON schema established in Listing 1.

Listing 3 – Interface perception output to communicate with the chain agent for the example case of a power switch and lamp.

```
{
    "id"           : "ID0-powerswitch",
    "perceptionvar": "powerswitch",
    "value"        : "up"
}
```

The action interface is responsible for controlling the lamp; for the example, it has therefore been called the *lampControlInterface*. The *Data Structure Adequacy* phase within the action interface receives a message from the agents, such as the example given in Listing 4 (it is in adequacy with the JSON Schema in Listing 2). After receiving the data, the *Data Structure Adequacy* phase sends it to the *Processing* step. The *Processing* phase implements a function that converts "on" decisions to the number one and "off" to zero.

The *Processing* stage then sends the result of its mapping to the *Device communication* stage, which converts the number one or zero into an electrical signal to turn the lamp on or off respectively.

Listing 4 – Interface action input to receive commands from the Chain agent for the example case of a power switch and lamp.

```
{
    "actionvar": "lamp",
    "value"    : "on"
}
```

### 4.3.2 Rx Window

The *Rx window module* is responsible for processing the data generated by different sensors and reaching the chain agent through the different *Interfaces* modules. Different sensors can transmit data at distinct throughput rates. For example, in a sensor outputting 10 messages each second, the *Rx window module* guarantees that every 0.1 seconds one data from the sensor is processed. As earlier mentioned, this module manages all the available interfaces. By defining the size of the data reception window, it is possible to control the frequency at which data is forwarded and analyzed by the subsequent modules in the chain agents' internal architecture. The reception window is configurable; hence, the user can decide, given the sensors available in the system, how much time the system should be accumulating data from the different sensors and then forward them at once. Due to different throughput rates, different sensors buffer a distinct number of data in the same reception window. Agents may be configured to stabilize at a single frequency the processing rate of the events received from sensors by using the right frequency with each sensor. Note that it is important that they synchronize the data they receive in time. This is important to ensure that data is collected in the right order and processed in the correct time sequence. This way, requirements R1 and R2 regarding the standardization of frequencies are met.

The window period is defined in advance in the description file (Section 4.3.6). After each window interval, the module forwards all received data to the *Ensemble*. The larger the window, the more there is a need for memory resources to gather all the data received, and consequently, the lower the response frequency. The smaller the window, the less amount of memory is required and the higher the response frequency becomes. However, if the reception window is shorter than the time it takes an interface to send a measurement to the agent, the agent can reach some decisions without considering any of the data from this interface.

The input for this module comes from *Interfaces*, so it follows the same JSON schema as the interface's perception output (Listings 1 and 3), as this represents the data structure for communication between these modules.

In the power switch and lamp example, if the *Power Switch Interface* (Fig. 11) outputs data at 10 Hz and the *Rx window module* receives this data at 5 Hz, then in each time window, two power switch data items are passed to the *Ensemble module* of the agent. The latter is detailed next.

### 4.3.3 Ensemble

The literature has been discussing Ensemble methods for some time now (Zhou (2012)). In fact, several benefits have been raised stemming from their use. These are mainly found in the field of machine learning applications such as object detection, recognition, tracking, etc. They are used to combine weak learners into a unified, more robust learner. Mainly, they are used to improve both the speed of algorithms and their accuracy (Avidan (2007) and Giacinto, Roli e Didaci (2003)). Different methods can provide ensembled inputs. They include averaging, voting, and combining by learning. The choice of assembly technique depends primarily on the application and the required accuracy and response time levels. The proposed architecture leverages this concept by allowing the application of ensembles in a network context where different models can be distributed across the IIoT and propagate their partial or final findings to a central server that assembles the received input messages into single output information.

In the proposed architecture, the *Ensemble module* is responsible for summarizing all the information received within a given data reception window to attend requirements R2 and R6. The presence of multiple sensors with different data transfer rates can give rise to conflicting information. Therefore, the purpose of this module is to decide on what is, most likely, the status of the system, given the data received. In addition to using it to unify responses, the ensemble can potentially produce better accuracy (requirement R10) in cases where there are perceptions that take different perspectives of the monitored environment.

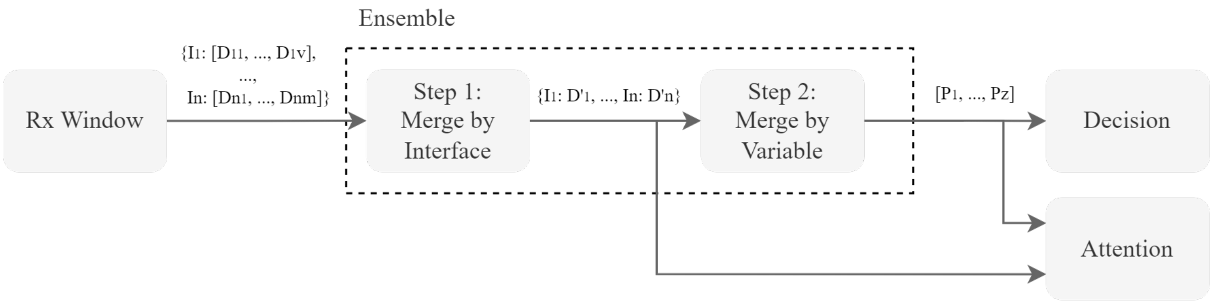
A case in point is a multi-camera system that analyzes IIoT scenes. The ensemble module must merge the received observations and suggestions into a single valid video perspective.

The *Ensemble module* internally adopts a two-step approach as depicted in Fig. 14. First, it merges the amount of information received by each interface. Different interfaces may have buffered a different amount of data messages reported during the same receive window (Rx window). Therefore, $I_i$ is a list of all data sent by the interface $i$ ($n$ is the total number of interfaces, so $1 \leq i \leq n$). $D_{ij}$ is the $j$-th data collected by the interface $i$. $D_{ij}$ follows the structure already presented in the Listing 1). After Step 1, each interface has only one data represented as $D'_i$, which also follows the Listing 1. In the Step 2, the *Ensemble module* combines the information received over all the different interfaces so that the perception variables (Section 4.3.1) are unified. The result is a list $[P_1, ..., P_z]$ where $P_z$ is the $z - th$ perception variable in the set (in other words, there are $z$ perception variables). Each $P$ component in the list describes a perception variable and its value as strings. $P$ has the structure in Listing 5. Listing 5 only differs from the Listing 1 in removing the "id", because the interfaces were merged. The output of Step 2 is a single value for each variable. The merged information is then forwarded to the *Decision* block for further processing.

Listing 5 – JSON Schema for each component of the ensemble output list.

```
{
    "type": "object",
    "properties": {
        "perceptionvar": "string",
        "value"        : "string"
    }
}
```

Figure 14 – Ensemble steps.



Source: The author.

The first step in this *Ensemble module* considers the received values as literals and uses majority voting for fusing the data from the same interfaces. As part of its second processing step, responsible for merging information from different interfaces, the implemented ensemble agent uses unanimity voting. Recall that the use of identifiers (or *id*) by interfaces makes it unique to the ensemble, so if two interfaces have the same id, they are considered the same one from the ensemble's point of view.

There are cases where a different ensemble strategy may be required. For instance, an ensemble module may adopt a majority vote as part of its second step. This usually depends on the application requirements and the type of perception variable. For example, if the risk is a perception variable, the edge chain agent can only claim that there is no risk if all the interfaces unanimously vote for no risk; otherwise, the second stage evaluation defaults to the presence of risk. The ensemble agent fuses the information based on the variables made available by the interfaces. Note that it only guarantees that information from the different sources is merged if different interfaces provide data related to the same variable. In other words, the ensemble's output is the fused value for all the variables exposed by the interfaces.

The ensemble agent receives as its input a list of all data received by the *Rx window module* during the configured window. Each data item in the list has the format and follows the schema explained in the Listing 1. According to our architecture, it produces two outputs, one for consumption by the *Decision module* and another one destined for use by the *Attention module*. The first output that is fed into the *Decision module* represents the ensemble's final result after its second step's execution. More specifically, the ensemble module provides the *Decision module* with a list in which each component has the structure defined according to the Listing in 6. This structure is similar to the Listing 1, but excludes the *id* parameter, as different interfaces with different ids have been ensembled in this module.

Listing 6 – JSON Schema for the ensemble output to the decision module.

```
{
    "type": "object",
    "additionalProperties": {
        "type": "array"",
        "items": {
            "type": "object"
            "properties": {
                "perceptionvar": "string",
                "value"        : "string"
            }
        }
    }
}
```

The ensemble's output destined to the *Attention module* adopts the information scheme described in Listing 7. The *input* parameter of this structure is a list with the result for the first step of the ensemble, that is, one that transforms many inputs into one single input per interface. Hence each list component represents one value related to a given interface. The *output* parameter receives the same output sent to the *Decision module*. Its use is better explored in the Section 4.3.5.

Listing 7 – JSON Schema for the ensemble output to the attention module.

```
{
    "type": "object",
    "properties": {
        "input": {
            "type": "object",
            "additionalProperties": {
                "type": "array"",
                "items": {
                    "type": "object"
                    "properties": {
                        "id"           : "string",
                        "perceptionvar": "string",
                        "value"        : "string"
                    }
                }
            }
        }
        "output": {
           "type": "object",
            "additionalProperties": {
                "type": "array"",
                "items": {
                    "type": "object"
                    "properties": {
                        "perceptionvar": "string",
                        "value"        : "string"
                        }
                    }
                }
            }
        }
}
```

Let us assume that, in the case of the power switch and light bulb example, there are two interfaces with different ids that send 3 messages in each receive window. The first interface sends the 3 observations "up", "up", "down", whereas the second interface reports the values "up", "down", "down". As part of the execution of the first ensemble step, the *Ensemble* module applies the majority vote to each different interface. Consequently, data from the first interface is summarized as "up" whereas data received by the second interface is reduced to the "down" output. In the second step of the *Ensemble* module, the latter merges the data from both interfaces. In this example, it applies the unanimous vote. However, as the first interface signals an "up" state and the second one reports a "down" state, there is no unanimity. In this case, the module considers a default value, which is the first variable value defined in the perceptions component of the description file. For example, taking "up" as the default value, then "up" is the adopted output. Therefore, there is only one output for the power variable. The Listing 8 presents the output for the *Decision module* and the Listing 9 shows the output for the *Attention module* in the present example.

Listing 8 – Ensemble output to the Decision module for the example case of a power switch and lamp.

```
[{
    "perceptionvar": "powerswitch",
    "value"      : "up"
}]
```

Listing 9 – Ensemble output to the Attention module for the example case of a power switch and lamp.

```
{
    "input" :[{"id"            : "ID0-powerswitch",
               "perceptionvar": "powerswitch",
               "value"        : "up"},
              {"id"            : "ID1-powerswitch",
               "perceptionvar": "powerswitch",
               "value"        : "down"}]
    "output": [{"perceptionvar": "powerswitch",
                "value"      : "up"}]
}
```

### 4.3.4 Decision

The *Decision module* receives the response from the *Ensemble* and applies user-defined decision rules (requirement R8). Such rules are specified in the description file (Section 4.3.6). The outcome of the decision module is a set of user-specified actions. Therefore, a user or safety manager responsible for industry safety could elaborate or propose the rules coded according to a human-friendly way. The idea is that they can define these rules next to the verbal vocabulary of if condition, then do something. For example, if someone is without protective eyewear, then activate a sound alarm. Specialists may extract from their experience rules and update these with time. Care must be taken in order to avoid conflicts. Some simple priority schemes may be used to solve possible conflicts among the rules. The resulting local actions can only be applied to the system if a higher-order chain agent (in the case of Figure 10, the agent on the server is higher than the edge) does not create an overriding decision. The decision from higher-order agents may invalidate the local one. The *Attention module* manages which action ends up taking effect in the system.

The behavior of actuating devices in the environment is related to industry rules, safety protocols, and the experience of the safety team. It can completely change from factory to factory. Applying machine learning techniques to this may not be practical as we could have significant variability in terms of risk requirements depending on the environment. In addition, it is not a common practice in industries to keep a record of accident data and the frequency of their occurrence. Also, note that it will take a long time to build a sizeable dataset with events of interest where safety has been compromised. This makes it difficult for some algorithms to generalize their observations and decisions while maintaining reasonable precision. With regard to our architecture, the application of learning models must be implemented as part of the *Interface modules*.

The *Decision module* processes output information from the *Ensemble* module. In fact, it interprets the rules described in its description file, combines these with received input from the *Ensemble* module, and triggers the rules that finally generate some results. The output is then sent to the *Attention module*.

The used rules follow an if-then structure. The "if" can have one or more conditions to trigger the "then" side. In case of all conditions are met, the values declared in the "then" side is output. A condition is simply a variable that must take on a specific value to be met. For example, a variable named "has protective eyewear" must be "true" to return in the "then" side the variable named "alert" with the value "on".

Listing 6 represents the input structures of the *Decision* module, and the output is in the Listing 2. For instance, when the *Decision* module receives the input: {"var": "powerswitch", "value": "up"} from the *Ensemble*, it returns {"var": "lamp", "value" : "on"} to the *Attention* module. Otherwise, when it receives the input: {"var": "powerswitch", "value" : "down"} from the *Ensemble* module, it returns {"var": "lamp", "value" : "off"} to the *Attention* module.

### 4.3.5 Attention module

The *Attention* module orchestrates the use of decisions made by the local device as well as those coming from a chain agent located at a higher position in the hierarchy (requirement

R7). The *Attention* module can apply orchestration in two different modes, called Active or Passive attention. Active attention is a leader-follower structure, where the leader's decisions are sent to the follower's agents, but since the follower agents also make their own decision, the leader's decision always takes priority. Simply put, active attention is a switch that chooses between local decisions or those from the leader. In such a case, every time a leader sends a message, it overrides the local decisions. The leader's decision remains for the period defined in the reception window (Section 4.3.2). If no other message from the leader arrives, the local decision is resumed. On the other hand, passive attention chooses the decision of the agent it considers to be the most relevant one at the moment among all the follower agents sending information. This can be used to prevent irrelevant agents from transmitting on the network, favoring the requirement R9. However, accuracy must be observed (requirement R10). The process for assessing such relevance in passive attention is explained in the following paragraphs of this section.

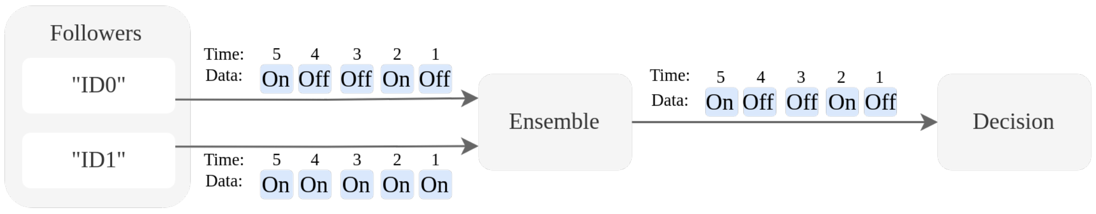
Interfaces and follower agents can be considered synonymous from the point of view of the attention module. Because although an interface is responsible for reading and interpreting the data of a follower agent, in the end, the main factor that describes both and differentiates two agents is the *id*. Interfaces and follower agents will be treated from now on in this section as followers only.

The output from the *Ensemble* and *Decision* modules is fed as input to the *Attention* module. The latter returns feedback to the followers. This feedback recommends whether the followers should process and send data in subsequent time intervals. The time interval, in this context, refers to a period between two sequential output events that is defined by the receive window (Section 4.3.2). This avoids the waste of computational and communication resources, which occurs when an agent has to continuously process data, often with a limited contribution to the *Ensemble* module. The objective is that the *Ensemble* output when applying the passive attention is similar to its output without the attention. In this case, the attention application should transmit less information between agents distributed over different hosts, while it cannot significantly impact reducing accuracy.

The passive attention checks the relevance of each follower and then decides which one can continue transmitting or if it has to stop transmitting for a period called sleep time. Therefore, passive attention has two steps, the first step is *Relevance Checking* and the second one is *Pertinence Decision*. *Relevance Checking* defines the relevance of each follower. It achieves this by comparing the information the followers sent with the output of the *Ensemble* module. The output of *Ensemble* is part of the "output" component of the data sent by the *Ensemble* to the *Attention* module, and the data of each follower is in the "input" component, as described in the Listing 7. The *Attention* module has a first-in, first-out (FIFO) queue which, by default, stores five messages. The queue size is adaptable. At the end of each period defined in the receive window (Section 4.3.2), new data is inserted into the queue, and the oldest data is removed from it. The *Attention* module uses the queue to calculate the accuracy of each agent. It compares the queues of each follower with the ensemble output and counts the number of hits. The number of hits is then divided by the queue size. With this, each follower obtains an accuracy value, representing the relevance of that specific agent.

To exemplify the above mechanism, Fig. 15 illustrates a case considering two followers with the same variable *perceptionvar* as output. Both intend to identify whether the power switch is up or down. Therefore, they essentially perform the same activity, albeit with the use of different algorithms. As a result, they may produce different outcomes during the same time interval. They both send their findings to an arbitrary *Ensemble* module, which combines these and returns the response as shown in Fig. 15. Note that this module does not necessarily adopt the same unanimity vote ensemble used in our architecture. The image illustrates the transmission of five sequential messages (between time instants 1 to 5 in Fig. 15). In this case, the queue size is 5. At time 1, the *Ensemble* module produced the first response (Off), from the messages sent at time 1 by followers ID0" and "ID1" ("ID0" sent Off and "ID1" sent On in time 1).

Figure 15 – Example of Ensemble for two different agents followers.



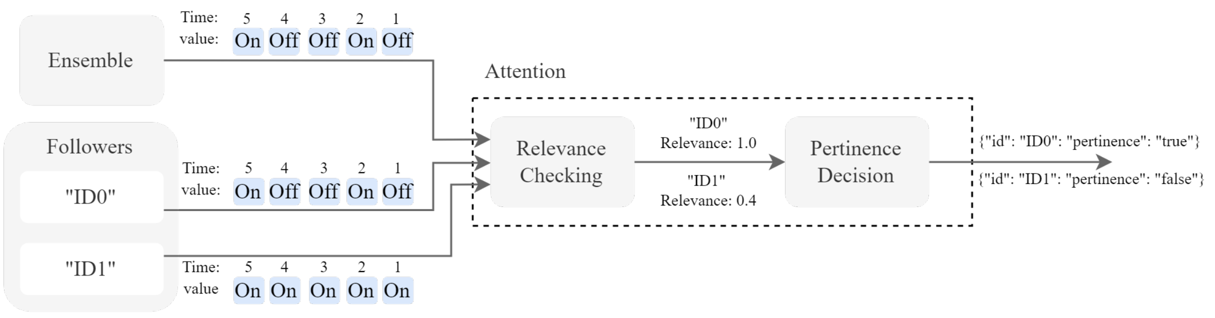
Source: The author.

Fig. 16 illustrates how the *Attention* module uses data from the *Ensemble* module to reach a decision. The *Relevance Checking* step has a queue with five received messages. It compares the follower value with the *Ensemble* module at all times, which results in the precision that is the relevance of that follower. Fig. 16 shows that follower "ID0" has 100% relevance, while follower "ID1" has 40% relevance. This means that information from "ID0" is seen as being more representative as all its data matches the *Ensemble* answer. On the other hand, follower "ID1" only managed a correct output in times 2 and 5, that is, 2 cases out of 5, resulting in a 0.4 relevance level.

After the execution of the *Relevance Checking* step, comes the *Pertinence Decision* step. It considers the relevance of each agent and assesses whether it is above a predefined threshold. If yes, it recommends that this agent remains active. Otherwise, it suggests disabling the agent for a given amount of time. Assuming that in the case of the example, the threshold value is set to 0.5, then follower "ID0" has its pertinence output set as "true", whereas follower "ID1" has its pertinence output set to "false", then it should stop sending data for a period.

Using the *Attention module* to stop transmission for a certain duration may cause some

Figure 16 – Example attention decision.



Source: The author.

data to be lost. However, there are benefits as this mechanism avoids the transmission of unnecessary and semantically irrelevant information. The threshold, queue size, and sleep time can be adapted as needed for the problem. The threshold and queue size are defined in the system instantiation of the *Attention* module, and the sleep time is defined in the description file (Section 4.3.6).

The threshold is a value between 0 and 1. To continue running, the agent must have a relevance value more significant than the threshold. Thus, the higher the threshold, the fewer agents meet this criterion, while the closer to zero, the more agents will easily meet the criterion. The queue size influences the amount of information stored to decide the relevance of an agent. The bigger it is, the more memory will be needed. In addition, if it is substantial, an agent's relevance assessment will consider data that arrived a long time ago, and that may no longer describe the current environment. The sleep time impacts how long an agent must remain without passing on information. So care must be taken because even if at the current time the agent is not being necessary, in the future, it may have the best view of a scene. That way, if the sleep time is too long, it may lose this information.

The output of the *Attention module* follows the structure shown in the Listing 10. The output refers to the decision produced in the *Decision* module and the pertinence recommendation calculated in the *Attention* module. For the previous example, the output is demonstrated in Listing 11.

Listing 10 – JSON Schema for the attention output.

```
{
    "type": "object",
    "properties": {
        "actionvar": "string",
        "value"    : "string",
        "attention": {
           "type": "object",
            "additionalProperties": {
                "type": "array"",
                "items": {
                    "id": "object"
                    "properties": {
                        "id"         : "string",
                        "pertinence" : "string"
                    }
                }
            }
        }
    }
}
```

Listing 11 – Attention output example.

```
{
    "actionvar": "lamp",
    "value"    : "on",
    "attention": [
        {"id" : "ID0", "pertinence" : "true"},
        {"id" : "ID1", "pertinence" : "false"}
    ]
}
```

### 4.3.6 Description File

As mentioned earlier, many of the actions of the chain agents can be configured by the user through the usage of *Description Files*. The system based on the architecture must interpret this description file and uses its definitions to guide its processing of the internal data and parameters of the different modules illustrated previously in Figs. 9 and 10. Such a file facilitates agents' reconfiguration (requirement R4) for applicability in different scenarios (requirement R5).

*Description Files* are documents where the following parameters must be defined:

- **id**: represents agent identification.
- **leader**: the id of an agent upper in the hierarchy if there is one; otherwise, this parameter takes the value "none".
- **rxperiod**: the value of the reception window for the *Rx Window module*, which needs to be regulated as a function of the throughput rate of the sensor present in the system and the desired reactivity of the edge chain agents. It is defined as units of seconds.
- **msgserver**: determines the network information of the message exchange server. Specifically, the IP and port where the server is hosted. The MQTT protocol is used for communication.
- **interfaces**: are the names of the classes containing the software integration interfaces to be used by the chain agents for determining the value of the different variables of interest.
- **perceptions**: represent the variables' names the interfaces have to use to send information to the chain agents. It is a structure where each key is the name of the variable, and the value for that key is an array of strings that represents the values that an interface can send when using that variable.

- **actions**: represents the variables' names that the interfaces have to use to receive information from the different components of the architecture. It is a structure where each key is the name of the variable, and the value for that key is an array of strings representing the values that an interface has to implement to receive data.

- **rules**: represent the decision rules that will be triggered by the system in the *Decision* module. They consist of a list of structures with the "if" and "then" tags. The "if" and "then" assume a structure with one or more components. The components of the "if" need to have the names of the variables defined in perceptions as keys. The value for that key has to be one of the values defined for that variable. In the "then" tag, it has to assume one of the variables defined in the actions and use the value that this variable assumes when the "if" conditions are triggered. The value has to be among the defined values for that action variable.

The description file follows a JSON (JavaScript Object Notation) (PEZOA et al., 2016). The choice of JSON is suited to model the architectural intentions, as it has a vocabulary compatible with human and machine interpretation, thereby meeting the requirement R3. Several applications already use JSON, which favors compatibility and reconfiguration. Hence humans can understand it while computers can process it. For instance, the JSON schema in (DROETTBOOM et al., 2015) is the structure users should follow when crafting the JSON description file for their specific safety systems. The idea is to make it as simple as possible so that a user unfamiliar with this notation can quickly learn and use it. The proposed JSON schema is in the Listing 12.

Listing 12 – JSON Schema for description files.

```
{
    "type": "object",
    "properties": {
        "id"          : "string",
        "leader"      : "string",
        "rxperiod"    : "string",
        "msgserver"        : {
            "type": "object",
            "properties": {"ip"   : "string", "port": "string"}
        },
        "interfaces" : {
            "type" : "array",
            "items": "string"
        },
        "perceptions": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "perception": "string", "values": {"type": "array", "itens": "string"}
                }
            }
        },
        "actions"    : {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "perception": "string", "values": { "type": "array", "itens": "string"}
                }
            }
        },
        "rules"      : {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "if": {
                        "type": "object",
                        "properties": {
                            "type": "array",
                            "itens":{"perception": "string", "value": "string" }
                        }
                    }
                    "then": {
                        "type": "object",
                        "properties": {
                            "type": "array",
                            "itens":{"action": "string", "value": "string" }
                        }
                    }
                }
            }
        }
    }
}
```

The description file can now be written based on the previous JSON schema. The switch and the lamp are applied to exemplify the use of the description file. Therefore, the rules are defined to match the requirement to turn the lamp on when the power switch is up and turn it off when the power switch is down. In the Listing 13 there is the JSON that the user can define for this use case.

Listing 13 – Description file example for the case of a power switch and lamp.

```
{
    "id"          : "ID0",
    "leader"      : "ID10",
    "rxperiod"    : "0.1",
    "msgserver"        : {"ip"  : "192.168.0.6", "port": "1883"},
    "interfaces" : ["powerSwitchInterface"],
    "perceptions": [
        {"perception": "powerswitch", "values": ["up", "down"]}
    ],
    "actions"     : [
        {"action": "lamp", "values": ["on", "off"]}
    ],
    "rules"       : [
        {
            "if"  : [{"perception": "powerswitch", "value": "up"}],
            "then": [{"action"    : "lamp"        , "value": "on"}]
        },
        {
            "if"  : [{"perception": "powerswitch", "value": "down"}],
            "then": [{"action"    : "lamp"        , "value": "off" }]
        }
    ]
}
```

The "powerswitch" is a perception variable (defined in the "perception" tag) that can take on two values, "up" or "down" ({"perception": "powerswitch", "values": ["up", "down"]}). The "lamp" is an action variable that can take two values "on" or "off", which are defined in the {"action": "lamp", "values": ["on", "off"]}. Also, there are two rules ("rules" tag in the description file). The first rule considers for the "if" statement the criterion when the input variable "powerswitch" is "up". In this case, the "then" instruction considers the "lamp" output variable to be "on". The second rule follows the same pattern, but considers for "if" when the "switch" is "off" and "then" the "lamp" as "off".

## 4.4 CHAIN AGENT IMPLEMENTATION

The Python 3.0 programming language (Rossum e Drake (2009)) is used to implement the proposed architecture. However, different projects can use the architecture described in this work as the basis to implement their systems considering other languages and tools.

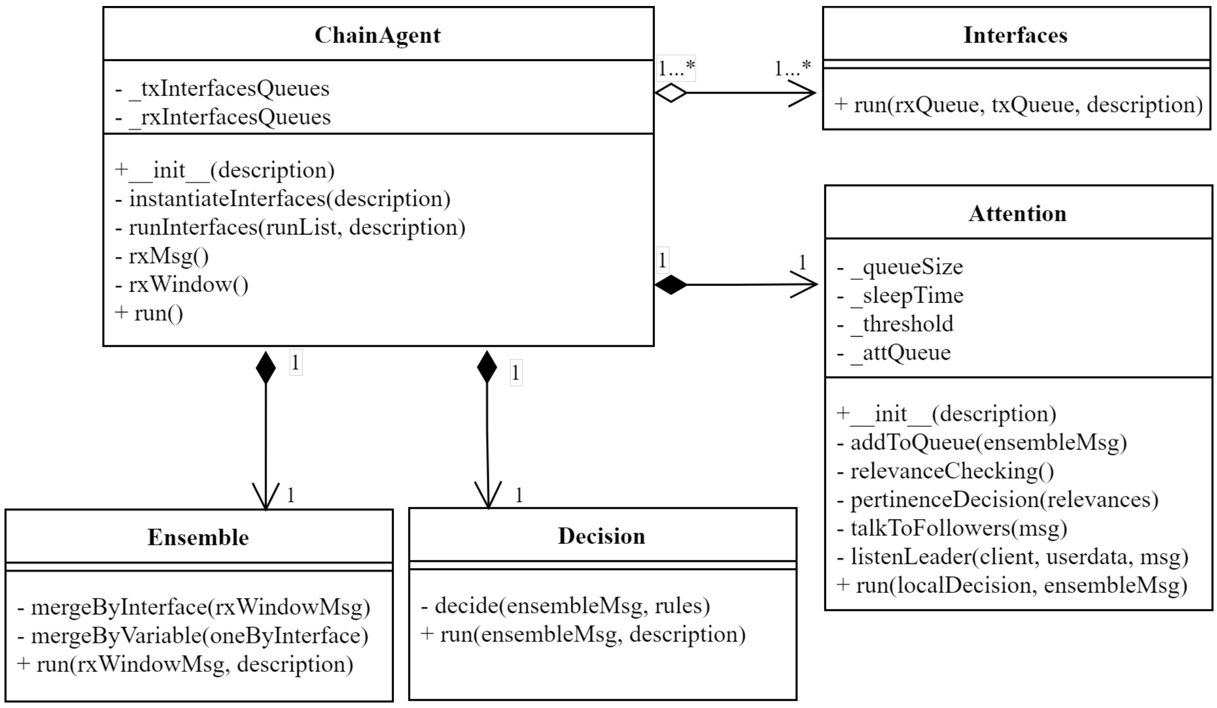
The communication between agents in the chain uses a publish/subscribe paradigm where agents situated lower in the hierarchy subscribe to the topics of agents higher up. MQTT (Message Queuing Telemetry Transport) (Standard (2019)) is the protocol used for exchanging messages between agents present in the architecture. MQTT adopts a topic-based Publisher/Subscriber abstraction for messaging. A device only needs to inform a topic where the message is published. This will then be distributed to all the subscribers to that topic. The Broker is the component responsible for decoupling the publishers from the subscribers and concentrates the reception and distribution of the messages. The Mosquitto server (Light (2017)) was chosen to mediate this communication, as it implements this protocol, is open source, lightweight, and well documented for maintenance.

MQTT runs on top of the TCP protocol, which has higher delivery guarantees than other messaging protocols that use UDP, such as CoAP (Bormann, Castellani e Shelby (2012)). Furthermore, it offers different layers for QoS (Quality of Service), which is essential for our safety-oriented application. It was designed for the IoT context. Therefore, it was initially built with concerns about devices with limited resources and low network bandwidth. Comparisons of MQTT with other protocols can be found in different contributions, including Naik (2017) and Thangavel et al. (2014).

The Chain agent implementation considers the following classes: ChainAgent, Interfaces, Ensemble, Decision, and Attention. Fig. 17 illustrates a UML diagram with the interaction between classes in the python implementation. Each class represents one of the modules previously presented in the reference architecture in section 4.3 and illustrated in Fig. 10. The only module not implemented as a class is the *Rx Window module*, which is a method of the *ChainAgent class.*

The *ChainAgent class* is the main class that manages the agent's execution. It instantiates the other inner class to operate on the correct flow as illustrated in Fig. 10. The *ChainAgent class* is composed of an *Ensemble class*, a *Decision class*, and an *Attention class.* The black diamond with an arrow in Fig. 17 represents the composition notation, and the number *1* in the composition indicates that each *ChainAgent class* is composed

Figure 17 – UML class diagram.


ChainAgent
- _txInterfacesQueues
- _rxInterfacesQueues
+__init__(description)
- instantiateInterfaces(description)
- runInterfaces(runList, description)
- rxMsg()
- rxWindow()
+ run()
Interfaces
+ run(rxQueue, txQueue, description)
Attention
- _queueSize
- _sleepTime
- _threshold
- _attQueue
+__init__(description)
- addToQueue(ensembleMsg)
- relevanceChecking()
- pertinenceDecision(relevances)
- talkToFollowers(msg)
- listenLeader(client, userdata, msg)
+ run(localDecision, ensembleMsg)
Ensemble
- mergeByInterface(rxWindowMsg)
- mergeByVariable(oneByInterface)
+ run(rxWindowMsg, description)
Decision
- decide(ensembleMsg, rules)
+ run(ensembleMsg, description)
1...*
1...*
1
1
1
1
1
1

Source: The author.

of exactly one instance of every other class. The *ChainAgent class* also adds an *Interface class*, represented by the white diamond with an arrow. A chain agent can have one or several interfaces, and different chain agents can use the same interface. To design the flow of the reference architecture described in the previous section, the *ChainAgent class* implements the following methods:

- *___init___(description)*: is responsible for initializing the *_txInterfacesQueues* and *_rxInterfacesQueues* variables, which are used to exchange messages with the Interfaces. In addition, it instantiates the required classes, namely Ensemble, Decision, and Attention. It calls the built-in methods *instantiateInterfaces* and *runInterfaces*, which are described below. The "description" input parameter represents the JSON description file as presented in Section 4.3.6.

- *instantiateInterfaces(description)*: is responsible for reading the names of the Interfaces declared in the description file and importing them into the python *ChainAgent class*s at runtime. This is necessary since the interfaces are application dependent. In this case, they are coupled to the system by the user, who makes the program available as a file that must be instantiated at runtime. So the user does not need to change the agent's python code to declare new interfaces. Python's importlib (importlib... (2022)) library is applied for this purpose. The "description" input parameter represents the JSON description file as presented in Section 4.3.6. This method returns an array with the python objects of each interface.

- *runInterfaces(runList, description)*: is responsible for executing each declared interface. For that, it runs each interface in parallel as an independent process so that everyone can manage its respective sensor or effector without blocking. The multiprocessing library (multiprocessing… (2022) and Palach (2014)) is applied for this purpose. Furthermore, this method creates the receive and transmit queues, which are used to receive and send messages to the interfaces, and include them in the *_txInterfacesQueues* and *_rxInterfacesQueues* arrays. Then this method executes the method *run(txQueue, rxQueue, description)* that must be declared in each interface, and passes the parameters *txQueue* and *rxQueue*. *txQueue* is the transmit queue and *rxQueue* is the receive queue and *rxQueue* is the json description file. The input variable "runList" is an array with the instances of each interface object that were declared in the *instModules(modules)* method. In addition, it receives the description file ("description") as input.

- *rxMsg()*: is responsible for receiving and validating if the messages coming from the interfaces have the correct semantics. This is a callback method that runs asynchronously as an independent thread. It constantly checks the *_rxInterfacesQueues* for new messages.

- *rxWindow()*: is responsible for implementing the behavior of the *Rx Window module*, as in Section 4.3.2. It blocks the operation of the thread that calls it for the period defined in the receive window. It then returns all valid messages stored by *rxMsg()* in this period. This method regulates the agent processing frequency.

- *run()*: defines the method that must be called to start the agent operation. All necessary modules are executed in an infinite cycle in sequence. It manages the passing of variables between the called methods. It starts with *rxMsgWindow()*, executes the ensemble, the decision, and finally the attention. Like the flow in Fig. 10.

The agent instantiates *Interfaces class* independently and runs them as a parallel process in the device. The communication between the interfaces and the agent uses an inter-process shared queue. There are two queues for each interface, one is used to send response messages from the attention module to the interfaces, and the second queue is used to receive the interface's output. This behavior is implemented in the prototype proposed in this work using the python multiprocessing library (Palach (2014)).

In the case of the present architecture prototype based on the Python language, the script for the *Interfaces class* implemented by the user has to provide a Python method called *run(rxQueue, txQueue, description)*:

- *run(rxQueue, txQueue, description)*: receives as input the parameters *rxQueue*, *txQueue* and *description*. The *rxQueue* is a multiprocessing queue object (Palach

(2014)) used to receive data from the agent. The *txQueue* is a multiprocessing queue object that passes data to the agent. The description parameter is a Python dictionary that follows a JSON structure with the definitions in the description file (Section 4.3.6). When the agent runs for the first time on a device, all interfaces that follow this pattern are also executed and become part of the online agent execution.

The *Ensemble class* implements the *Ensemble module* presented in Section 4.3.3. This class has the following methods:

- *mergeByInterface(rxWindowMsg)*: is responsible for merging the input coming from the *Rx Window* module (*rxWindow()* method in the ChainAgent class) by interface, as described in step 1 of Fig. 14 . The input variable "rxWindowMsg" is the message with all the data received for each interface.
- *mergeByVariable(oneByInterface)*: is responsible for merging the input coming from the *mergeByInterface* method by variable, as described in step 2 of Fig. 14. The data from the *mergeByInterface* method is in the "oneByInterface" variable.
- *run(rxWindowMsg, description)*: is the method called to manage the ensemble routines. Therefore, it calls the previously reported methods *mergeByInterface* and *mergeByVariable*, in the correct sequence, and passes the output from the first to the second method. The "rxWindowMsg" input is data from all input interfaces coming from the *Rx module*, and the "description" variable is the JSON description file.

The *Decision class* represents the *Decision module* (Section 4.3.4). It considers the methods below:

- *decide(ensembleMsg, description)*: is responsible for applying the rules predefined by the user to the online data. Therefore, the "description" variable shows a JSON declation where the user-defined rules are, and the "ensembleMsg" variable is the data on which the rules are applied. The output of this method represents the answer to the rules inference. If no rules match the input data, *None* is the output.
- *run(ensembleMsg, description)*: execute the *decide* method, and it returns a string with the output of the rules processing as defined in the Listing 2. The "ensembleMsg" is the output of the ensemble module, and "description" is the JSON description file.

Finally, there is *Attention class*, which describes the behavior presented in Section 4.3.5 . The methods are presented below:

- *___init___(description)*: is responsible for initializing the MQTT connection by subscribing to the lead agent topic. For that, it uses the leader's ID that is in the "description" input variable. MQTT is implemented in the program using the Python Paho library (Eclipse... (2022)). It also initializes the *_queueSize*, *_sleeptime* and *_threshold* used in the instance of the attention class methods.

- *addToQueue(ensembleMsg)*: is responsible for including the data coming from the ensemble ("ensembleMsg") in the internal *_attQueue*.

- *relevanceChecking()*: implements relevance checking. For this, it evaluates the data in the internal variable *_attQueue*, as described in Section 4.3.5. The output is the relevance of each interface loaded by the agent.

- *pertinenceDecision(relevances)*: implements the pertinence decision routine. It takes as input the variable "relevances", which was calculated in the *relevanceChecking* method.

- *talkToFollowers(msg)*: is responsible for publishing the message to the MQTT topic. The topic name is the same as the agent's Id that is calling this method. So the subscribed follower agents receive their decisions whenever this method is called.

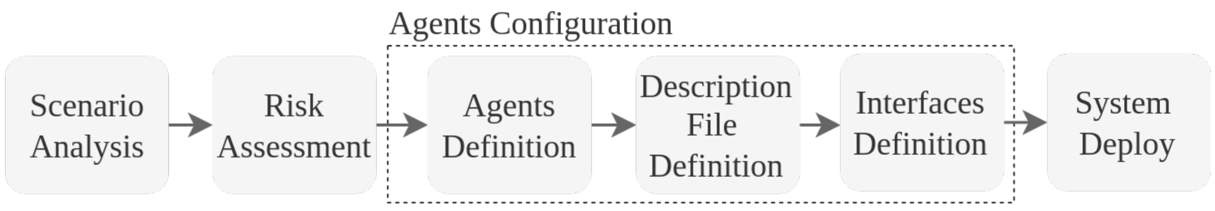
- *listenLeader(client, userdata, msg)*: is a method that runs a thread parallel to the general flow of attention. It is always listening if a new message is received from the leader. This is a mandatory method used by the Python Paho library (Eclipse... (2022)) which calls this method every time an asynchronous message event is identified. The client's input parameters are "client", which is the unique id of the MQTT client instance, "userdata" which is MQTT user data passed as parameters to callbacks, and "msg", which is the message received in the topic.

- *run(localDecision, ensembleMsg)*: controls the operation of the attention module by executing the methods in the correct order. It executes *addToQueue*, *relevanceChecking*, *pertinenceDecision* methods in that order. Finally, it evaluates if there is any message from the leader agent, if yes, it returns it; otherwise it returns the variable "localDecision" and concatenates the calculated memberships (Listing 10). It also triggers the *talkFollowers* method, to send your decision to all followers. The "localDecision" variable comes from *Decision module* and the "ensembleMsg" variable comes from *Ensemble module*.

## 4.5 SAFETY SYSTEM STEPS

After presenting the architecture, it is time to explore how to instantiate it as a safety system use case. Therefore, this section is dedicated to showing the implementation process and sharing some of the experiences gained in this process. Fig. 18 illustrates the followed steps and gives their description.

Figure 18 – Architecture implementation steps.



Source: The author.

The *Scenario Analysis* step requires a good view of the environment to determine the available devices, how to extract information from each device, and the communication infrastructure. It also selects the available hosts for processing and where decides each agent module should be deployed and how they communicate with each other. As a result, this step provides a list of devices divided into sensors and effectors. Likewise, there is a list of available processing units, like high-end computers or embedded devices. In addition, it provides a representation of the physical communication infrastructure. This is important for visualizing each component that the scenario connects. In summary, this step is needed to produce the following deployment components:

1. List of sensors and effectors.
2. List of processing units.
3. Representation of the physical communication infrastructure.

When implementing the adopted safety system use case scenario, it is first necessary to review, understand and analyze the risks in the environment. To achieve this, there is a need to design a *Risk Assessment* step. Some standards provide guidelines for this goal. In (SMITH; BROOKS, 2013) the author argues that systems must comply with the recommendations of ISO 31000:2009 ((ISO, 2009) (PURDY, 2010)). This standard represents the risk assessment process in different stages. The steps it considers are: setting the context, identifying the risk, analyzing the risk, assessing the risk, treating the risk, communicating and consulting, and monitoring and reviewing. It is beyond the scope of this work to detail these processes. Some industries may also consider their own risk assessment process or rely solely on specialist knowledge. Regardless of the method used, the result of this step is applied to describe the agent description file.

After that, three essential steps directly related to agent configuration are grouped as *Agent Configuration*. The first is the *Agents Definition* step, which evaluates the number of agents and the hierarchy between them considering the machines listed in *Scenario Analysis*. It is important to consider the processing capacity of each device and the sensors available to define the number of agents and consider which machine should process the specific sensor. The communication capacity of the machines must also be considered, for example, whether it has wireless or wired communication. This stage must also evaluate who should be followers or leaders. Remember that followers pass raw information to leaders, who in turn feed their decisions back to followers. A good practice is to consider as leader agents machines with a high processing capacity that may be in more distant positions in the network since they must process data from more agents, while agents directly connected to sensors do not receive data from other agents.

The second step of *Agent Configuration* involves the necessary definition of the file used as input for the agent configuration. This is achieved as part of the *Description File Definition* step (Section 4.3.6). This is necessary to analyze the information extracted from the devices and the responses produced according to the risk assessment and scenario analysis. In addition, the rules produced must match these requirements.

The *Interfaces Definition* is the third step of *Agent Configuration* which takes place when the programs that extract information from the sensors are implemented. The used data structure must conform to the description in the 4.3.1 section. This step results in a program compatible with the architecture and can read data from sensors or respond to effectors.

Finally, the last step is *System Deploy* where the system is run by merging all previous settings. This means connecting all the devices, configuring the machines, configuring the network, and deploying the different programs on the machines previously defined in the scenario analysis. Only then can the system be put into operation. Note that this phase can have many infrastructure compatibility complexities such as library installation, physical resource configuration, etc. Perhaps some testing and reruns may be required before final system deployment.

## 4.6 DISCUSSION

This section presented the proposed reference architecture that describes agents and their roles in an IIoT infrastructure. Its design allows communicating parts to avoid unnecessary transmission over the network when primary safety risk information is not identified. In this case, edge devices may process information until they identify imminent risks. Only then is the information transmitted to the centralized server for more accurate and further processing. This chained connection between agents allows high-level agents in the infrastructure to send their decision to other agents, which in turn decide whether to stop or continue data transmission to the other side of the network, resulting in a reduction in network traffic. The proposed architecture is in line and fits well with the two computing paradigms: edge computing and orchestration.

The configuration file, along with the inclusion of the interface, allows the architecture to meet reconfigurability and scalability requirements. Therefore, by changing a single file, the agents' behavior can be adapted to the needs of the user or the environment. Interfaces are the only external software required by the architecture. The interfaces are necessary to simplify the integration of new sensors or actuators into the system. Their usage bridges the gap between physical and cybernetic semantics.

The adopted architecture may be used to equalize the different sensor input frequencies using the interfaces and ensemble for this. Furthermore, it allows summarizing of all information from multiple variables, avoiding conflicts in the decision stage through the use of rules defined by the user to infer the final result.

Although the primary motivation for the proposed IIoT architecture has been the development of a solution for safety risk situations, the language presented to configure the agents allows other use cases besides safety. This is made possible by using configuration tags that are more user-friendly for industry users than low-level technology or context-specific programming languages.

# 5 APPLICATION IN INDUSTRY

This chapter instantiates the architecture implementation to enhance safety in an aircraft assembly line. The following sections detail the adopted scenario and the selected system configuration to support this application. In addition, the chapter presents some experiments conducted to evaluate several characteristics of the proposed architecture.

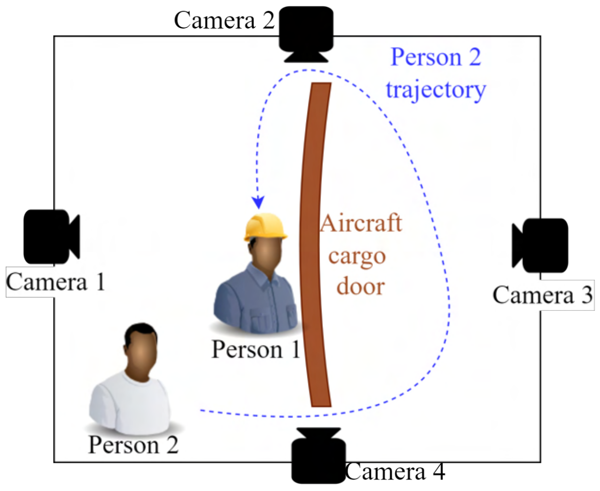
## 5.1 SCENARIO

The instantiating of the architecture was demonstrated a mock-up IIoT scenario that emulates an assembly operation in an aircraft cargo door. Due to the nature of some operations, only one is allowed to be in the vicinity of the cargo door to avoid accidents; therefore, the entry of a second person into this workspace represents a safety risk. In the current scenario, whenever a second person is detected in the vicinity of the cargo door, the system exhibits a message, producing a sound signal. Note, however, that more appropriate real-world actions would include activating emergency stops in motorized tools in operation, alerting the operators, raising alarms, etc.

The current observed system is monitored by four security cameras connected to a network and placed around the cargo door (Fig. 19). Standstill pictures from the live stream of the four cameras can be seen in (Fig. 20). In this case, the four cameras correspond to four inputs in the system and four sources of information for the same set of variables, namely the collision risk. It is important to observe that as the different sensors transmit their views of the monitored scene, there is possibly a significant level of overlay and redundancy among their transmitted information, common in safety applications. Although the explored scenario relates to a safety application, one must stress the fact that the system, as is, does not meet existing criteria for real-world usage (i.e., it is not a safety system). It serves only the purpose of demonstrating the architecture. Note also that despite focusing in this chapter on a single use case scenario, the adopted architecture can equally apply to safety monitoring in other contexts, such as human-robot interaction, an example is in the work of Barbosa et al. (2022).

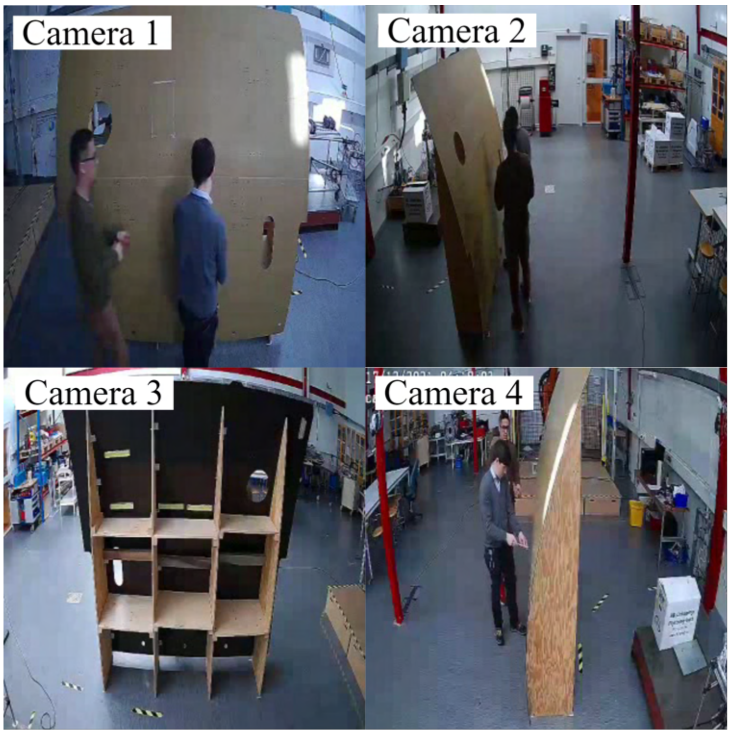
In the scenario considered, one operator is working on the cargo door when a second operator invades the scene and comes in contact with the first one. The safety system processes the live streams coming from the cameras. The scenario was recorded for 30 seconds to generate a representative input dataset, which is used to collect performance metrics and compare the obtained results. In the first 16 seconds, Person 2 (the external invader) walks around the aircraft cargo door. Then, it collides with Person 1 for 10 seconds. Finally, Person 2 walks away for a duration of 4 seconds. Each camera perspective is recorded for the dataset at 10 fps rate and a 320x320 pixels resolution.

Figure 19 – Scenario.



Source: The author.

Figure 20 – Scenario visualization from different cameras' perspectives.



Source: The author.

The used infrastructure is summarized in Table 3. It shows four edge devices (Raspberry PIs), a WiFi router, and a server machine. Each camera is directly connected to an edge device by cable using the Fast Ethernet (up to 100 Mbps) interface. The edge communicates with the server via a WiFi network 802.11n (up to 300 Mbps).

Table 3 – Specification of the devices in the scenario.

| Device | Model | Specification | |
|---|---|---|---|
| Camera | Foscam R2M | Resolution | 1920 x 1080 (2.0 MP) |
| | | Frame rate | 25fps (1080P) |
| | | Others | Camera IP, Pan/Tilt control |
| WiFi Router | D-Link GO-RT-N300 | Standard | 802.11b/g/n |
| | | Speed | 300 Mbps 2.4 GHz |
| Edge | Raspberry Pi 3 B+ | Processor | Quad Core 1.2GHz Broadcom BCM2837 |
| | | RAM | 1 Gb |
| | | OS | Raspberry Pi OS 32bit |
| Server | HP Worksta tion Z2 G5 | Processor | Intel(R) Xeon(R) W-1250 CPU @ 3.30GHz |
| | | RAM | 16GB SSD |
| | | GPU | NVIDIA Quadro RTX 4000 8GB GDDR6 |
| | | OS | Ubuntu 20.04.3 LTS |

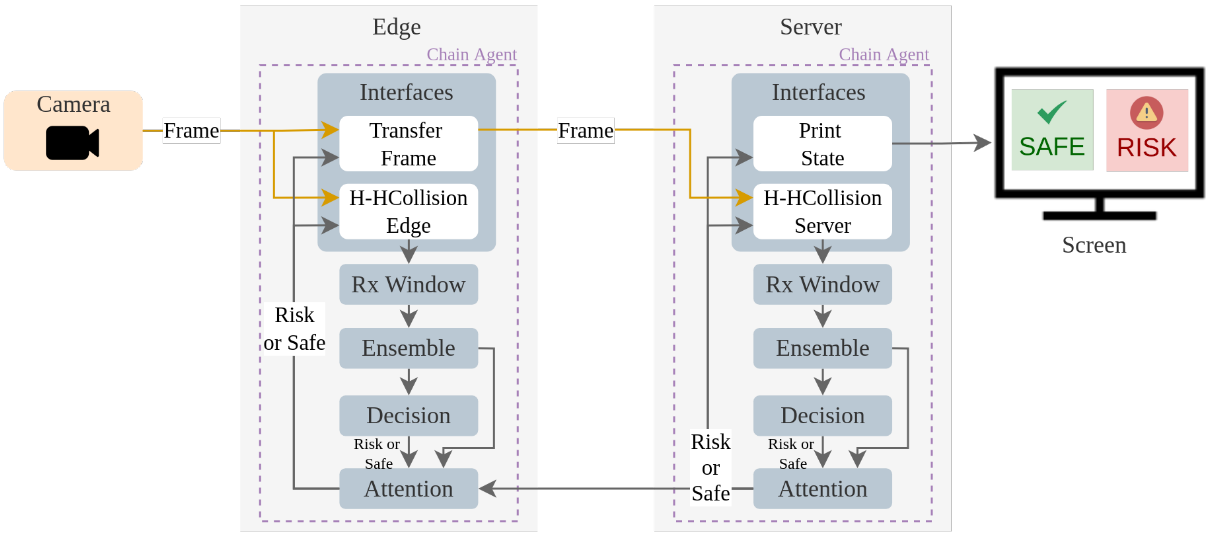
Source: The author.

## 5.2 INTERFACES AND DESCRIPTION FILES

Specific interfaces were defined for this scenario (Fig. 21). As already mentioned, integration interfaces guarantee the harmonization of raw sensor/actuator data with the execution semantics of the system. For the chain agent at the edge device, two interfaces are implemented, the first is called *H-H Collision Edge*, which allows the system to receive data from the cameras and detect collisions. The second interface (*Transfer Frame*), working in tandem with the first one, allows redirecting the video frames towards the chain for further processing. The chain agent in the server also offers a collision detection interface (*H-H Collision Server*) with a more accurate collision detection algorithm compared to the one from the edge device. It also supports an interface that raises alarms (*Print State*) when a collision between two operators is detected.

The Description File for this edge agent is presented in Listing 14, and for the server agent is in the Listing 15. The id of this edge device is "edge1". Its leader, up the chain, has the id "server1". It sets a window time for the reception of data to 0.5 seconds. The MQTT server is hosted on a machine with IP address: "192.168.0.6" and port number: "1883". It executes the interfaces "HHCollisionEdge" and "TransferFrame". There is only one possible perception variable, that is called "collision". This variable can assume two literal values, "true" or "false". Also, there is only, in this example, a single possible action variable, that is called "risk", that can be "true" or "false". There are two rules in the file. The first rule states that if a collision event is true ("if" : [{"perception": "collision", "value": "true"}]), then there is a risk ("then": [{"action": "risk", "value": "true"}]). The second rule determines that if a collision is false ("if" : [{"perception": "collision", "value":

Figure 21 – Reference architecture with the interfaces for the specific application.



Source: The author.

"false"}]), then there is not a risk ("then": [{"action": "risk", "value": "false" }])

Listing 14 – Description File for edge agent.

```
{
    "id": "edge1",
    "leader": "server1",
    "rxperiod": "0.5",
    "mqtt":{"ip":  "192.168.0.6", "port": "1883"},
    "interfaces": ["HHCollisionEdge", "TransferFrame"],
    "perceptions": [
        {"perception": "collision", "values": ["true", "false"]}
    ],
    "actions"    : [
        {"action": "risk", "values": ["true", "false"]}
    ],
    "rules":[
        {
            "if"  : [{"perception": "collision", "value": "true"}],
            "then": [{"action"    : "risk"     , "value": "true"}]
        },
        {
            "if"  : [{"perception": "collision", "value": "false"}],
            "then": [{"action"    : "risk"     , "value": "false" }]
        }
    ]
}
```

Listing 15 – Description File for server agent.

```
{
    "id": "server1",
    "leader": "none",
    "rxperiod": "0.5",
    "mqtt":{"ip":  "192.168.0.6", "port": "1883"},
    "interfaces": ["HHCollisionServer", "PrintState"],
    "perceptions": [
        {"perception": "collision", "values": ["true", "false"]}
    ],
    "actions"    : [
        {"action": "risk", "values": ["true", "false"]}
    ],
    "rules":[
        {
            "if"  : [{"perception": "collision", "value": "true"}],
            "then": [{"action"    : "risk"     , "value": "true"}]
        },
        {
            "if"  : [{"perception": "collision", "value": "false"}],
            "then": [{"action"    : "risk"     , "value": "false" }]
        }
    ]
}
```

The implementation of the interfaces is discussed next. The *Transfer Frame* interface that is at the edge device initiates the transmission of the frames to the server when the decision received is {"risk":"true"}. It uses the OpenCV library (BRADSKI; KAEHLER, 2000) to read the frames and leverages the ImageZMQ library (BASS, 2020) to transmit OpenCV frames to the server. Otherwise, it does nothing. The *Print State* interface, in the server, is a visual feedback that prints onto a screen the safety state in the environment. In other words, it prints the text *Risk* when receives {"risk":"true"}, else it prints the text *Safe* on the screen.

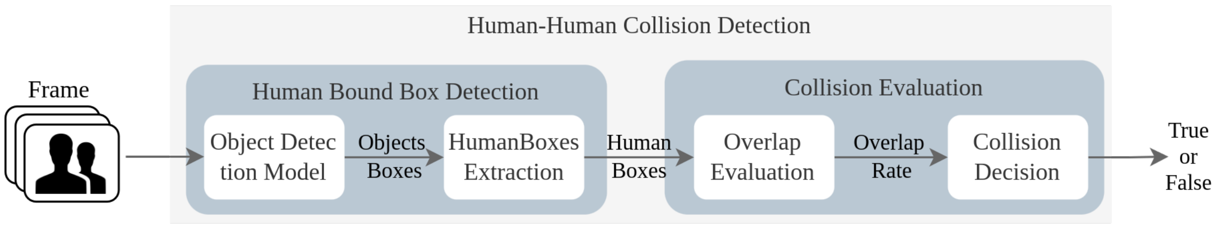
The *H-H Collision* interface is located on both the edge and the server. The difference in the setup of this interface across the different devices is only in terms of the applied model for object detection and how it reads the frames. At the edge, it uses a simpler model that requires low memory and processing power to return responses, namely, the Mobilenet v2 library (SANDLER et al., 2018). To read the frames at an edge device, it uses the OpenCV library. When running at the server, it uses the YOLOv3 base version (REDMON; FARHADI, 2018) that requires more memory and processing power, but it is, on the other hand, more precise than the Mobilenet v2. The server uses the ImageZMQ library to read the images.

Fig. 22 illustrates the process responsible for collision detection. First, an object detection model is applied, which returns an array with the boxes for all the objects identified in the processed frame. Among the identified objects, the boxes with humans are selected. Considering all the detected human boxes, the algorithm next evaluates if there is a collision. For that, it checks the overlap between each two by two combinations of human boxes. The calculus of the overlap is based on the Intersection over Union (IoU) metric. However, instead of the Union of boxes, it applies the smallest human box in the image as a reference because a smaller box completely covered by a bigger one has to return the maximum level of possible contact. This represents a short person in front of a taller one. The use of IoU would only return the proportion of the smaller box related to the bigger one. Therefore, the overlap is calculated using the Intersection over Smaller (IoS), Eq. 5.1 refers to the adopted IoS metric:

$$IoS(x,y) = \frac{H_x \cap H_y}{\min\left(H_x, H_y\right)} \tag{5.1}$$

$H_x$ and $H_y$ are the boxes of the humans $x$ and $y$, respectively, where $x \neq y$.

Figure 22 – Human-Human Collision detection module.



Source: The author.

With the IoS for each pair of humans in the scene, the Overlap Evaluation step returns the largest IoS among all the pairs evaluated. This represents the Overlap Rate (OR) for that frame. It is calculated following the Eq. 5.2.

$$OR = \max_{\forall x,y \in P, x \neq y} IoS(x,y) \tag{5.2}$$

Where $P$ is the set of all People identified in the frame.

Lastly, the collision decision considers a pre-established threshold. If the OR is equal or is above this threshold, it returns that the collision is True for that frame. Whereas if the OR is below the threshold, it returns False as a collision indicator. In the present application, we assume a threshold of 5% to be considered a collision. This value was empirically determined to make the system as sensitive as possible to collisions. Values greater than this cause some collisions to not be alerted.

## 5.3 TECHNICAL EXECUTION FLOW

The execution initiates with the camera sending a frame to their respective edge device. The *H-H Collision Edge* interface processes this frame and identifies if there is a collision. It then passes its findings to the *Rx Window* that receives all the observations made within a pre-defined window period. Then, it passes all received data to the *Ensemble* module. The latter converts it into one single perception, represented with the variable "collision", which it then forwards to the *Decision* module. This module takes the collision information and applies the rules defined in the description file. Depending on the result of executing the rules, if there is a collision, it flags a *Risk* situation; otherwise, it signals a *Safe* state. Considering that the server did not send any additional message, the *Attention* module only passes along the edge decision to its interfaces. If the decision is *Safe*, the *H-H Collision Edge* continues processing more frames. However, if the response is one of *Risk*, then the *H-H Collision Edge* stops its processing, and the *Transfer Frame* step takes over the processing sending all the received frames to the server.

When the server receives the frames from the edge, it follows the same process reported in the previous paragraph. However, with one difference, the *Attention* module output is also passed on to the edge and the *Print State* interface, which prints in the server information showing whether there is a risk being reported or not.

The *Attention* module located at the edge uses the server's decision. It passes along to the local interfaces when the server sends the decision indicating either *Risk* or *Safe* to the edge. When the server reports a *Risk* state, the *Transfer Frame* interface keeps sending the video frames, and the *H-H collision Edge* remains halted. However, when the server reports a *Safe* state, *Transfer Frame* stops, and the *H-H collision Edge* restarts to process the frames locally in the edge.

## 5.4 EXPERIMENTS AND RESULTS

This section discusses the results obtained while considering the use case in Section 5.1 and the dataset generated and mentioned therein. Section 5.4.1 focuses on analyzing the impact on accuracy by including our proposal, considering the increase in the number of cameras observing the environment from different perspectives. Next, Section 5.4.2 analyzes the impact of applying edge devices on the network. Finally, Section 5.4.3 evaluates the inclusion of the proposed attention mechanism to reduce the network load.

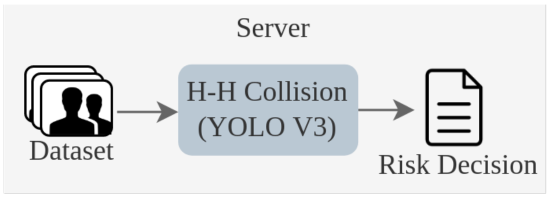
### 5.4.1 Accuracy analysis

This experiment evaluates the impact on the general accuracy when using the proposed architecture. The scenario presented in Section 5.1 is used in this experiment. Accuracy is measured based on the correct identification of a *Risk* or *Safe* situation for each dataset frame. The experiment compares the results produced when using the Proposed Architecture (PA) with other models: YOLO v3 (YV3), Mobilenet v2 (MN), and Mobilenet v2→YOLO v3 (MN→YV3). Fig. 23 illustrates the application of each of these models. The YV3, MN, and MN→YV3 are applied directly to each dataset frame running on a single machine (server device described in Table 3), i.e., without considering the distribution of processing in the network. The MN→YV3 first applies the MN to the frame; if this model detects risk, then the YV3 is applied to the other frames offering different perspectives, with an unanimity ensemble as described in Section 4.2. The MN→YV3 is close to our approach, although it runs on a single computer and uses no network between the executing models. The PA is used by applying Configuration in Fig. 23d for network communication and the active attention support (Section 4.3.5). The edge device and the server are wirelessly connected to the router (300Mbps 802.11n WiFi router). Device specifications are in Table 3.

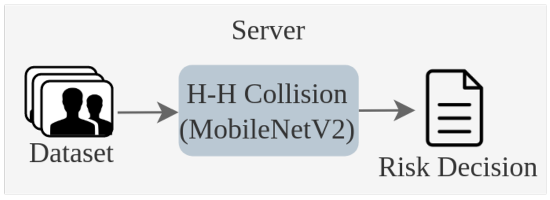
Different cameras are used in this evaluation, considering the positions illustrated in Fig. 19. Each camera is evaluated individually, cameras 1, 2, 3, and 4. We also evaluate multi-camera scenarios, one with the four cameras being used at once (labeled as configuration 1-2-3-4), whereas the other setup using cameras 2, 3, and 4 is labeled as

Figure 23 – Configurations for the accuracy analysis.

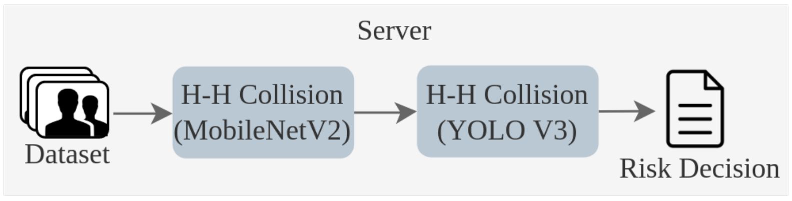
(a) YV3



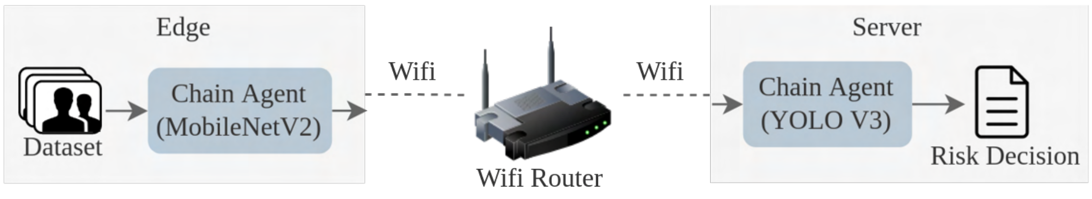
(b) MN



(c) MN→YV3



(d) PA



Source: The author.

configuration 2-3-4. For all the cameras, frame resolution is set to 320x320 pixels, the video frame rate is 10 fps, and different combinations of models are tested. Table 4 lists the parameters applied in the experiment.

Table 4 – Accuracy experiment parameters.

| Parameter | Description |
|---|---|
| Frames Resolution | 320x320 px |
| Frame rate | 10 FPS |
| Number of executions | 30 |
| Execution time | 30 seconds |
| Cameras configuration | 1, 2, 3, 4, 2-3-4 and 1-2-3-4 |
| Models | YV3, MN, YV3→MN and PA |

Source: The author.

The results are evaluated in terms of the metrics: accuracy (ACC), sensitivity (SEN), and specificity (SPE). They are calculated as in the equations 5.3, 5.4 and 5.5, respectively, based on the classification metrics: True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN).

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.3}$$

$$SEN = \frac{TP}{TP + FN} \tag{5.4}$$

$$SPE = \frac{TN}{TN + FP} \tag{5.5}$$

Also, the mean time per frame and Frames Per Second (FPS) that the models YV3, MN, and MN→YV3 require to process in a single machine are evaluated. The MN→YV3, in this case, considers only the MN detection of frames from one single camera perspective. Three different machines are considered: the edge device, the server device with GPU, and the server device without GPU. The experiment processed 300 frames of the dataset.

Table 5 presents the timing information and FPS rate achieved for the execution of different models and processing machines. When using the YV3 model, the edge device has one frame processed every 4.5 seconds. Hence one cannot run this model in the edge device for a real-time application. That is why this device uses the MN, which can process 6.3 frames each second. Note that the MN could be applied at the server to reach a higher FPS. However, for our application, it is more critical to ensure a more accurate model in the server, that is the case when using YV3.

Table 5 – FPS for different models and processing machines.

| Model | Edge | Server CPU | Server GPU |
|---|---|---|---|
| YV3 | 0.223±0.008 | 9.233±0.423 | 38.028±2.717 |
| MN | 6.306±0.671 | 102.270±5.327 | 159.042±9.842 |
| MN→YV3 | 0.211±0.006 | 8.432±0.337 | 32.484±2.098 |

Source: The author.

Given that the server CPU processes 9.2 FPS with YV3 and that the edge processes 6.3 fps when using MN, the application of at least two edge devices processing in parallel surpasses the processing performance achieved by YV3 in a server without GPU. On the other hand, when the server has a built-in GPU, it becomes necessary to have at least 7 edge devices in order to equal the performance of such GPU based server. However, the network becomes the bottleneck and limiting factor with more devices. When processing takes place at the edge devices, the network load level evaluated in Table 8 (configuration C) is often avoided when there is no risk. The FPS of the MN→YV3 is lower than that of the other models for all devices due to having to process first the MN and then the YV3.

Table 6 presents the results for accuracy experiments. The PA model is the only one with the Standard Deviation once each execution can result in different results due to network load and instability. The others setups exhibit a somehow deterministic behavior as they run on a single computer.

Table 6 – Accuracy results.

| Cams | Model | ACC | SEN | SPE |
|---|---|---|---|---|
| 1 | YV3 | 0.96 | 1.00 | 0.94 |
| | MN | 0.98 | 0.95 | 1.00 |
| | MN→YV3 | 0.97 | 0.98 | 0.97 |
| | PA | 0.97±0.01 | 1.00±0.01 | 0.96±0.01 |
| 2 | YV3 | 0.92 | 1.00 | 0.89 |
| | MN | 0.67 | 0.02 | 0.98 |
| | MN→YV3 | 0.89 | 0.71 | 0.97 |
| | PA | 0.93±0.04 | 0.85±0.13 | 0.96±0.03 |
| 3 | YV3 | 0.68 | 0.00 | 1.00 |
| | MN | 0.67 | 0.00 | 1.00 |
| | MN→YV3 | 0.67 | 0.00 | 1.00 |
| | PA | 0.68±0.00 | 0.00±0.00 | 1.00±0.00 |
| 4 | YV3 | 0.55 | 0.05 | 0.79 |
| | MN | 0.63 | 0.03 | 0.92 |
| | MN→YV3 | 0.60 | 0.02 | 0.87 |
| | PA | 0.62±0.03 | 0.00±0.00 | 0.92±0.05 |
| 1-2-3-4 | YV3 | 0.80 | 1.00 | 0.70 |
| | MN | 0.91 | 0.95 | 0.89 |
| | MN→YV3 | 0.86 | 0.98 | 0.81 |
| | PA | 0.86±0.05 | 0.99±0.02 | 0.80±0.07 |
| 2-3-4 | YV3 | 0.82 | 1.00 | 0.73 |
| | MN | 0.62 | 0.05 | 0.89 |
| | MN→YV3 | 0.79 | 0.71 | 0.83 |
| | PA | 0.82±0.07 | 0.76±0.17 | 0.85±0.06 |

Source: The author.

When using only a single camera, the camera 1 perspective has the best ACC of all models, followed by camera 2, camera 4, and finally camera 3. This is justified, given that camera 1 is positioned to capture humans side by side, camera 2 and camera 4 suffer from human occlusion since one person is in front of the other person. However, for camera 2 this is more favorable once it is a little more inclined to a position that enjoys a better vision of the complete body for different persons. Camera 3 clearly does not see human interaction. It only captures the passage of Person 2, which does not represent a collision.

This justifies its low scores SEN = 0 and SPE=1 for all models, given that it always returns a *Safe* state once it cannot see any collision. This is due to the occlusion caused by the cargo door. Camera 4 achieved ACC is even worse than that of Camera 3 as it has a worse SPE. It sometimes detects a *Risk* when there is none. This occurs when Person 1 passes in front of Camera 4 at the beginning of the dataset processing, which causes Camera 4 to perceive a collision event.

When comparing experiments with multiple cameras (1-2-3-4 and 2-3-4) to experiments with single cameras (1, 2, 3, and 4), the ACC results with single Cameras 1 and 2 surpass those with multi-cameras, being around 15% more accurate. However, when using single cameras, one must trust their result and ensure that they are positioned at the best location. For the multi-camera setup, one can consider different perspectives and make a unified decision that combines the different camera findings. This makes the multi-camera scenario more attractive as it achieves good results. For instance, if a collision occurs in front of camera 3 instead of camera 1. Camera 1 ACC would drop, but the multi-camera model still can keep the overall ACC as it also considers the response of camera 3. The configuration 2-3-4 is interesting; once it disregards the camera in the best position, that is camera 1. Moreover, even without it, it manages to keep accuracy close to the one with 1-2-3-4, except for the MN model.

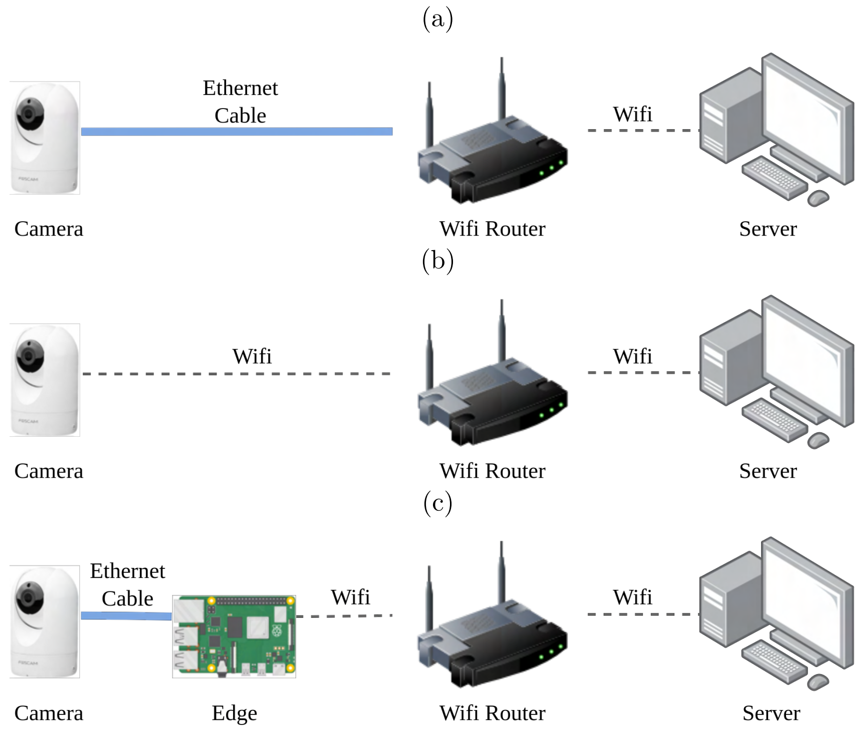
At first sight, the MN model has the best ACC for most configurations (1, 4, and 1-2-3-4). This is an unexpected result because this model is more restricted and prioritizes processing speed over accuracy. However, the fact is that for Camera 1 all the models have similar results with at most 2% of difference. For camera 4 the MN outperforms the others once it, most of the time, returns a *Safe* state. This improves its SPE, while YV3 makes more mistakes in this case. The models MN→YV3 and CA take advantage of it because they first execute a pre-detection using MN, then apply YV3, so that ACC and SPE are close to those of the MN.

The PA follows the MN→YV3 ACC for all configurations while staying in the standard deviation range. This shows that including the PA does not impact the general ACC of the system.

### 5.4.2 Network impact of edge devices

The equipment defined in Table 3 was set up in different manners according to Fig. 24 to validate the extent to which the edge/cloud architecture proposed impacts the performance of the network. The difference among the configurations is mainly in terms of how the devices (camera or edge device) are connected to the router (300Mbps 802.11n WiFi router). In the first case (Fig. 24a), the camera is directly connected to the router using a Fast Ethernet link (100Mbps IEEE 802.3 Local Area Network). The second configuration (Fig. 24b) uses a wireless camera communicating using the multi-antenna high data rate 802.11n standard. Finally, the third configuration (Fig. 24c) connects the camera to an edge device Ethernet interface. In turn, the edge device wirelessly connects to the server. TCP is the underlying transport protocol to exchange flows among the devices.

Figure 24 – Configurations for the communication devices used for the network experiment.



Source: The author.

The following evaluation metrics are considered: Acknowledgement Round Trip Time (ARTT), packet loss, the number of exchanged packets, and the total data length transmitted. The ARTT metric is calculated as the time the cameras or edge devices take to respond with an acknowledgment (Ack) to packets sent by the server. As the cameras do not implement a network interface that can be easily monitored, all packet level measurements are carried out at the server's network interface. The lost packets metric refers to the level of packets that are dropped during their transmission. We also calculate the number of packets and the length of data successfully received by the server. The higher these two metrics are, the more packet processing overhead the CPU requires to handle the incoming traffic.

A live video transmission resolution of 640x480 pixels and a frame rate corresponding to 15 FPS were adopted. These represent upper bounds that allow the processing of frames by edge devices in the experiments. An experiment is set to last 10 minutes, at the end of which a network trace is captured using the tcpdump kernel-level packet sniffer public tool (Jacobson (1989) and McCanne e Jacobson (1993)). The Tcpdump network packet analyzer executes at the server machine. Selecting a duration of 10 minutes for each experiment is estimated to provide sufficient time to extract meaningful insights and identify possible transmission patterns. Running longer experiments has shown no advantages while it generates unnecessarily large packet capture files. The number of cameras is gradually increased from 1 to 4 cameras. Observe that in the case of configuration C (Fig. 24), increasing the number of cameras also increases the number of the edge devices, as each camera is directly connected to an edge device. Table 7 gathers the network parameters adopted in the experiments.

Table 7 – Parameters of network experiments.

| Parameter | Description |
|---|---|
| Transmission Resolution | 640x480 px |
| Transmission Frame rate | 15 FPS |
| Execution time | 10 min |
| Number of cameras | 1, 2, 3 and 4 |
| Network configuration | A, B and C (Figure 24) |

Source: The author.

Table 8 collects the results generated by the suggested experiments. It shows that the inclusion of a an intermediate processing unit (Raspberry Pi) as an edge device considerably impacts the ARTT. Observe that ARTT deteriorates even further when increasing the number of these simple ARM-based devices. Setups A and B (i.e., without using edge devices) exhibit no significant difference with regard to this metric. In the case of setup C, ARTT returns a more significant standard deviation due to the reconnections that may occur. As the ImageZMQ software responsible for forwarding captured images by Raspberry devices requires an ACK response for each frame, reconnections can occur when ACK packets are lost from the server, and there is a timeout each time a reconnection event takes place. Therefore, configuration C is more sensitive to problems inherent to the WiFi network. ARTT may increase due to factors such as the interference caused by nearby WiFi networks, noise caused by nearby machines and persons operating in a confined space, as well due to increased traffic from other competing connected devices. Observe that the results show a high ARTT when using three cameras as this scenario considerably suffered from packet loss.

The number of exchanged packets increases by approximately 75000 for Configurations A and B, while it increases by as much as 130000 in the case of Configuration C. This last configuration manages the transfer of a more significant number of packets from a single camera. Although the number of transmitted packets increases linear, more packets are needed to connect the first camera. The increase in the total length of transferred data follows the same trend as that of the number of packets. Observe that the wired connection, using Ethernet, is loss-free, independently of the number of used cameras. On the other hand, the WiFi configuration suffers some level of packet loss. For example, the configuration with the edge device suffered a maximum packet loss of 0.29% when using

Table 8 – Network results.

| Qty cams | Config | Mean ARTT/ packet (ms) | qty packets | packet loss | % loss | Length (MB) |
|---|---|---|---|---|---|---|
| 1 | A | 0.03±0.18 | 63746 | 0 | 0.00% | 364.35 |
| | B | 0.04±0.21 | 69088 | 258 | 0.37% | 342.66 |
| | C | 0.15±5.69 | 161830 | 246 | 0.15% | 713.21 |
| 2 | A | 0.03±0.02 | 142637 | 0 | 0.00% | 678.38 |
| | B | 0.03±0.21 | 125440 | 400 | 0.32% | 663.84 |
| | C | 0.14±3.48 | 298281 | 569 | 0.19% | 1041.89 |
| 3 | A | 0.03±0.24 | 217571 | 0 | 0.00% | 1026.63 |
| | B | 0.03±0.2 | 194207 | 384 | 0.20% | 1051.26 |
| | C | 0.34±8.73 | 408447 | 1178 | 0.29% | 1249.31 |
| 4 | A | 0.03±0.19 | 294433 | 0 | 0.00% | 1380.00 |
| | B | 0.04±0.49 | 270021 | 568 | 0.21% | 1418.29 |
| | C | 0.21±4.67 | 533296 | 402 | 0.08% | 1662.59 |

Source: The author.

three devices. Note that this packet loss level does not significantly affect the transmission quality (JR; ANTONAKOS, 2009). The use of wireless communications in the context of IIoT has advantages that outweigh such low levels of packet loss. Cabling is often difficult and cumbersome to install on a factory floor and restricts flexibility and machine movement.

The results show that using the inclusion of edge devices (Raspberry Pi) led to a degradation in the network transmission of flows compared to the other configurations that did not use such a device. Some factors for this are the lack of optimization of the retransmission libraries of frames made for the general purpose processor and the limitation of the network card of the embedded device, while the cameras are already optimized for the networked stream. Although, it is not possible to say that this is directly linked to using Raspberry Pi because it would be necessary to assess whether other machines replacing the raspberry would bring the same result. However, this impact is directly linked to the inclusion of an intermediate processing node between the camera and the server.

Nonetheless, the network does not result in a significant packet loss of information considering the intended application. Hence given our application context, the most significant benefit of adopting such devices is their ability to process the video streams locally and relieve the network from their transmission, saving resources and allowing more scalable deployments. Our application takes advantage of this result in a simple manner. As intuitively for an industry to operate, it is expected that there are mostly accident-free periods where no risk or safety issues are raised. It becomes reasonable to carry out processing at a local level, i.e., at the edge device itself. In the rare event of a safety risk being detected, the relevant data must be transmitted from these devices to a central server for further processing.

Additional advantages of using an edge device, such as a Raspberry Pi, include the fact that an edge device may be made to operate under different system configurations. Unlike the used simple cameras, a Raspberry Pi device may, for example, alter frame size and the compression rate parameters. Finally, transmission efficiency may be improved using this type of device by upgrading its processing power and optimizing the installed software libraries, especially those related to the TCP/IP stack.

### 5.4.3 Attention

Previous experiments independently evaluated the impact of the edge device on the network and the accuracy of the proposed architecture while increasing the number of devices. Therefore, this experiment aims to evaluate the accuracy and impact of the network together when applied to the proposed use case. However, here the assessments focus on the attention module, which is responsible for interrupting transmission when necessary to avoid network overload and reduce traffic by removing what it sees as being unnecessary flows.

Three configurations for the attention module are considered for assessment. The first is the case without attention, which represents a continuous transmission of frames on the network without stopping any flow. The other scenarios consider the two different *Attention module* modes presented in Section 4.3.5, called active and passive attention. Active attention is applied in the second scenario. Edge agents turn off local processing and send frames to the server when a risk is identified locally or when the server alerts the edge that there is a risk; otherwise (under safe conditions), they process locally and stop transmitting frames to the server. The third scenario applies Passive attention, in which, in a risky situation, the server selects the most relevant among the connected edge devices to continue transmitting and stops the others from transmitting.

Note that Active and Passive attention only differ when the risk is identified. For both, when it is safe, there is no transmission. However, when there is a risk, the Active attention requests all agents to transmit, while the Passive attention selects only the most relevant agents to transmit and stops the others.

Passive attention requires some additional parameters to be configured on the agent: the queue size, threshold, and sleep time. The queue size is set to 3, which is the minimum value to have a tiebreaker when different data is sent. The adopted threshold is 0.5, so the agent maintains the transmission when at least two correct data items are queued. Also, the sleep time is set to 3 seconds, which is a period that allows for a reassessment of the risk window for the dataset. This value is not too low to have many reconnection events, which is when the connection between the edge and the server is lost and needs to be re-established.

The parameters applied are shown in Table 9. This experiment considers the inclusion of four agents with all four cameras, as shown in Fig. 19.

Table 9 – Attention experiment parameters.

| Parameter | Description |
|---|---|
| Frames Resolution | 320x320 px |
| Frame rate | 10 FPS |
| Number of executions | 30 |
| Execution time | 30 seconds |
| Cameras configuration | 4 cameras |
| Attention | Without, Active, Passive |

Source: The author.

The evaluations consider as metrics the overrall ACC, SEN and SPE, as defined in the equations 5.3, 5.4 and 5.5 respectively. Also we evaluate the ARTT, number of packets, total length of data and packet loss as in Section 5.4.2.

Table 10 gathers the results for the accuracy of the considered attentions. The case without attention is similar to what we have in Table 6 for the situation using model YV3 and configuration 1-2-3-4. As explained in Section 5.4.1, it has lower accuracy than the others (Active and Passive attentions) because it does not benefit from the previous processing by the MN model, which ends up identifying more safe states because of camera 4. The lower SPE of the YV3 compared to the MN for camera 4 indicates this.

Table 10 – Accuracy results for different attentions.

| Attention | ACC | SEN | SPE |
|---|---|---|---|
| Without | $0.79 \pm 0.02$ | $0.98 \pm 0.02$ | $0.70 \pm 0.02$ |
| Active | $0.85 \pm 0.03$ | $0.95 \pm 0.06$ | $0.80 \pm 0.06$ |
| Passive | $0.88 \pm 0.04$ | $0.93 \pm 0.07$ | $0.86 \pm 0.04$ |

Source: The author.

Active and Passive attentions have standard deviations that overlap for all metrics, and both bring greater accuracy as opposed to not having attention. The passive approach has a slight advantage over the active one, which remains with the same standard deviation. This shows that applying agent selection and stopping those that have no relevance does not negatively impact our use case's accuracy while considerably saving processing and networking resources.

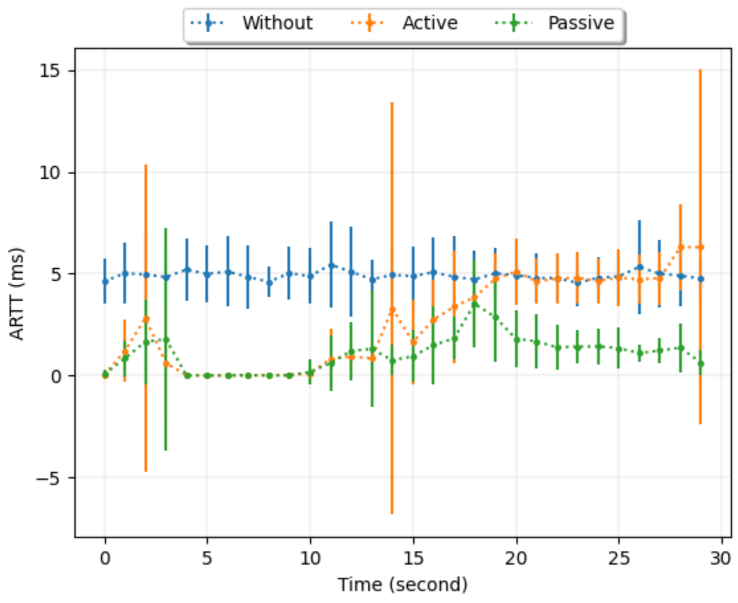
The network evaluation results for the different attentions is demonstrated in Table 10. Figs. 25, 27 and 26 have respectively the average ARTT, number of packets and data length per second achieved for the adopted dataset. The vertical bars represent the standard deviations of the means per second.

Table 11 – Network results for different attentions.

| Attention | Mean ARTT/ packet (ms) | qty packets | packet loss | % loss | Length (MB) |
|---|---|---|---|---|---|
| Without | 4.94±0.31 | 6384.03±247.85 | 0.03±0.18 | 0.00% | 10.48±0.08 |
| Active | 4.31±0.53 | 3519.77±422.47 | 0.03±0.18 | 0.00% | 5.63±0.62 |
| Passive | 1.78±0.46 | 2766.13±325.60 | 0.00±0.00 | 0.00% | 4.03±0.45 |

Source: The author.

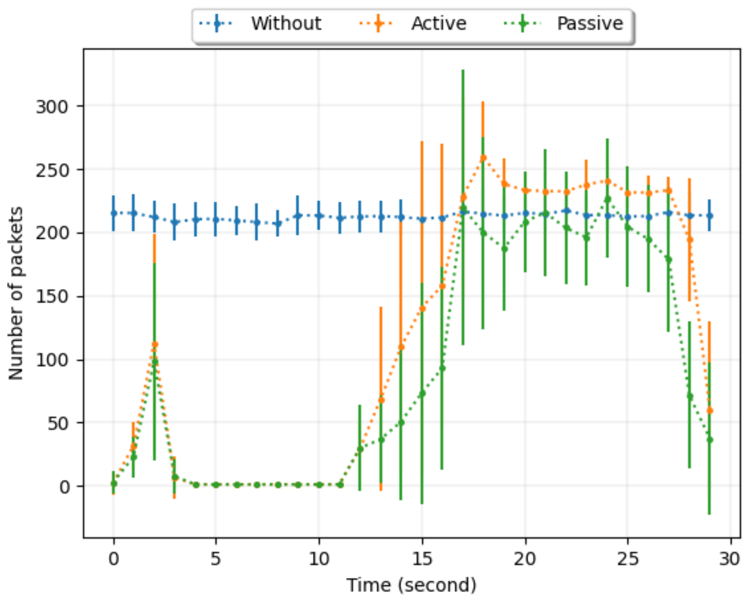
Figure 25 – ARTT per second.



Source: The author.

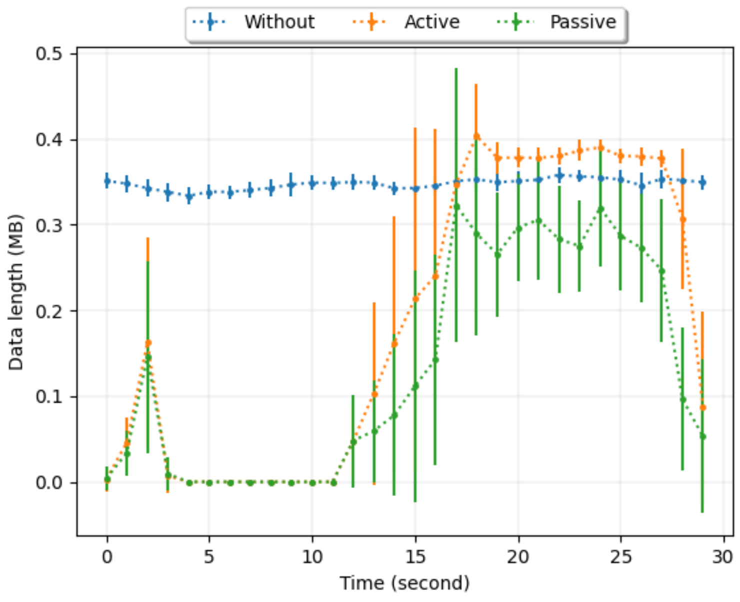
The configuration without attention suffers the highest levels for all metrics. This includes suffering higher ARTT as frames are transmitted all the time, which causes large packet queuing, impacting with delay in receiving associated acknowledgments. Active attention reduces ARTT. At first glance, such reduction cannot be considered relevant enough compared to when not using attention (in the without attention case), as they exhibit an intersection in the same standard deviation range. Although, as can be seen in Fig. 25, active attention shows at a few instants (seconds 3, 14, and 30) an unusual standard deviation that represents outliers. These occur mainly in the transition of states, from *Safe* to *Risk* or vice-versa. This is due to the rare situations where agents have to establish reconnections. The passive approach presents a reduction of 63.97% in terms

Figure 26 – Number of packets per second.



Source: The author.

Figure 27 – Data length per second.



Source: The author.

of the ARTT metric in relation to no attention and a reduction of 58.70% in relation to active attention.

In the without attention setup, the architecture transmits more packets and consequently reaches a higher data length (total amount of transmitted data). This is because the other attention modes interrupt the transmission even when there is no risk. Despite this, neither approach has a significant packet loss level. Active attention reduces the number of transmitted packets by 44.87% and the data length by 53.67% compared to when using without attention. Also, the passive attention mode reduces the number of packets by 46.33% and the data length by 61.52% compared to without attention.

Figs. 27 and 26 show that from second 4 to 11 there is no data being exchanged. This should happen from 0 to 15 seconds, in an ideal case, as the dataset considers safe situations in this period. However, from second 0 to 4, camera 4's perspective sometimes detects risk when person 2 passes in front of it, which causes the graph to rise in that period. Instants from seconds 11 to 15 correspond to when person 1 enters the overlay area for cameras 2 and 4 perspectives, representing the period of increase in packet number and data length. From seconds 11 to 25, the state switches from *Risk* to *Safe*, which is when such metrics begin to decline. In the period from 15s to 25s, when there is *Risk*, active attention is slightly worse than no attention due to resetting in real-time on/off the transmission. Passive attention has more advantages than others, as it has the selection of some agents to interrupt transmission due to their relevance. However, with active and passive attention, the data has more dispersion due to variations caused by toggling agents states which have to restart transmission and select agents at different times for different runs. In contrast, when using a without attention setup, the latter maintains a single connection throughout the experiment.

## 5.5 DISCUSSION

In this section, a prototype of the proposed architecture was designed and evaluated to identify risks in an industrial scenario. It represents a multi-camera vision system and applies real-time object detection models to human detection. In general, it was presented that the inclusion of more cameras presents benefits by adding different perspectives for a more understandable view of the environment by the system. In addition, the evaluation showed that the proposal of this work does not affect the accuracy. However, adding more cameras causes more traffic to be added to the network, which is also impacted by the inclusion of the edge device to intermediate the transmission. Nevertheless, the proposed solution reduces traffic by including the attention mechanism.

The application of edge devices for streaming video is evaluated considering network metrics. Its inclusion impacts the network negatively when in continuous transmission. However, our architecture can prevent edge devices from transmitting all the time and only transmits the raw data to be processed on the server in case of a potential *Risk*, which is less likely to occur than in situations judged as being *Safe*.

The inclusion of the attention mechanism to select the most relevant agents when there is risk was also evaluated. Compared to the continuous transmission of frames from edge devices, its inclusion results in a 63.97% reduction in network delay, 56.67% reduction in the number of packets, and a 61.52% reduction in the total data length. Furthermore, the architecture's inclusion maintains the accuracy of the models close to the one achieved by a centralized baseline configuration that does not rely on underlying network infrastructure.

# 6 FINAL CONSIDERATIONS

This work presented and discussed the construction of a reference architecture for MAS-based systems considering IIoT and a security risk application. The proposal presents an edge/cloud computing structure adopted to distribute and parallelize sensor data processing in an industrial environment. The proposed architecture avoids the centralization of transmission and processing of information to a local server. One of the main goals of the proposed architecture was to reduce the network load and eventually assess whether more mission-critical applications could co-exist and share network resources with the introduction of additional devices.

The evaluated scenario showed that, compared to conventional solutions where images are continuously streamed to a cloud infrastructure, the inclusion of edge devices generates more traffic during transmission. Part of the increased traffic can be justified by the less efficient protocols used in the current implementation of the architecture and the choice of Raspberry Pi as the edge device. However, it does not continuously transmit, using network resources only when needed. Nonetheless, given that safety occurrences are relatively rare, one can argue that the overall reduction in network load is justifiable, attainable, and concrete.

However, such reduction is not useful if the accuracy becomes compromised. In this respect, the research has also evaluated the accuracy of different risk detection models. Due to their limited computational power, edge devices must necessarily use simpler detection algorithms. The results suggest that it is possible to find a balanced compromise that makes the edge/cloud solution proposed useful. Indeed the higher computational power of cloud resources, particularly using GPU processing, enables very high processing rates. However, these come at the cost of network load. The positioning of the sensors matters naturally, but the results also show the benefits of the sensorial fusion offered by the platform. Applying a mechanism to select the most relevant transmitting agents brings advantages by reducing the delay, the number of packets, and the total amount of transmitted data (length), compared to a continuous transmission.

## 6.1 CONTRIBUTIONS

From what was discussed earlier, the following contributions can be pointed out:

- A MAS-based reference architecture for IIoT edge-cloud computing.
- A system to detect collision risks in industrial environments Multi-camera.
- A dataset that describes a human interaction activity in the industry.

- The evaluation of the utilization of edge devices to retransmit camera stream in a WiFi network.

Table 12 presents works published as a result of the research proposed in this thesis.

Table 12 – Published paper.

| Reference | Title |
|---:|---|
| Barbosa et al. (2022) | HOSA: An End-to-End Safety System for Human-Robot Interaction |
| Rodrigues et al. (2022) | Modeling and assessing an intelligent system for safety human-robot collaboration using deep and machine learning techniques |
| Filho et al. (2022) | Experimental network performance evaluation for human-robot interaction collision detection using cameras |
| Rodrigues et al. (2021b) | A New Mechanism for Collision Detection in Human-Robot Collaboration using Deep Learning Techniques |
| Barbosa, Sadok e Ribeiro (2022) | Improving Network Load using a Cloud-Edge MAS-based Architecture for ... |
| Silva et al. (2020) | Assessing Deep Learning Models for Human-Robot Collaboration Collision Detection in Industrial Environments |
| Lynn et al. (2020) | The Internet of Things: Definitions, Key Concepts, and Reference Architectures |

Source: The author.

## 6.2 LIMITATIONS

Despite the benefits of using the architecture, some limitations can be discussed. First, edge devices add more packet and data traffic to the network. This fact may occur due to the use of less efficient protocols used in the current implementation and the choice of edge devices. However, the architecture intends not to broadcast continuously, limiting itself to when a risk is identified. In this case, network resources are used only when necessary. Hazardous occurrences are relatively rare.

Another limitation can be found in implementing interfaces and system configuration by including rules which require human intervention. These human interventions can produce malformed rules, which, although the system can identify syntax and semantic structure errors, it cannot understand the security needs of users and identify if the configured rules really reproduce this need. It also depends on humans to design, analyze and connect the infrastructure, which can be underestimated and result in delays in response time.

## 6.3 FUTURE WORKS

The evolution of this system requires conducting a broader analysis with experts from different industries to understand their specific system requirements. Such interaction helps to analyze the workflow to insert the proposed system in different workplaces. It also helps to assess the worker's needs and identify possible fears regarding the presence of an automatic system for safety purposes. In addition, it is important to analyze the practicality of stakeholders in defining rules and configuring the different agents.

Regarding network infrastructure optimization, we plan to introduce and analyze the impact of using new technologies such as Software Defined Networks to adapt the network depending on the agent's decision automatically. For example, the software could configure camera traffic to have a higher priority than concurrent background data traffic. This is highly beneficial in case of network congestion. Agents can also ask the network to dynamically change the priority and bandwidth sharing of flows from devices that help detect a potential risk.

In addition, future work is directed towards applying the architecture to more complex chained infrastructure, such as an edge-fog-cloud network. Finally, it is intended to evaluate the applicability of different actuators integrated into the architecture for an automatic reaction to risk cases or improvement in perception, for example controlling PTZ cameras to direct their attention to the most relevant areas of the scenario.

# REFERENCES

ABAS, K.; OBRACZKA, K.; MILLER, L. Solar-powered, wireless smart camera network: An iot solution for outdoor video monitoring. *Computer Communications*, Elsevier, v. 118, p. 217–233, 2018.

ABNT. Iso 12100: 2013. *Segurança de máquinas — Princípios gerais de projeto — Apreciação e redução de riscos*, ABNT - Associação Brasileira de Normas Técnicas, 2013.

ADOLPHS, P.; BEDENBENDER, H.; DIRZUS, D.; EHLICH, M.; EPPLE, U.; HANKEL, M.; HEIDEL, R.; HOFFMEISTER, M.; HUHLE, H.; KÄRCHER, B. et al. Reference architecture model industrie 4.0 (rami4. 0). *ZVEI and VDI, Status report*, 2015.

ALOMAN, A.; ISPAS, A.; CIOTIRNAE, P.; SANCHEZ-IBORRA, R.; CANO, M.-D. Performance evaluation of video streaming using mpeg dash, rtsp, and rtmp in mobile networks. In: IEEE. *2015 8th IFIP Wireless and Mobile Networking Conference (WMNC)*. [S.l.], 2015. p. 144–151.

ASKARPOUR, M.; MANDRIOLI, D.; ROSSI, M.; VICENTINI, F. Formal model of human erroneous behavior for safety analysis in collaborative robotics. *Robotics and Computer-Integrated Manufacturing*, v. 57, p. 465 – 476, 2019. ISSN 0736-5845. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0736584518303247>.

AVIDAN, S. Ensemble tracking. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 29, n. 2, p. 261–271, 2007.

AZIZ, A.; AHMED, S.; KHAN, F. I. An ontology-based methodology for hazard identification and causation analysis. *Process Safety and Environmental Protection*, Elsevier, v. 123, p. 87–98, 2019.

BAARS, B. J. In the theatre of consciousness. global workspace theory, a rigorous scientific theory of consciousness. *Journal of Consciousness Studies*, Imprint Academic, v. 4, n. 4, p. 292–309, 1997.

BABENKO, B.; YANG, M.-H.; BELONGIE, S. Visual tracking with online multiple instance learning. In: IEEE. *2009 IEEE Conference on computer vision and Pattern Recognition*. [S.l.], 2009. p. 983–990.

BARBOSA, G.; LEDEBOUR, C.; FILHO, A. T. de O.; RODRIGUES, I. R.; SADOK, D.; KELNER, J.; SOUZA, R. Hosa: An end-to-end safety system for human-robot interaction. *Journal of Intelligent & Robotic Systems*, Springer, v. 105, n. 4, p. 1–24, 2022.

BARBOSA, G.; SADOK, D.; RIBEIRO, L. Improving network load using a cloud-edge mas-based architecture. 2022.

BASS, J. *ImageZMQ: Transporting OpenCV images.* [S.l.]: GitHub, 2020. <https://github.com/jeffbass/imagezmq>.

BELL, R. Introduction to iec 61508. v. 162, p. 3–12, 2006.

BERARDINI, D.; MANCINI, A.; ZINGARETTI, P.; MOCCIA, S. Edge artificial intelligence: A multi-camera video surveillance application. In: AMERICAN SOCIETY OF MECHANICAL ENGINEERS. *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. [S.l.], 2021. v. 85437, p. V007T07A006.

BORGO, S.; CESTA, A.; ORLANDINI, A.; UMBRICO, A. Knowledge-based adaptive agents for manufacturing domains. *Engineering with Computers*, Springer, v. 35, n. 3, p. 755–779, 2019.

BORMANN, C.; CASTELLANI, A. P.; SHELBY, Z. Coap: An application protocol for billions of tiny internet nodes. *IEEE Internet Computing*, IEEE, v. 16, n. 2, p. 62–67, 2012.

BOSSE, E.; ROY, J.; GRENIER, D. Data fusion concepts applied to a suite of dissimilar sensors. In: IEEE. *Proceedings of 1996 Canadian Conference on Electrical and Computer Engineering*. [S.l.], 1996. v. 2, p. 692–695.

BOYES, H.; HALLAQ, B.; CUNNINGHAM, J.; WATSON, T. The industrial internet of things (iiot): An analysis framework. *Computers in industry*, Elsevier, v. 101, p. 1–12, 2018.

BRACHMAN, R. J.; LEVESQUE, H. J. Knowledge representation and reasoning. *Morgan Kaufmann Publishers, Massachusetts, US*, v. 9, p. 9, 2004.

BRADSKI, G.; KAEHLER, A. Opencv. *Dr. Dobb's journal of software tools*, v. 3, p. 2, 2000.

BRIDEWELL, W.; BELLO, P. Incremental object perception in an attention-driven cognitive architecture. In: *CogSci*. [S.l.: s.n.], 2015.

BYNER, C.; MATTHIAS, B.; DING, H. Dynamic speed and separation monitoring for collaborative robot applications – concepts and performance. *Robotics and Computer-Integrated Manufacturing*, v. 58, p. 239 – 252, 2019. ISSN 0736-5845. Disponível em: <http://www.sciencedirect.com/science/article/pii/S073658451830259X>.

CAMPERO-JURADO, I.; MÁRQUEZ-SÁNCHEZ, S.; QUINTANAR-GÓMEZ, J.; RODRÍGUEZ, S.; CORCHADO, J. M. Smart helmet 5.0 for industrial internet of things using artificial intelligence. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 20, n. 21, p. 6241, 2020.

CHANDRASEKARAN, B.; CONRAD, J. M. Human-robot collaboration: A survey. p. 1–8, 2015.

CHEN, J.; LI, K.; DENG, Q.; LI, K.; PHILIP, S. Y. Distributed deep learning model for intelligent video surveillance systems with edge computing. *IEEE Transactions on Industrial Informatics*, IEEE, 2019.

CHEN, S.; ZHANG, S.; SHANG, J.; CHEN, B.; ZHENG, N. Brain-inspired cognitive model with attention for self-driving cars. *IEEE Transactions on Cognitive and Developmental Systems*, IEEE, v. 11, n. 1, p. 13–25, 2019.

DANIYAL, F.; CAVALLARO, A. Multi-camera scheduling for video production. In: IEEE. *2011 Conference for Visual Media Production*. [S.l.], 2011. p. 11–20.

DANIYAL, F.; TAJ, M.; CAVALLARO, A. Content and task-based view selection from multiple video streams. *Multimedia tools and applications*, Springer, v. 46, n. 2, p. 235–258, 2010.

de Souza Baptista, C.; Vasconcelos, K. F.; Arruda, L. S. Ontoeditor: a web tool for manipulating ontologies stored in database servers. In: *Proceedings. 15th International Workshop on Database and Expert Systems Applications, 2004.* [S.l.: s.n.], 2004. p. 151–155.

DENTLER, K.; CORNET, R.; TEIJE, A. T.; KEIZER, N. D. Comparison of reasoners for large ontologies in the owl 2 el profile. *Semantic Web*, IOS Press, v. 2, n. 2, p. 71–87, 2011.

DESHPANDE, S.; SOWMYA, A.; YADAV, P.; LADHA, S.; VERMA, P.; VAIAPURY, K.; GUBBI, J.; BALAMURALIDHAR, P. Cogvis: Attention-driven cognitive architecture for visual change detection. In: *Proceedings of the Symposium on Applied Computing.* New York, NY, USA: Association for Computing Machinery, 2017. (SAC '17), p. 151–154. ISBN 9781450344869. Disponível em: <https://doi.org/10.1145/3019612.3019857>.

DROETTBOOM, M. et al. Understanding json schema. *Available on: http://spacetelescope. github. io/understanding-jsonschema/UnderstandingJSONSchema. pdf (accessed on 14 April 2014)*, 2015.

ECLIPSE Paho MQTT Python client library. Python Software Foundation, 2022. Disponível em: <https://pypi.org/project/paho-mqtt/>.

ELMENREICH, W. An introduction to sensor fusion. *Vienna University of Technology, Austria*, v. 502, p. 1–28, 2002.

FAGHIHI, U.; MCCALL, R.; FRANKLIN, S. A computational model of attentional learning in a cognitive agent. *Biologically Inspired Cognitive Architectures*, v. 2, p. 25 – 36, 2012. ISSN 2212-683X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S2212683X12000321>.

FERNÁNDEZ-LÓPEZ, M.; GÓMEZ-PÉREZ, A.; JURISTO, N. Methontology: from ontological art towards ontological engineering. American Asociation for Artificial Intelligence, 1997.

FFmpeg. *FFmpeg.* 2022. Disponível em: <https://ffmpeg.org/>.

FILHO, A. T. de O.; BARBOSA, G.; RODRIGUES, I. R.; CANI, C.; KELNER, J.; SADOK, D.; SOUZA, R. Experimental network performance evaluation for human-robot interaction collision detection using cameras. *Research, Society and Development*, v. 11, n. 8, p. e8811830543–e8811830543, 2022.

FILTEAU, J.; LEE, S. J.; JUNG, A. Real-time streaming application for iot using raspberry pi and handheld devices. In: IEEE. *2018 IEEE Global Conference on Internet of Things (GCIoT).* [S.l.], 2018. p. 1–5.

FRAILE, F.; SANCHIS, R.; POLER, R.; ORTIZ, A. Reference models for digital manufacturing platforms. *Applied Sciences*, v. 9, n. 20, 2019. ISSN 2076-3417. Disponível em: <https://www.mdpi.com/2076-3417/9/20/4433>.

Franklin, S.; Madl, T.; D'Mello, S.; Snaider, J. Lida: A systems-level architecture for cognition, emotion, and learning. *IEEE Transactions on Autonomous Mental Development*, v. 6, n. 1, p. 19–41, 2014.

FUGINI, M.; RAIBULET, C.; UBEZIO, L. Risk assessment in work environments: modeling and simulation. *Concurrency and computation: Practice and experience*, Wiley Online Library, v. 24, n. 18, p. 2381–2403, 2012.

GALL, H. Functional safety iec 61508/iec 61511 the impact to certification and the user. p. 1027–1031, 2008.

GAO, T.; BAKER, C. L.; TANG, N.; XU, H.; TENENBAUM, J. B. The cognitive architecture of perceived animacy: Intention, attention, and memory. *Cognitive science*, Wiley Online Library, v. 43, n. 8, p. e12775, 2019.

GEORGE, A.; RAVINDRAN, A.; MENDIETA, M.; TABKHI, H. Mez: A messaging system for latency-sensitive multi-camera machine vision at the iot edge. *arXiv preprint arXiv:2009.13549*, 2020.

GIACINTO, G.; ROLI, F.; DIDACI, L. Fusion of multiple classifiers for intrusion detection in computer networks. *Pattern recognition letters*, Elsevier, v. 24, n. 12, p. 1795–1803, 2003.

GIBAUT, W. S. P. *Uma arquitetura cognitiva para o aprendizado instrumental em agentes inteligentes*. Tese (Doutorado) — Ph. D. thesis, DCA-FEEC-UNICAMP, 2018.

GLIMM, B.; HORROCKS, I.; MOTIK, B.; STOILOS, G.; WANG, Z. Hermit: an owl 2 reasoner. *Journal of Automated Reasoning*, Springer, v. 53, n. 3, p. 245–269, 2014.

GNONI, M. G.; BRAGATTO, P. A.; MILAZZO, M. F.; SETOLA, R. Integrating iot technologies for an "intelligent" safety management in the process industry. *Procedia manufacturing*, Elsevier, v. 42, p. 511–515, 2020.

GOPINATH, V.; JOHANSEN, K.; ÖLVANDER, J. *Risk assessment for collaborative operation: a case study on hand-guided industrial robots*. [S.l.]: InTech, 2018.

GRUBER, T. R. Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies*, Elsevier, v. 43, n. 5-6, p. 907–928, 1995.

HASHIMOTO, S.; JI, Y.; KUDO, K.; TAKAHASHI, T.; UMEDA, K. Anomaly detection based on deep learning using video for prevention of industrial accidents. *arXiv preprint arXiv:2005.13734*, 2020.

HAYEK, A.; TELAWI, S.; KLOS, J.; BÖRCSÖK, J.; DAOU, R. A. Z. Smart wearable system for safety-related industrial iot applications. In: *Interoperability, Safety and Security in IoT*. [S.l.]: Springer, 2017. p. 154–164.

HEIDEL, R. *Industrie 4.0: the reference architecture model RAMI 4.0 and the Industrie 4.0 component*. [S.l.]: Beuth Verlag GmbH, 2019.

HENRIQUES, J. F.; CASEIRO, R.; MARTINS, P.; BATISTA, J. High-speed tracking with kernelized correlation filters. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 37, n. 3, p. 583–596, 2014.

HORROCKS, I.; PATEL-SCHNEIDER, P. F.; BOLEY, H.; TABET, S.; GROSOF, B.; DEAN, M. et al. Swrl: A semantic web rule language combining owl and ruleml. *W3C Member submission*, v. 21, n. 79, p. 1–31, 2004.

HU, Y. C.; PATEL, M.; SABELLA, D.; SPRECHER, N.; YOUNG, V. Mobile edge computing—a key technology towards 5g. *ETSI white paper*, v. 11, n. 11, p. 1–16, 2015.

IGLESIA, D. G. D. L.; WEYNS, D. Mape-k formal templates to rigorously design behaviors for self-adaptive systems. *ACM Trans. Auton. Adapt. Syst.*, Association for Computing Machinery, New York, NY, USA, v. 10, n. 3, set. 2015. ISSN 1556-4665. Disponível em: <https://doi.org/10.1145/2724719>.

IMPORTLIB – An implementation of import. Python Software Foundation, 2022. Disponível em: <https://docs.python.org/3/library/importlib.html>.

INAM, R.; RAIZER, K.; HATA, A.; SOUZA, R.; FORSMAN, E.; CAO, E.; WANG, S. Risk assessment for human-robot collaboration in an automated warehouse scenario. v. 1, p. 743–751, 2018.

INSURANCE, L. M. *Liberty Mutual Workplace Safety Index, Risk Control services from Liberty Mutual Insurance.* 2021.

Islam, N.; Siddiqui, M. S.; Shaikh, Z. A. Tode : A dot net based tool for ontology development and editing. In: *2010 2nd International Conference on Computer Engineering and Technology.* [S.l.: s.n.], 2010. v. 6, p. V6–229–V6–233.

ISO. Iso 31000: 2009. *Risk management– Guideline*, 2009.

ISO, E. *Safety of machinery–General principles for design–Risk assessment and risk reduction (ISO 12100:2010).* [S.l.]: ISO, 2010.

ISO, E. *Robots and robotic devices — Safety requirements for industrial robots — Part 1: Robots (ISO 10218-1:2011).* [S.l.]: ISO, 2011.

ISO, E. *Robots and robotic devices — Safety requirements for industrial robots — Part 2: Robot systems and integration (ISO 10218-2:2011).* [S.l.]: ISO, 2011.

ISO, E. *Robots and robotic devices — Collaborative robots (ISO/TS 15066:2016).* [S.l.]: ISO, 2016.

JACOBSON, V. Tcpdump. *ftp://ftp. ee. lbl. gov*, 1989.

JENA, A. A free and open source java framework for building semantic web and linked data applications. *Available online: jena. apache. org/(acessado em 02 de junho de 2020)*, 2020.

JR, K. C. M.; ANTONAKOS, J. L. *Computer networking for LANS to WANS: hardware, software and security.* [S.l.]: Cengage Learning, 2009.

KALAL, Z.; MIKOLAJCZYK, K.; MATAS, J. Forward-backward error: Automatic detection of tracking failures. In: IEEE. *2010 20th international conference on pattern recognition.* [S.l.], 2010. p. 2756–2759.

KALAL, Z.; MIKOLAJCZYK, K.; MATAS, J. Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 34, n. 7, p. 1409–1422, 2011.

KAPLAN, F.; HAFNER, V. V. The challenges of joint attention. *Interaction Studies*, John Benjamins, v. 7, n. 2, p. 135–169, 2006.

KHAKZAD, N.; KHAN, F.; AMYOTTE, P. Dynamic safety analysis of process systems by mapping bow-tie into bayesian network. *Process Safety and Environmental Protection*, Elsevier, v. 91, n. 1-2, p. 46–53, 2013.

KHAN, A.; KHATTAK, K. S.; KHAN, Z. H.; GULLIVER, T.; IMRAN, W.; MINALLAH, N. Internet-of-video things based real-time traffic flow characterization. *EAI Endorsed Transactions on Scalable Information Systems*, EAI, v. 8, n. 33, p. e9, 2021.

KHAN, W. Z.; REHMAN, M.; ZANGOTI, H. M.; AFZAL, M. K.; ARMI, N.; SALAH, K. Industrial internet of things: Recent advances, enabling technologies and open challenges. *Computers & Electrical Engineering*, Elsevier, v. 81, p. 106522, 2020.

KIRTON, D. Risk assessment [electric machines]. p. 5–1, 1995.

KOLLER, G. *Risk assessment and decision making in business and industry: A practical guide*. [S.l.]: CRC Press, 2005.

KUDRYAVTSEV, S. S.; YEMELIN, P. V.; YEMELINA, N. K. The development of a risk management system in the field of industrial safety in the republic of kazakhstan. *Safety and health at work*, Elsevier, v. 9, n. 1, p. 30–41, 2018.

KUMAR, S. P.; SELVAKUMARI, S.; PRAVEENA, S.; RAJIV, S. Deep learning enabled smart industrial workers precaution system using single board computer (sbc). In: *Internet of Things for Industry 4.0*. [S.l.]: Springer, 2020. p. 91–101.

KWON, J.-H.; KIM, E.-J. Accident prediction model using environmental sensors for industrial internet of things. *Sensors and Materials*, v. 31, n. 2, p. 579–586, 2019.

LAMY, J.-B. Owlready: Ontology-oriented programming in python with automatic classification and high level constructs for biomedical ontologies. *Artificial intelligence in medicine*, Elsevier, v. 80, p. 11–28, 2017.

LANE, P. C. R.; GOBET, F.; SMITH, R. L. Attention mechanisms in the chrest cognitive architecture. In: PALETTA, L.; TSOTSOS, J. K. (Ed.). *Attention in Cognitive Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 183–196.

LASKEY, K. B. Mebn: A language for first-order bayesian knowledge bases. *Artificial intelligence*, Elsevier, v. 172, n. 2-3, p. 140–178, 2008.

LASOTA, P. A.; FONG, T.; SHAH, J. A. et al. A survey of methods for safe human-robot interaction. *Foundations and Trends® in Robotics*, Now Publishers, Inc., v. 5, n. 4, p. 261–349, 2017.

LEE, E. A. Cyber physical systems: Design challenges. In: IEEE. *2008 11th IEEE international symposium on object and component-oriented real-time distributed computing (ISORC)*. [S.l.], 2008. p. 363–369.

LEE, J.; BAGHERI, B.; KAO, H.-A. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing letters*, Elsevier, v. 3, p. 18–23, 2015.

LEE, S. K.; BAE, M.; KIM, H. Future of iot networks: A survey. *Applied Sciences*, MDPI, v. 7, n. 10, p. 1072, 2017.

LI, C.; LI, J.; XIE, Y.; NIE, J.; YANG, T.; LU, Z. Multi-camera joint spatial self-organization for intelligent interconnection surveillance. *Engineering Applications of Artificial Intelligence*, v. 107, p. 104533, 2022. ISSN 0952-1976. Disponível em: <https://www.sciencedirect.com/science/article/pii/S095219762100381X>.

LI, Y.; BHANU, B. Fusion of multiple trackers in video networks. In: IEEE. *2011 Fifth ACM/IEEE International Conference on Distributed Smart Cameras*. [S.l.], 2011. p. 1–6.

LIEBIG, T. Reasoning with owl-system support and insights. Universität Ulm, 2013.

LIGHT, R. A. Mosquitto: server and client implementation of the mqtt protocol. *Journal of Open Source Software*, v. 2, n. 13, p. 265, 2017.

LIN, S.-W.; MILLER, B.; DURAND, J.; JOSHI, R.; DIDIER, P.; CHIGANI, A.; TORENBEEK, R.; DUGGAL, D.; MARTIN, R.; BLEAKLEY, G. et al. Industrial internet reference architecture. *Industrial Internet Consortium (IIC), Tech. Rep*, 2015.

LIN, S.-W.; MURPHY, B.; CLAUER, E.; LOEWEN, U.; NEUBERT, R.; BACHMANN, G.; PAI, M.; HANKEL, M. Architecture alignment and interoperability. *Industry IoT Consortium: Boston, MA, USA*, 2017.

LUKEZIC, A.; VOJIR, T.; ZAJC, L. C.; MATAS, J.; KRISTAN, M. Discriminative correlation filter with channel and spatial reliability. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 6309–6318.

LYKOURENTZOU, I.; PAPADAKI, K.; KALLIAKMANIS, A.; DJAGHLOUL, Y.; LATOUR, T.; CHARALABIS, I.; KAPETANIOS, E. Ontology-based operational risk management. In: IEEE. *2011 IEEE 13th Conference on Commerce and Enterprise Computing*. [S.l.], 2011. p. 153–160.

LYNN, T.; ENDO, P. T.; RIBEIRO, A. M. N.; BARBOSA, G. B.; ROSATI, P. The internet of things: definitions, key concepts, and reference architectures. In: *The Cloud-to-Thing Continuum*. [S.l.]: Palgrave Macmillan, Cham, 2020. p. 1–22.

MAGRINI, E.; FERRAGUTI, F.; RONGA, A. J.; PINI, F.; De Luca, A.; LEALI, F. Human-robot coexistence and interaction in open industrial cells. *Robotics and Computer-Integrated Manufacturing*, v. 61, p. 101846, 2020. ISSN 0736-5845. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0736584518303338>.

MARVEL, J. A. Performance metrics of speed and separation monitoring in shared workspaces. *IEEE Transactions on automation Science and Engineering*, IEEE, v. 10, n. 2, p. 405–414, 2013.

MAYER, S.; HODGES, J.; YU, D.; KRITZLER, M.; MICHAHELLES, F. An open semantic framework for the industrial internet of things. *IEEE Intelligent Systems*, IEEE, v. 32, n. 1, p. 96–101, 2017.

MCCANNE, S.; JACOBSON, V. The bsd packet filter: A new architecture for user-level packet capture. In: *USENIX winter*. [S.l.: s.n.], 1993. v. 46.

MCGUINNESS, D. L.; HARMELEN, F. V. et al. Owl web ontology language overview. *W3C recommendation*, v. 10, n. 10, p. 2004, 2004.

MELLA, P. *The Holonic Revolution Holons, Holarchies and Holonic Networks. The Ghost in the Production Machine*. [S.l.: s.n.], 2009. ISBN 978-88-96764-00-8.

Microsoft Dynamics 365. *2019 Manufacturing Trends Report*. 2019. Accessed: 2021-09-07.

MIRAZ, M. H.; ALI, M.; EXCELL, P. S.; PICKING, R. A review on internet of things (iot), internet of everything (ioe) and internet of nano things (iont). *2015 Internet Technologies and Applications (ITA)*, IEEE, p. 219–224, 2015.

MOLKA-DANIELSEN, J.; ENGELSETH, P.; WANG, H. Large scale integration of wireless sensor network technologies for air quality monitoring at a logistics shipping base. *Journal of Industrial Information Integration*, Elsevier, v. 10, p. 20–28, 2018.

MORGAN Stanley Automation World Industrial Automation Survey. In: . [S.l.]: AlphaWise, 2015.

MORIOKA, K.; KOVACS, S.; LEE, J.-H.; KORONDI, P. A cooperative object tracking system with fuzzy-based adaptive camera selection. *International Journal on smart sensing and intelligent systems*, Exeley Inc., v. 3, n. 3, 2017.

MPLAYER Features. 2022. <http://www.mplayerhq.hu/design7/info.html>. Accessed: 2022-07-19.

MULTIPROCESSING — Process-based parallelism. Python Software Foundation, 2022. Disponível em: <https://docs.python.org/3/library/multiprocessing.html>.

MURPHY, K. P. *Dynamic bayesian networks: representation, inference and learning*. [S.l.]: University of California, Berkeley, 2002.

MUSEN, M. A. The protégé project: a look back and a look forward. *AI matters*, ACM New York, NY, USA, v. 1, n. 4, p. 4–12, 2015.

NAIK, N. Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http. In: IEEE. *2017 IEEE international systems engineering symposium (ISSE)*. [S.l.], 2017. p. 1–7.

NING, H.; YE, X.; SADA, A. B.; MAO, L.; DANESHMAND, M. An attention mechanism inspired selective sensing framework for physical-cyber mapping in internet of things. *IEEE Internet of Things Journal*, IEEE, v. 6, n. 6, p. 9531–9544, 2019.

NOY, N. F.; MCGUINNESS, D. L. et al. *Ontology development 101: A guide to creating your first ontology*. [S.l.]: Stanford knowledge systems laboratory technical report KSL-01-05 and . . . , 2001.

OLAGOKE, A. S.; IBRAHIM, H.; TEOH, S. S. Literature survey on multi-camera system and its application. *IEEE Access*, v. 8, p. 172892–172922, 2020.

OSHACAMPUS.COM. *6 Ways to Minimize Workplace Accidents.* 2018. <https://www.oshacampus.com/blog/workplace-accidents-minimize-6-ways/>. Accessed: 2010-09-06.

OSORIO, J.; MANOTAS, D.; GARCÍA, J. Ontology-based operational risk identification in 3pl. *Research in Computing Science*, v. 147, p. 55–63, 2018.

PALACH, J. *Parallel programming with Python.* [S.l.]: Packt Publishing Ltd, 2014.

PATEL, A.; JAIN, S. Formalisms of representing knowledge. *Procedia Computer Science*, v. 125, p. 542 – 549, 2018. ISSN 1877-0509. The 6th International Conference on Smart Computing and Communications. Disponível em: <http://www.sciencedirect.com/science/article/pii/S187705091732834X>.

PENG, Y.; TAN, A.; WU, J.; BI, Y. Hierarchical edge computing: A novel multi-source multi-dimensional data anomaly detection scheme for industrial internet of things. *IEEE Access*, IEEE, v. 7, p. 111257–111270, 2019.

PEZOA, F.; REUTTER, J. L.; SUAREZ, F.; UGARTE, M.; VRGOČ, D. Foundations of json schema. In: *Proceedings of the 25th International Conference on World Wide Web.* [S.l.: s.n.], 2016. p. 263–273.

PIVOTO, D. G.; de Almeida, L. F.; da Rosa Righi, R.; RODRIGUES, J. J.; LUGLI, A. B.; ALBERTI, A. M. Cyber-physical systems architectures for industrial internet of things applications in industry 4.0: A literature review. *Journal of Manufacturing Systems*, v. 58, p. 176–192, 2021. ISSN 0278-6125. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0278612520302119>.

PRASAD, G. S. C.; PILLAI, A. S. Role of industrial iot in critical environmental conditions. In: IEEE. *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS).* [S.l.], 2018. p. 1369–1372.

PURDY, G. Iso 31000: 2009—setting a new standard for risk management. *Risk Analysis: An International Journal*, Wiley Online Library, v. 30, n. 6, p. 881–886, 2010.

RAMIREZ-PEDRAZA, R.; VARGAS, N.; SANDOVAL, C.; VALLE-PADILLA], J. L. [del; RAMOS, F. A bio-inspired model of behavior considering decision-making and planning, spatial attention and basic motor commands processes. *Cognitive Systems Research*, v. 59, p. 293 – 303, 2020. ISSN 1389-0417. Disponível em: <http://www.sciencedirect.com/science/article/pii/S138904171930508X>.

RAPTIS, T. P.; PASSARELLA, A.; CONTI, M. A survey on industrial internet with isa100 wireless. *IEEE Access*, IEEE, v. 8, p. 157177–157196, 2020.

REA, F.; METTA, G.; BARTOLOZZI, C. Event-driven visual attention for the humanoid robot icub. *Frontiers in neuroscience*, Frontiers, v. 7, p. 234, 2013.

REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

REEGU, F.; KHAN, W. Z.; DAUD, S. M.; ARSHAD, Q.; ARMI, N. A reliable public safety framework for industrial internet of things (iiot). In: IEEE. *2020 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET).* [S.l.], 2020. p. 189–193.

REILY, B.; REARDON, C.; ZHANG, H. Multi-modal sensor fusion and selection for enhanced situational awareness. In: INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. *Virtual, Augmented, and Mixed Reality (XR) Technology for Multi-Domain Operations II.* [S.l.], 2021. v. 11759, p. 117590K.

REILY, B.; ZHANG, H. Simultaneous view and feature selection for collaborative multi-robot perception. *arXiv preprint arXiv:2012.09328*, 2020.

REIS, L. P. *Coordenação em Sistemas Multi-Agente: Aplicações na Gestão Universitária e Futebol Robótico. 2003. 451 f.* Tese (Doutorado) — Tese (Doutorado)-Curso de Engenharia, Universidade do Porto, Porto, 2003 . . . , 2003.

ROBINSON, S. H. Living with the challenges to functional safety in the industrial internet of things. IET, 2019.

ROBLA-GOMEZ, S.; BECERRA, V. M.; LLATA, J. R.; GONZALEZ-SARABIA, E.; TORRE-FERRERO, C.; PEREZ-ORIA, J. Working together: A review on safe human-robot collaboration in industrial environments. *IEEE Access*, IEEE, v. 5, p. 26754–26773, 2017.

ROCHA, F.; GARCIA, G.; PEREIRA, R. F.; FARIA, H. D.; SILVA, T. H.; ANDRADE, R. H.; BARBOSA, E. S.; ALMEIDA, A.; CRUZ, E.; ANDRADE, W. et al. Rosi: A robotic system for harsh outdoor industrial inspection-system design and applications. *Journal of Intelligent & Robotic Systems*, Springer, v. 103, n. 2, p. 1–22, 2021.

RODRIGUES, I. R.; BARBOSA, G.; FILHO, A. O.; CANI, C.; DANTAS, M.; SADOK, D. H.; KELNER, J.; SOUZA, R. S.; MARQUEZINI, M. V.; LINS, S. Modeling and assessing an intelligent system for safety in human-robot collaboration using deep and machine learning techniques. *Multimedia Tools and Applications*, Springer, p. 1–27, 2021.

RODRIGUES, I. R.; BARBOSA, G.; FILHO, A. O.; CANI, C.; SADOK, D. H.; KELNER, J.; SOUZA, R.; MARQUEZINI, M. V.; LINS, S. A new mechanism for collision detection in human–robot collaboration using deep learning techniques. *Journal of Control, Automation and Electrical Systems*, Springer, p. 1–13, 2021.

RODRIGUES, I. R.; BARBOSA, G.; FILHO, A. O.; CANI, C.; DANTAS, M.; SADOK, D. H.; KELNER, J.; SOUZA, R. S.; MARQUEZINI, M. V.; LINS, S. Modeling and assessing an intelligent system for safety in human-robot collaboration using deep and machine learning techniques. *Multimedia Tools and Applications*, Springer, v. 81, n. 2, p. 2213–2239, 2022.

ROSSUM, G. V.; DRAKE, F. L. *Python 3 Reference Manual.* Scotts Valley, CA: CreateSpace, 2009. ISBN 1441412697.

RUDOY, D.; ZELNIK-MANOR, L. Viewpoint selection for human actions. *International journal of computer vision*, Springer, v. 97, n. 3, p. 243–254, 2012.

RÜSSMANN, M.; LORENZ, M.; GERBERT, P.; WALDNER, M.; JUSTUS, J.; ENGEL, P.; HARNISCH, M. Industry 4.0: The future of productivity and growth in manufacturing industries. *Boston Consulting Group*, v. 9, n. 1, p. 54–89, 2015.

SANDLER, M.; HOWARD, A.; ZHU, M.; ZHMOGINOV, A.; CHEN, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* [S.l.: s.n.], 2018. p. 4510–4520.

SHARMA, N.; SHAMKUWAR, M.; SINGH, I. The history, present and future with iot. In: *Internet of Things and Big Data Analytics for Smart Generation*. [S.l.]: Springer, 2019. p. 27–51.

SHEARER, R.; MOTIK, B.; HORROCKS, I. Hermit: A highly-efficient owl reasoner. v. 432, p. 91, 2008.

SILVA, I. R.; BARBOSA, G. B.; LEDEBOUR, C. C.; FILHO, A. T. O.; KELNER, J.; SADOK, D.; LINS, S.; SOUZA, R. Assessing deep learning models for human-robot collaboration collision detection in industrial environments. p. 240–255, 2020.

SIRIN, E.; PARSIA, B.; GRAU, B. C.; KALYANPUR, A.; KATZ, Y. Pellet: A practical owl-dl reasoner. *Journal of Web Semantics*, Elsevier, v. 5, n. 2, p. 51–53, 2007.

SISINNI, E.; SAIFULLAH, A.; HAN, S.; JENNEHAG, U.; GIDLUND, M. Industrial internet of things: Challenges, opportunities, and directions. *IEEE transactions on industrial informatics*, IEEE, v. 14, n. 11, p. 4724–4734, 2018.

SMITH, C. L.; BROOKS, D. J. Chapter 3 - security risk management. In: SMITH, C. L.; BROOKS, D. J. (Ed.). *Security Science*. Boston: Butterworth-Heinemann, 2013. p. 51 – 80. ISBN 978-0-12-394436-8.

SORO, S.; HEINZELMAN, W. A survey of visual sensor networks. *Advances in multimedia*, Hindawi, v. 2009, 2009.

SPURLOCK, S.; SOUVENIR, R. Dynamic view selection for multi-camera action recognition. *Machine Vision and Applications*, Springer, v. 27, n. 1, p. 53–63, 2016.

STANDARD, O. Mqtt version 5.0. *Retrieved June*, v. 22, p. 2020, 2019.

STD, I. Industrial communication networks—wireless communication network and communication profiles–wirelesshart; iec 62591. *International Electrotechnical Commission: Geneva, Switzerland*, 2009.

STUART, R.; PETER, N. et al. *Artificial intelligence: a modern approach*. [S.l.]: Prentice Hall Upper Saddle River, NJ, USA:, 2003.

SUN, R. Desiderata for cognitive architectures. *Philosophical Psychology*, Taylor & Francis, v. 17, n. 3, p. 341–373, 2004.

SUN, S.; ZHENG, X.; VILLALBA-DÍEZ, J.; ORDIERES-MERÉ, J. Indoor air-quality data-monitoring system: Long-term monitoring benefits. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 19, n. 19, p. 4157, 2019.

SURE, Y.; STAAB, S.; STUDER, R. On-to-knowledge methodology (otkm). In: *Handbook on ontologies*. [S.l.]: Springer, 2004. p. 117–132.

SYAIFUDIN, Y.; ROZI, I.; ARIYANTO, R.; ERFAN, R.; ADHISUWIGNJO, S. Study of performance of real time streaming protocol (rtsp) in learning systems. *International Journal of Engineering and Technology(UAE)*, v. 7, p. 216–221, 12 2018.

TEIMOURIKIA, M.; FUGINI, M. Ontology development for run-time safety management methodology in smart work environments using ambient knowledge. *Future Generation Computer Systems*, Elsevier, v. 68, p. 428–441, 2017.

TEIMOURIKIA, M.; FUGINI, M.; RAIBULET, C. Run-time security and safety management in adaptive smart work environments. In: IEEE. *2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*. [S.l.], 2017. p. 256–261.

TELAWI, S.; HAYEK, A.; BÖRCSÖK, J. Safety-related wireless communication via rf modules for industrial iot applications. In: *Interoperability, Safety and Security in IoT*. [S.l.]: Springer, 2017. p. 96–112.

TESSENS, L.; MORBEE, M.; AGHAJAN, H.; PHILIPS, W. Camera selection for tracking in distributed smart camera networks. *ACM Transactions on Sensor Networks (TOSN)*, ACM New York, NY, USA, v. 10, n. 2, p. 1–33, 2014.

THANGAVEL, D.; MA, X.; VALERA, A.; TAN, H.-X.; TAN, C. K.-Y. Performance evaluation of mqtt and coap via a common middleware. In: IEEE. *2014 IEEE ninth international conference on intelligent sensors, sensor networks and information processing (ISSNIP)*. [S.l.], 2014. p. 1–6.

TSAROUCHI, P.; MAKRIS, S.; CHRYSSOLOURIS, G. Human-robot interaction review and challenges on task planning and programming. *International Journal of Computer Integrated Manufacturing*, Taylor & Francis, v. 29, n. 8, p. 916–931, 2016.

UMA, S.; ESWARI, R. Accident prevention and safety assistance using iot and machine learning. *Journal of Reliable Intelligent Environments*, Springer, p. 1–25, 2021.

VARGHESE, B.; WANG, N.; BARBHUIYA, S.; KILPATRICK, P.; NIKOLOPOULOS, D. S. Challenges and opportunities in edge computing. In: IEEE. *2016 IEEE International Conference on Smart Cloud (SmartCloud)*. [S.l.], 2016. p. 20–26.

VERNON, D. Cognitive system. Springer, p. 100–106, 2014.

VILLANI, V.; PINI, F.; LEALI, F.; SECCHI, C. Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications. *Mechatronics*, Elsevier, v. 55, p. 248–266, 2018.

VLC Features. 2022. <https://www.videolan.org/vlc/features.html>. Accessed: 2022-07-19.

VYSOCKY, A.; NOVAK, P. Human-robot collaboration in industry. *MM Science Journal*, v. 9, n. 2, p. 903–906, 2016.

WU, F.; WU, T.; YUCE, M. R. An internet-of-things (iot) network system for connected safety and health monitoring applications. *Sensors*, MDPI, v. 19, n. 1, p. 21, 2018.

WU, Y.; LIM, J.; YANG, M.-H. Online object tracking: A benchmark. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2013.

XHAFA, F.; KILIC, B.; KRAUSE, P. Evaluation of iot stream processing at edge computing layer for semantic data enrichment. *Future Generation Computer Systems*, Elsevier, v. 105, p. 730–736, 2020.

YOUNAN, M.; HOUSSEIN, E. H.; ELHOSENY, M.; ALI, A. A. Challenges and recommended technologies for the industrial internet of things: A comprehensive review. *Measurement*, Elsevier, v. 151, p. 107198, 2020.

ZANCHETTIN, A. M.; ROCCO, P.; CHIAPPA, S.; ROSSI, R. Towards an optimal avoidance strategy for collaborative robots. *Robotics and Computer-Integrated Manufacturing*, v. 59, p. 47 – 55, 2019. ISSN 0736-5845. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0736584518302886>.

ZHANG, Z.; WU, J.; ZHANG, X.; ZHANG, C. Multi-target, multi-camera tracking by hierarchical clustering: Recent progress on dukemtmc project. *arXiv preprint arXiv:1712.09531*, 2017.

ZHOU, C.; DAMIANO, N.; WHISNER, B.; REYES, M. Industrial internet of things:(iiot) applications in underground coal mines. *Mining engineering*, NIH Public Access, v. 69, n. 12, p. 50, 2017.

ZHOU, Z.; LIAO, H.; GU, B.; HUQ, K. M. S.; MUMTAZ, S.; RODRIGUEZ, J. Robust mobile crowd sensing: When deep learning meets edge computing. *Ieee network*, IEEE, v. 32, n. 4, p. 54–60, 2018.

ZHOU, Z.-H. *Ensemble methods: foundations and algorithms.* [S.l.]: CRC press, 2012.

ZUBAL', M.; LOJKA, T.; ZOLOTOVÁ, I. Iot gateway and industrial safety with computer vision. In: IEEE. *2016 IEEE 14th International Symposium on Applied Machine Intelligence and Informatics (SAMI).* [S.l.], 2016. p. 183–186.

# APPENDIX A – ONTOLOGY FOR SAFETY IN HUMAN-ROBOT INTERACTION

There are several ways to represent knowledge as reported by Patel e Jain (2018). They include network-based representations, structures, production rules, and logic. Essential characteristics when choosing the mode of representation of knowledge are the power of expression that allows and whether it is necessary to handle exceptions, for cases where knowledge is incomplete or obtained inappropriately. Other factors to consider are whether there is well-structured documentation, whether it allows efficient reasoning and the level of complexity for implementing a particular strategy.

The purpose of this chapter is to describe HRI scenes in order to anticipate the identification of imminent risk situations for humans. Therefore, we need an efficient and robust vehicle to transmit this knowledge and measure risk. Ontology in computer science and artificial intelligence is a formal tool for encoding knowledge that has been applied in many domains in recent years. Declarative ontology languages also allow the inclusion of reasoning rules that enable knowledge processing and are often based on first-order logic or description logic. The Ontology Web Language (OWL) is used in this work. It represents a family of knowledge representation languages capable of creating several ontologies. In addition to reasoning, ontologies are highly flexible as they can handle heterogeneous data sources. Currently, the W3C's OWL 2.0 is supported by many software tools, including the well-known Protégé semantic editor, as well as other common reasoners (Musen (2015)). Ontologies support reasoning through OWL, among other benefits listed in Patel e Jain (2018), such as interoperability, sharing, balance expressiveness, detection of inconsistency and complexity, ease of use, and compatibility with existing standards. We are applying knowledge based on ontologies that allow agents to understand the world around them and collaborate to achieve the best possible solutions to the challenges that arise.

This chapter presents a brief ontology context description, followed by some ontology specifications for risk assessment. We also present the representation of the proposed ontology for HRI and explain the way reasoning is applied to the ontology.

## A.1 ONTOLOGY DESCRIPTION

Gruber (1995) report that the goal of an ontology is that it formally describes something in the world, that machines can understand it, and that the knowledge it describes has the consensus of experts. It is also important that an ontology is understandable, extensible, and reusable in multiple contexts. Therefore, the ontologies must be formalized in a language that computers can interpret and allow the integration with other pre-existing ontologies.

Several methodologies help in the ontology production process, such as *METHONTOLOGY* Fernández-López, Gómez-Pérez e Juristo (1997), *On-to-knowledge methodology* (OTKM) Sure, Staab e Studer (2004) and *Ontology Development 101* Noy, McGuinness et al. (2001). Thus, initially, the objective and scope of the ontology are determined, then the reuse of other ontologies is considered, important terms are registered, the classes and the hierarchy between them are defined, the properties are structured, and, finally, the instances are created.

The formalization of an ontology is given by knowledge description languages such as Knowledge Interchange Format (KIF), Resource Description Framework (RDF), Extensible Markup Language (XML), and Ontology Web Language (OWL). Visual development tools can also be used for the production of ontologies, such as OntoEditor (de Souza Baptista, Vasconcelos e Arruda (2004)), TODE (Islam, Siddiqui e Shaikh (2010)) and Protégé (Musen (2015)). Another approach to building ontologies is based on Application Programming Interfaces (APIs) that are integrated into high-level programming languages. APIs provide objects and methods that manipulate classes, individuals, properties, constraints, etc. An example is the Apache Jena framework (Jena (2020)) for the Java language and the owlready2 package (Lamy (2017)) for the Python programming language.

Ontologies also allow the inclusion of reasoning rules to make deductions about specified knowledge. Reasoners like Hermit (Glimm et al. (2014)) and Pellet (Sirin et al. (2007)) use these rules to deduce new information and aggregate it into the knowledge base. Rules are specified using rule definition languages such as Semantic Web Rule Language (SWRL). SWRL rules are in the form of antecedent (body) and consequent (head) implications. A rule can be interpreted as follows: whenever the conditions specified in the antecedent are met, the conditions specified in the consequent must apply.

## A.2 ONTOLOGIES FOR RISK ASSESSMENT

The concept of operational risk management or ORM (*Operational Risk Management*) was described by Koller (2005) as the systematic process for identifying, analyzing, generating an answer, reporting, and monitoring operational risks. The authors define risks as events that negatively impact finance and business due to failures of people, processes, systems, or external events. Lykourentzou et al. (2011) uses this concept to propose an ORM ontology, focusing on registering domain knowledge and allowing risk management approaches to be adopted and shared collaboratively, enabling an organization to gather the knowledge related to operational risks. This consolidation allows inferences to be made with a more understandable knowledge base. Osorio, Manotas e García (2018) also applies the idea of operational risks. It seeks to improve risk identification and information sharing by proposing an ontology to identify operational risks. This ontology is aimed at the context of supply chain management for Third-Party Logistics (3PL) operations companies.

The research by (AZIZ; AHMED; KHAN, 2019) proposes an ontology-based hazard identifier, which is used to standardize information, facilitate computer processing, improve knowledge sharing between humans and robots, and establish information that other industries' processes can reuse. The reasoning uses multi-entity Bayesian networks, which produce quantitative results when estimating risks. For this, the research identifies the likely hazards, considers the mutual influence between the different hazards, and identifies the operational and environmental conditions. The authors contribute to the dynamic identification and standardizing of the information through an ontology that can be applied later for knowledge sharing across different industries.

The work by Borgo et al. Borgo et al. (2019) presents an ontology for manufacturing domains. Gopinath, Johansen e Ölvander (2018) presents a case study with the implementation of a risk assessment for human-robot collaboration. It shows the result of the application of risk assessment in the construction of a collaborative workspace with hand-guided industrial robots. The work identifies hazards and considers the experts' and workers' opinions in the process. The researchers insert themselves into the environment, participating in the construction process guided by the operators and linking this to the documentation of industrial equipment and scientific literature.

## A.3 REPRESENTATION

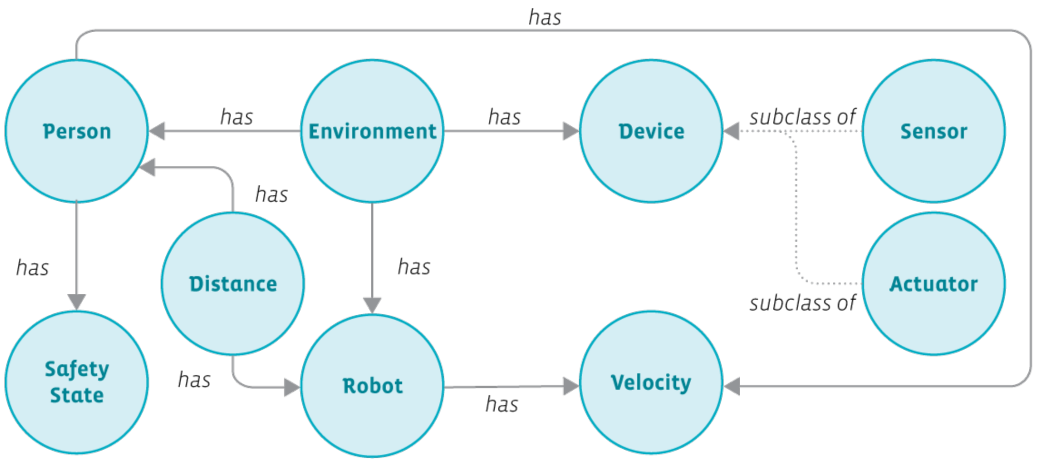
Ideally, it is necessary to identify the actions that people take and determine the safety state that this action causes to react when there is a need to reverse a potentially harmful situation. We seek to answer the following question: What is the person's security status?

Next, we demonstrate how representation structuring and reasoning were supported to avoid human-robot collision accidents. As mentioned earlier, we use the OWL ontology for this. For this purpose, the specification of semantic knowledge is initially presented, followed by a description of its implementation.

First, it is necessary to analyze the essential factors that make up the environment. In this case, the variables needed to make a decision. Once the semantic knowledge must be implemented in a machine, it must reason over the environment's state. Fig. 28 illustrates the environment's critical factors and how they relate to our application context. The circles represent the classes, and the arrows are the relationships linking them. The representation considers only the elements part of HRI in the industry for collision issues. This description is implemented as an Ontology. It was modeled using Protégé 5.5.0 Musen (2015). The result of the implementation is represented visually in Fig. 29 using a Visual Notation for OWL Ontologies (VOWL).

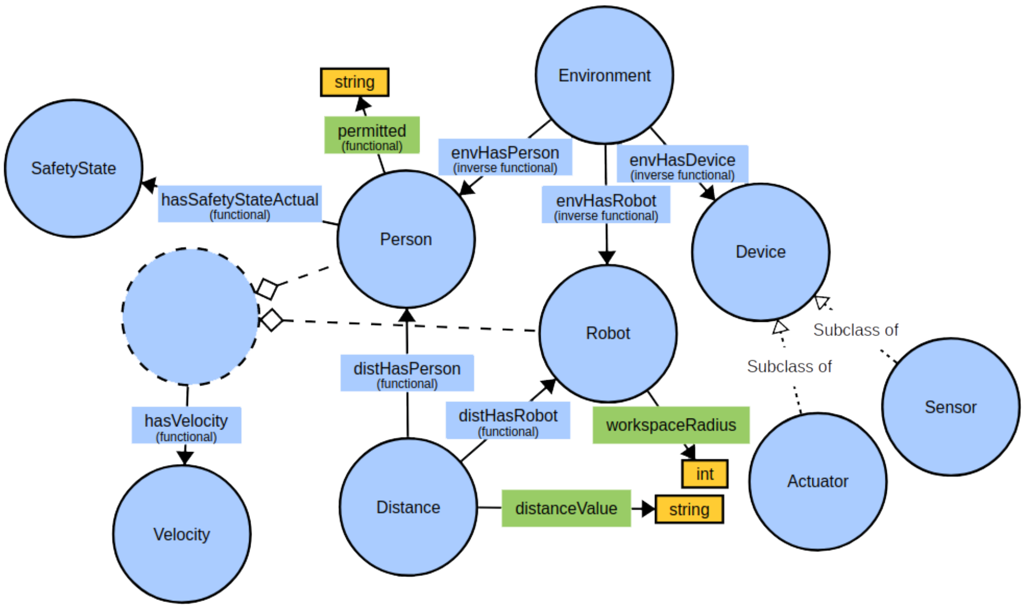
The environment includes people, sensors, and devices located somewhere in space. The Device class represents the devices inserted in the environment, with two subclasses,

Figure 28 – Diagram with the classes and relationships for the components in an industry environment for HRI.



Source: The author.

Figure 29 – Ontology for industry risk in VOWL notation.



Source: The author.

sensor, and actuator. The devices have a type and a model. The class Sensor includes devices capable of capturing, analyzing, and discovering a person's safety status. Cameras, temperature sensors, and audible sensors are examples of sensors. The class Actuator includes the devices that can cause some intervention in the environment, like audible and light alerts. As the name indicates, the Robot class represents the robots that perform tasks in the factory, and their actions can change a person's safety status. Robots have a workspace range and, like sensors, have a type and a model.

The class Person represents people included in the environment and who have a safety status associated with them. People have a data property to determine if they have permission to remain close to the robot in the environment.

The class Distance expresses the qualitative distance between robots and people. The class Velocity represents the qualitative speed of people and robots. The class Safety State is the possible safety condition in which a person can be. Examples are safe or risk states.

The entities of some classes need to be defined by specialists before the system goes into operation. This represents the static definitions. Static definitions can be divided into two types: Registry or Behavior. Registry is for those who register components that interact with the environment. Behavior definitions represent classes where the values of entities are predetermined. The created instances are restricted to such predetermined values and ranges. For example, it can only be received as slow or fast when dynamically receiving the person's speed. The Table 13 shows how classes are divided into static definitions. The classes that instances can dynamically receive during system execution are Velocity and Distance. The entities of the SafetyState class are inferred by the reasoning of each person registered in the system.

Table 13 – Division of the classes by static definitions types.

| Static definitions | Class | Example of instances |
|---|---|---|
| Registry | Environment | Laboratory |
| | Robot | Robotic arm |
| | Person | Mary, John |
| | Device | Pressure sensor |
| Behaviour | Velocity | Slow or fast |
| | SafetyState | Safe or risk |
| | Distance | Close or far |

Source: The author.

## A.4 REASONING

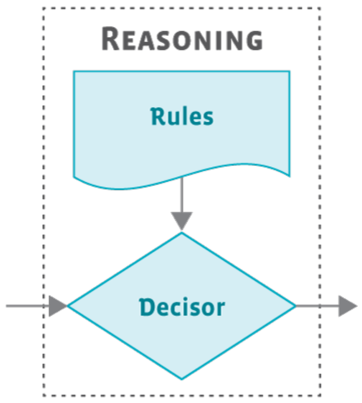
The reasoning is responsible for taking conclusions from the current knowledge represented in memory. It considers the information described in the representation step (Section A.3) to generate new knowledge from a set of beliefs or rules. Our proposal can be used to decide if someone is safe or about to suffer an accident (risk). The solution chooses actions based on what was reasoned. The use of reasoning expands the knowledge and is also central to extracting the safety status of any person in the scenario. It takes the current description of the environment as input. It returns the safety states.

We need to define rules that consider the variables influencing risk composition to make this decision. These rules apply to the current environment representation to infer risk. Thus, the reasoning considers the knowledge presented in Section A.3 and its role in defining a person's safety state that is used to alert the right environment agents and avoid accidents.

The knowledge representation used is based on rules in OWL notation. There are several OWL reasoning engines like HermiT and Pellet. They are applied for classification, production of new knowledge, and consistency check. Liebig (2013) analyzes different OWL reasoners, whereas Dentler et al. (2011) compares their applications for large ontologies.

Under reasoning, we have two modules: the reasoner and the rules, as illustrated in Fig. 30. The reasoner is the process that makes a risk state decision based on the gathered semantic knowledge. To perform this task, we use the semantic reasoner HermiT (Glimm et al. (2014) and Shearer, Motik e Horrocks (2008)), which is among the most efficient ones commonly adopted to execute ontology reasoning. The rules establish the decisions. They need to be formulated by the specialists in advance.

Figure 30 – Reasoning module steps.



Source: The author.

Decision-making relies on the definition of an ontology, and this is often accompanied by the definition of rules using, for example, the SWRL Horrocks et al. (2004). SWRL defines an abstract language that extends OWL semantics McGuinness, Harmelen et al. (2004) by including rules. There are no restrictions on how reasoning should be performed. As a result, various rule engines can be applied to reasoning from SWRL rules. Next, we present an example of an SWRL rule:

```
Rule: Person(?p1), Person(p2),
isParentOf(?p1,?p2) -> isChildOf(?p2,?p1)
```

This example reports that if a person *p1* is parent of *p2*, then person *p2* is child of *p1*. Note that the SWRL notation is very intuitive. A rule engine consumes the rules defined by the SWRL notation to deduce new information and aggregate it with existing knowledge.

A safety specialist who wants to use this system in the workplace must define safety rules relevant to its specific environment. These can be drawn from the local safety-oriented practices.

# APPENDIX B – PREVIOUS ARCHITECTURE VERSION APPLIED TO HUMAN-ROBOT INTERACTION

Several studies report an increase in Human-Robot Interaction (HRI) in industrial environments (Microsoft Dynamics 365 (2019)). Robots may operate autonomously to replace humans in hazardous tasks (Rocha et al. (2021)) or in tandem with workers in a manufacturing plant sharing confined spaces and tools (Vysocky e Novak (2016) and Villani et al. (2018)). We observe that collaborative robots in the industry deserve special attention since advances in this area directly impact productivity. HRI may result in unplanned and unwanted risks, leading to work accidents despite the many benefits. The insertion of collaborative robots in industrial environments brings new safety challenges and open issues, as presented by Tsarouchi, Makris e Chryssolouris (2016). There is a need to mitigate undesirable accidents between humans and robots while maintaining efficiency and effectiveness in the co-execution of tasks (Chandrasekaran e Conrad (2015)). It is also necessary to analyze the variables involved in this context to propose a more precise system.

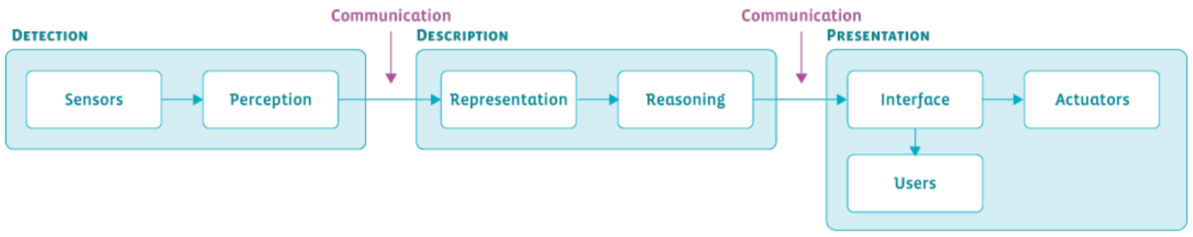
Therefore, the context of HRI was the first case study that motivated the development of the present thesis. By leveraging the context of HRI, an initial version of the reference architecture was proposed, which has been presented in this chapter. It uses ontology to represent knowledge. This architecture was used to develop the Human-RObot SAfety (HOSA) system. It is an end-to-end safety system that governs HRI. It seeks to deal with the resources available in the environment to offer safety. The system includes sensors (like cameras, smoke, and presence sensors), a network, reasoning machines, and the actuators (like robots, emergency lights, and buzzers). Next, we share the acquired experience by implementing the necessary modules as part of the design, highlighting and justifying the project's decisions. We believe that other developers stand to benefit from such insights.

## B.1 ARCHITECTURE

Building a system that prevents human-robot collision requires the design and integration of several essential modules into a complete solution. This section describes the modules that make up the proposed architecture and shows how they are combined. We also introduce the adopted underlying technologies and justify their selection to meet the previous requirements. Fig. 31 offers an overview of the system's architecture.

Our architecture considers three primary modules: Detection, Description, and Presentation. The Detection module builds and feeds detected information to the Description module, which interprets it and passes it to the Presentation module. The Communication infrastructure (highlighted in red in Fig. 31) is responsible for the information distribution

Figure 31 – Architecture overview.



Source: The author.

efficiently in a reliable and timely manner.

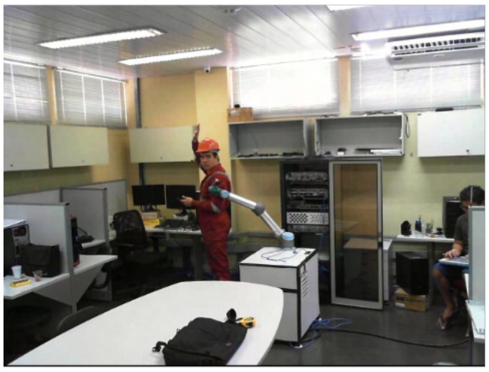
The Detection module is composed of the embedded modules sensors and perception. The Sensor module is responsible for converting real stimuli in the environment to electric signals that have to be processed by other parts of the system. They are specialized, relatively low-cost hardware devices that receive specific information. Examples include presence sensors, smoke sensors, and cameras. The Perception module processes the raw data obtained by sensors in the environment, transforming it into something more understandable to the system. It extracts relevant information from the sensed data.

The perception module is tasked with identifying the distance between humans and robots in real-time. This is needed to monitor and determine a risk of collision. In previous work Silva et al. (2020), we tried to make collision detection using only a single 2D camera and deep learning models for classification. However, the missing depth information only provided overlap detection between robots and humans. This work integrates the proposal of Rodrigues et al. (2022) into the HOSA system. It considers three cameras used to obtain information next to depth to make better collision inference. The surveillance cameras are strategically positioned at different angles in the environment. Fig. 32 presents examples of images from the cameras. An application receives the images from the three cameras. Processing is made using computational intelligence for collision and Personal Protective Equipment (PPE) detection. It is crucial to associate PPE detection with worker permissions and access privileges and use it to infer risk. One assumes that a person wearing PPE is at a lesser risk level when interacting with a robot than one who does not.

To produce a response to stimuli obtained from the environment, it is necessary that the system understands what was transmitted and that the computer must represent the knowledge. The Description step formalizes the information so that the computer semantically understands what happens around it. This knowledge can be used later to reach conclusions and generate new information based on reasoning. The Description step consists of two modules: Representation and Reasoning. We apply the representation and reasoning presented in Appendix A. The Representation module is dedicated to semantically instantiating the data received from the Detection phase to be generalized,

Figure 32 – Example of images obtained by the three cameras positioned from three different perspectives.

(a) Camera A



(b) Camera B

(c) Camera C

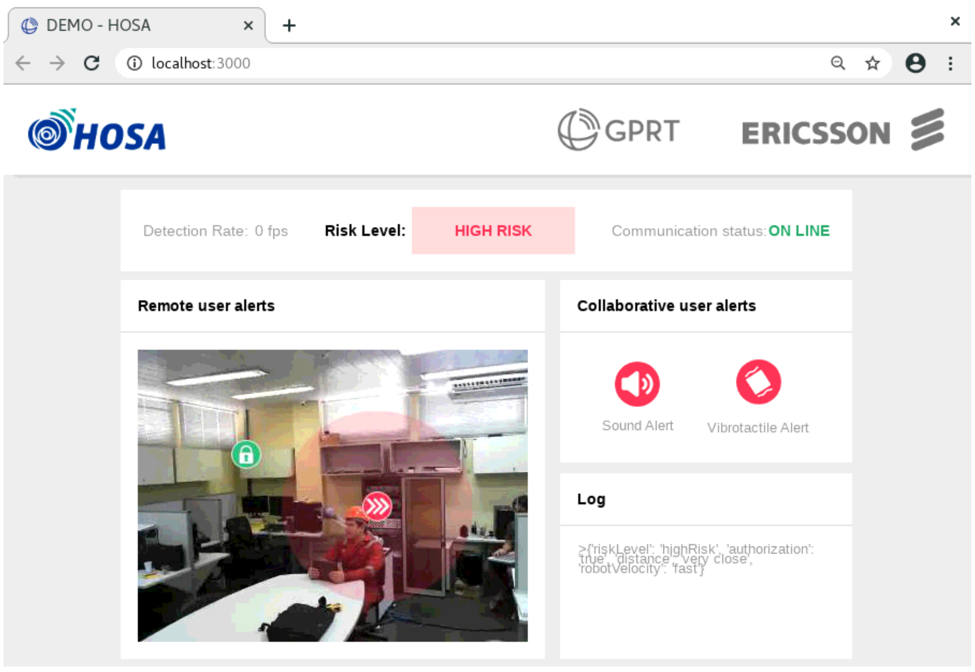
Source: The author.

interpreted, and prepared for understanding by machines. In sequence, we have the Reasoning module that must interpret the data from a scene. It should infer whether people are at risk or not.

Finally, considering the three main modules, we have the Presentation module. It reports the eminence of a risk to a worker in risk or someone following the risks situations in the environment from a distance, like using a computer web interface. It sends alerts to the actuators module and the users based on an advanced web interface. The architecture supports the triggering of actions to prevent accidents.

Fig. 33 illustrates the remote interface that can be followed by the remote user on a web page. Once the user is not where the accidents are displayed, it needs to have as much information as possible concerning the environmental situation. So we can see on the top of the page that there is information about the detection rate, inferred risk level, and active communication system. In remote user alerts, there is a visualization of the camera available in the environment with an augmented reality component that indicates if the person is in the area where the robot is working, the robot's speed and user permission icons can also be seen in the camera image. The alerts spaces have information on the state of audible and vibrotactile alerts. There is also a log for obtaining raw information transmitted by the system's components, which helps in debugging the platform.

Figure 33 – Web interface for remote user evaluation.



Detection Rate: 0 fps    Risk Level: HIGH RISK    Communication status: ON LINE

```
>{'riskLevel': 'highRisk', 'authorization':
'true', 'distance': 'very close',
'robotVelocity': 'fast'}
```

Source: The author.

Although there are several types of risk that people are exposed to when interacting with robots, like hooking, crushing, or tool cutting, this work is limited to the scope of collision risk to developing an initial yet promising end-to-end solution. Later extensions of the system design may address other types of risk.

## B.2 USE CASE

This use case aims to present the joint operation of all modules in the proposed safety system for HRI. We initially describe the scenario (Section B.2.1), then we show how the system configuration is achieved before starting its operation (Section B.2.2). Finally, we demonstrate its operation from the moment the sensors capture information from the monitored environment until some command is sent to the actuator in the event of a safety risk (Section B.2.3). Section B.2.3 also demonstrates the proposed user interface that a remote operator can use to follow the safety situation. This exercise aims to demonstrate the architecture's applicability in the design and building of a real system for monitoring human-robot collisions.

### B.2.1 Scenario

The scenario is set in the Industry 4.0 context where HRI takes place as part of a maintenance task of radio base stations. There is the presence of IoT devices, robots, and people interacting with the environment and its components. More specifically, the adopted scenario has three cameras, a robotic arm, an emergency light for visible alerts, and a speaker for audible warnings. A human and a robot interact as part of a maintenance task. We consider a network rack where the human and robot collaborate in activities of cable connection or disconnection and cable exchange to maintain the network operation. Note that this maintenance activity in the network rack can be generalized to maintenance in other machines in different scenarios or assembly tasks using a conveyor.

The robot is mainly used to insert and remove cables from the panels in the communication rack and it also collaboratively receives and delivers cables from and to people. The worker has to collaborate with the robot to conduct tests. It performs more sophisticated handling of the cables, which requires precise insertion or corrects some imprecise robot insertion to avoid a faulty link connection. People may become at risk as they walk and stay near the robot when the arm moves or hands over to and recovering cables from the robot or in the patch panel. Collisions may result in minor, medium, or even severe injuries. Fig. 34 illustrates the environment and Table 14 shows the specifications of the devices present in the scenario. The choice of the devices included those usually found in workplaces, without the necessity to include specific and more expensive ones. This is important for our solution since we propose a solution applicable to different contexts without small changes.

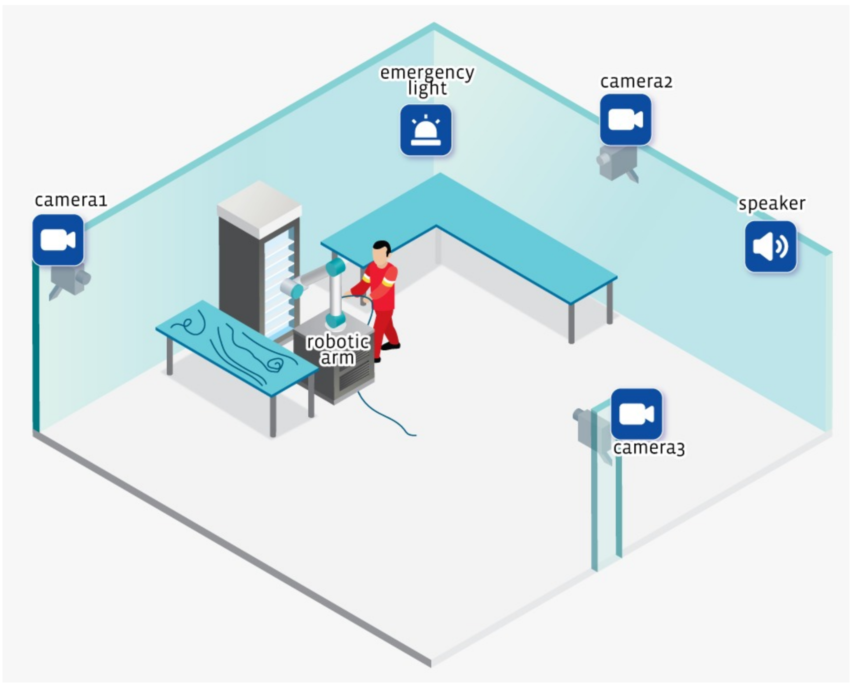
Table 14 – Description of the equipment that composes the scenario.

| Equipment | Qty | Specification |
|---|---|---|
| Camera | 3 | Yiequin W5820 |
| Router | 1 | TPLINK WR1043NDv2 |
| Robotic Arm | 1 | UR5-Universal Robots |
| Emergency light | 1 | Software simulation |
| Speaker | 1 | Software simulation |

Source: The author.

The cameras were used only to capture 2D images. These are ordinary surveillance cameras. They have no support for depth information, a feature that could help obtain better precision in identifying the distance between people and objects. Therefore, the

Figure 34 – Visual representation of the scenario.



Source: The author.

cameras need to be used together to manage, offering more accurate information about the scenario. Using only one camera cannot capture the entire field of action of the scenario and is prone to losing details. Furthermore, with more cameras, we can have redundancy of information, which reduces the impact of possible capture failures due to any possible camera malfunction or occlusions.

We use the Real-Time Streaming Protocol (RTSP) to control the data transfer from cameras, as the cameras support this protocol natively. Being a consolidated technology in video transmission, RTSP offers several functionalities, such as supporting multiple end-points and secure communication through authentication. We captured the images by running several video applications such as VLC Player (VLC… (2022)), and Mplayer (MPlayer… (2022)). Also, RTSP achieves higher performance over WiFi networks and sustains a quick reproduction when compared to other video protocols, such as MPEG DASH (Syaifudin et al. (2018) and Aloman et al. (2015)).

Initially, we used the OpenCV library to help capture images directly with the camera. However, this tool did not operate appropriately and lacked communication support with the service available on the camera. Thus, it was necessary to develop an intermediate streaming server, using the tool ffmpeg (FFmpeg (2022)), to provide communication between the agent (model processing machine) and the cameras.

In the development of our scenario, we used a Wireless Router. It is important to note that in the research scope, where damage or fire may occasionally occur, the system must be planned to remain operational and withstand these problems. The presence of cables usually hampers the robot's movement. Moreover, wired network connections can be disconnected, cut, or degraded by the environment, machines, or robots in motion and the presence of chemical substances. As a result, we avoid using cables in the network infrastructure. The wireless router made it possible to isolate our safety network from the operational network, to fulfill the function of forwarding information between the elements of a network.

The UR5 robotic arm was developed for collaboration with humans, prioritizing safety issues. It has predefined fixed activities set automatically, such as placing and removing cables from a panel, and receives and delivers cables to a human operator. The possible commands to send to the robot are: varying its speed, determining whether it should stop, whether it should keep the speed low or if it can keep the operation at maximum speed.

### B.2.2 System setup

Before using the system, we must define the necessary variables for its initial operation. In this case, the safety specialist's role is critical to define the parameters and semantic rules.

The values of the static registry and behavior definitions are necessary to instantiate the ontology, as presented in Section A.3. Thus, considering the previously presented scenario, we define these parameters according to the Table 15. We consider that three people can circulate in this space. It also has three cameras and a speaker for emitting emergency alarms. The system's behavior must consider three possible speeds (stopped, slow, and fast), which is based on SSM's idea with three operational speed states reported by Marvel (2013). Note that they were initially called pause, slow and clear, respectively. We consider the five safety status levels (safe, low Risk, medium Risk, high Risk, and accident) and three distance levels (far, close, and contact). The UR5 arm Application Programming Interface (API) provides the robot's current speed.

Fig. 35 illustrates the ontology instantiated with the definitions presented above, obtained from the Ontograf visualization of the Protégé software (Musen (2015)).

Next, it is necessary to define the rules that orchestrate the decisions made in our reasoning. The rules determine the person's safety state level from observing the dynamic data received and instantiated in the ontology. Thus, we defined the rules shown in Table 16. However, these rules are translated to SWRL notation. An example of rule 1 (if a person is far from the robot, then the person is safe) presented in Table 16 may be translated in SWRL notation as shown below:

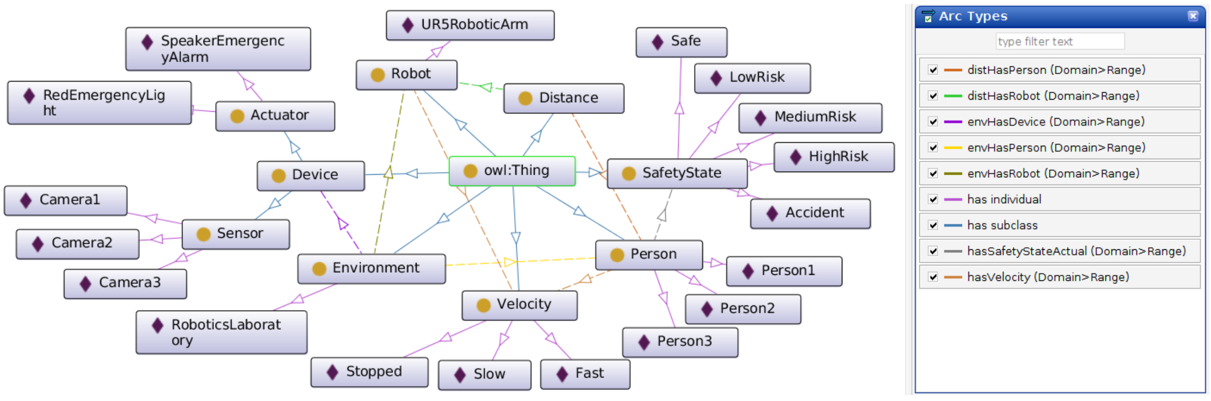
```
Rule 1 in SWRL notation:
Robot(?r), Person(?p), Distance(?d),
distHasPerson(?d,?p), distHasRobot(?d,?r),
distanceValue(?d, "far")
```

Table 15 – Static definitions of the components that make up the environment and system's behavior inserted as instances in the ontology.

| Static definitions | Class | Value |
|---|---|---|
| Registry | Environment | Robotics Laboratory |
| | Robot | UR5 Robotic arm |
| | Person | Person1, Person2 and Person3 |
| | Sensor | Camera1, Camera2, Camera3 |
| | Actuator | Speaker for emergency alarm |
| Behaviour | Velocity | Stopped, Slow and Fast |
| | SafetyState | Safe, LowRisk, MediumRisk, HighRisk and Accident |
| | Distance | Far, Close and Contact |

Source: The author.

Figure 35 – The ontology visualization with the static definitions components instantiated.



Source: The author.

```
->
hasSafetyStateActual(?p, safe)
```

Note that the set of decisions used in Table 16 does not depend on the human velocity. When a person is at high risk or accident state, the robot must be moved to a safe position by moving the arm upwards, which reduces the area of lateral contact. If the state of medium or low risk occurs, then the robot's speed is reduced to slow, and if the person is safe, the robot can continue its regular operation as long as this state remains valid.

### B.2.3 System execution

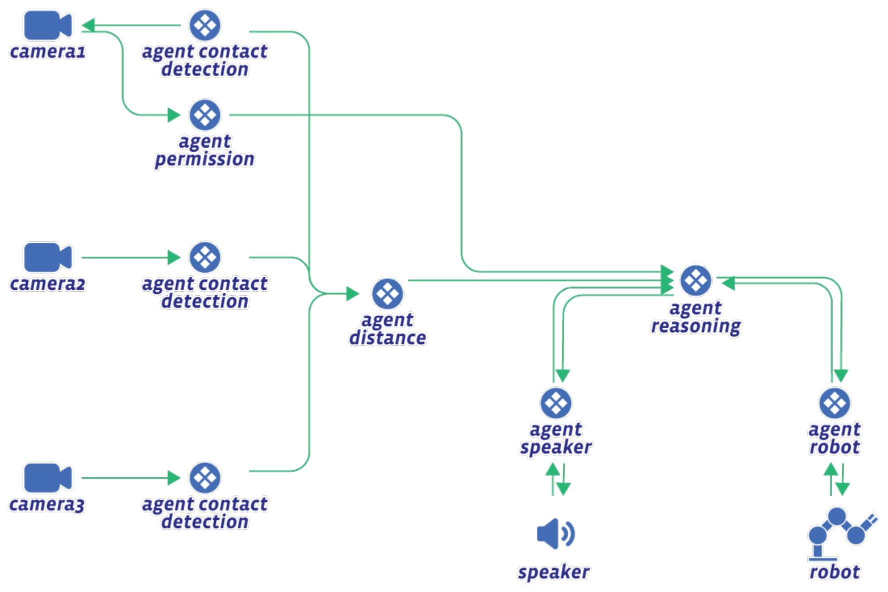
This section reports the system's execution going through each module according to an established execution workflow. The scenario, presented in Fig. 34, can be unraveled in an architectural visualization of the components that are part of the complete flow of perception until the reaction, see Fig. 36.

Table 16 – Rules defined for the reasoning. Represents an if-then operation, where the variables on the if side have an and operation. In case, If (human-robot distance) and (robot velocity) and (human velocity) and (human permission) Then (safety state).)

| Rule | If | | | | Then |
|---|---|---|---|---|---|
| | Human-Robot Distance | Robot Velocity | Human Velocity | Human Permission | Safety State |
| 1 | Far | - | - | - | Safe |
| 2 | Close | Stopped | - | True | LowRisk |
| 3 | Close | Stopped | - | False | MediumRisk |
| 4 | Close | Slow | - | True | LowRisk |
| 5 | Close | Slow | - | False | HighRisk |
| 6 | Close | Fast | - | - | HighRisk |
| 7 | Contact | Stopped | - | True | LowRisk |
| 8 | Contact | Stopped | - | False | HighRisk |
| 9 | Contact | Slow | - | True | MediumRisk |
| 10 | Contact | Slow | - | False | HighRisk |
| 11 | Contact | Fast | - | - | HighRisk |

Source: The author.

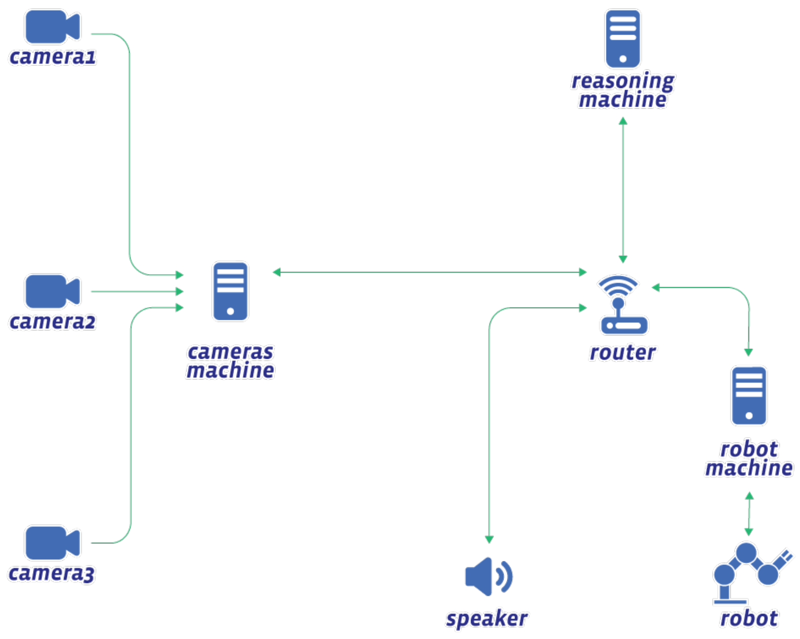
Figure 36 – Overview of the system architecture with a focus on processing agents for specific tasks.



Source: The author.

Fig. 37 illustrates the physical hardware devices in the scenario. The camera's machine (computer hosting the agent responsible for processing video input) includes the three contact detection agents (one per camera) and the permission detection agent. This same machine also hosts the distance agent that infers the distance from the contact detection agents' output. The reasoning machine runs the agent reasoning and holds the ontology, the rules, and the rules engine for decision making. The robot agent executes separately and processes the commands sent to control the robot. The wearable agent executes over the physical wearable device and processes the commands sent to it. Such commands may ask the user to move away or activate some device.

Figure 37 – Overview of the system architecture focusing on the devices and machines present in the scenario.

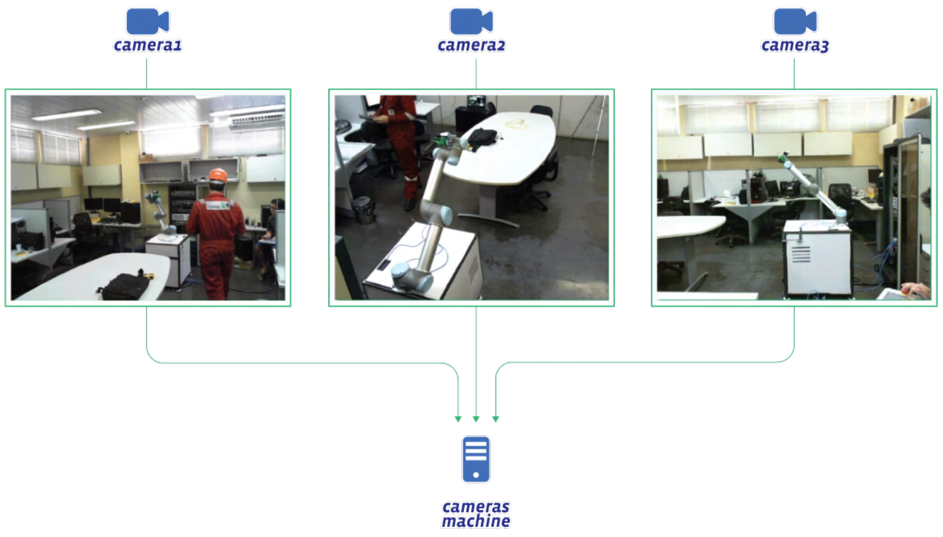


Source: The author.

The cameras are always capturing images and streaming video using the RTSP protocol. Fig. 38 illustrates the images that are transmitted from each camera perspective. Upon receiving a new image, the three collision detection processes associated with each camera perform the initial processing by removing the background from the image and leaving only the area of interest (the robots and humans). The resulting images can be seen in Fig. 39.

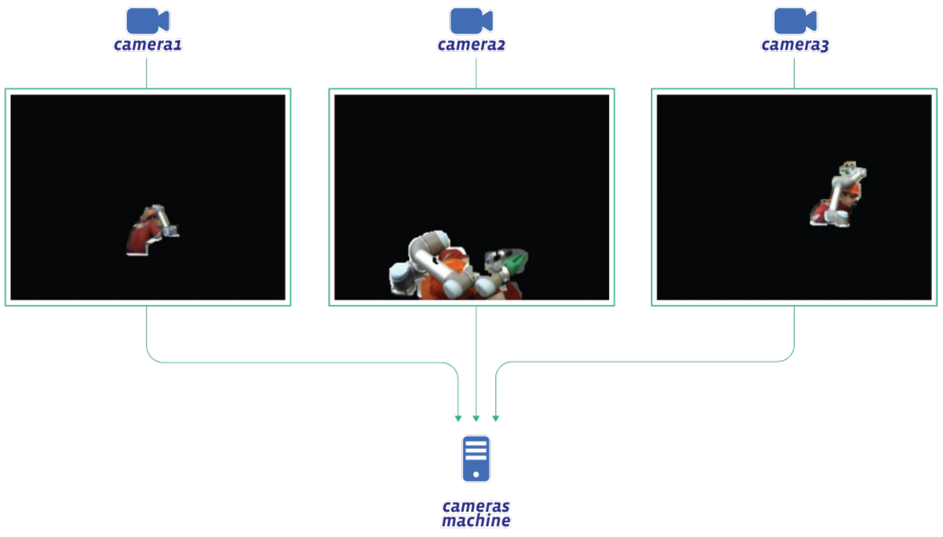
The algorithm now chooses if there is an overlap between the person and the robot by verifying if there is an intersection between the pixels of these components. If this is the case, then we consider that there is an overlap. This information is then used to determine the distance between humans and robots. When there is overlap in the images from all three camera perspectives, we consider that the person is in contact with the robot. If only two perspectives can observe overlap while another says that there is none, we have a case where the person is close to the robot. The person is considered distant from the

Figure 38 – Video stream to the camera machine with images that demonstrate the capture perspective of each camera.



Source: The author.

Figure 39 – Visualization of each camera image after removing the background.



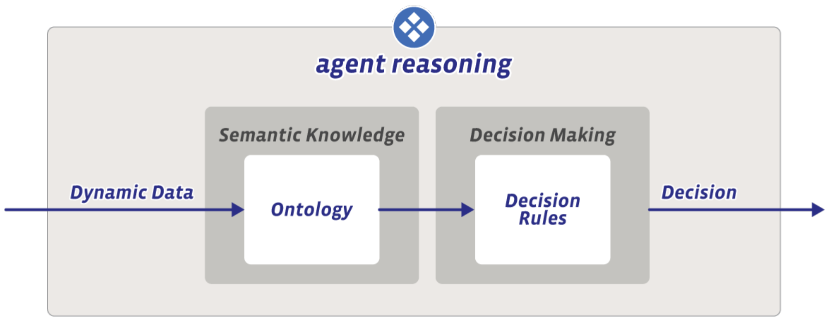
Source: The author.

robot in any other combination of responses. As a result, this distance response is passed on to the reasoning agents.

The agent permission considers that a person has permission when using the PPE. It uses the perception presented in Section B.1 to set if the person is using such equipment, and it also feeds this information to the reasoning agent. Note that this computation is performed only on one camera.

By combining PPE usage, a person's permission, and the human-robot distance, the reasoning agent makes its final decision about the level of risk. Fig. 40 illustrates an overview of the modules that make up the reasoning agent. This dynamic information that arrives is inserted in the ontology, which, after instantiating, is passed to the decision module to process and generate new knowledge. This processing follows the predefined rules and policies. Finally, the inferred safety state is communicated to the devices that provide feedback to users, the remote interface, and the robot for possible action.

Figure 40 – Overview of the agent reasoning internal modules.

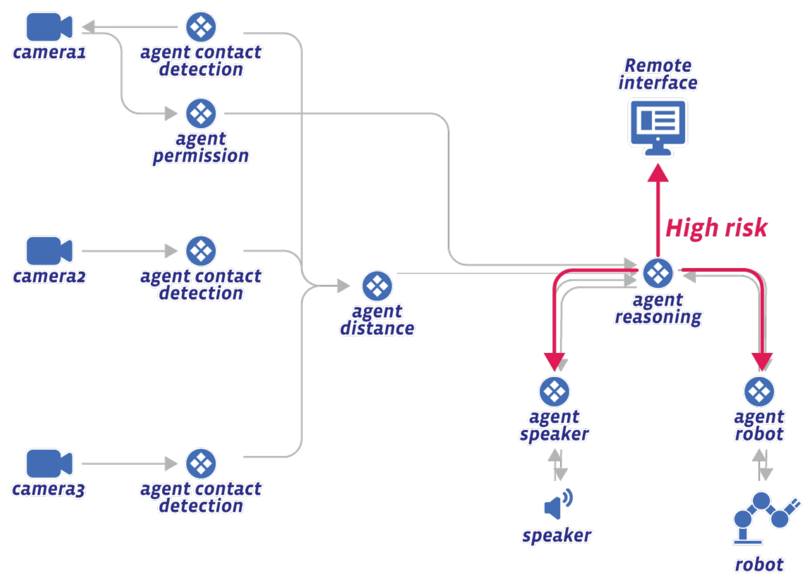


Source: The author.

Fig. 41 illustrates a message with the inferred safety status sent to the devices and the remote interface with a risk status decision. The devices react upon receiving this message. For example, when a person is considered at a high-risk state, the wearable (e.g., smartwatch) vibrates continuously and uninterruptedly, and the speaker makes a continuous alert noise. These events should draw a person's attention to the current risk and give them time to exit the premises or avoid it. Similarly, the robotic arm performs the routine of moving to a safe position, preventing it from being in a user-collision position.

The answer retrieved for the remote user on the web interface can be seen in Fig. 33. It should observe the decision that the user is at high risk. That is also made clear from the camera image in the remote user alerts area. The interface also shows that the person has permission to operate the robot (as he is using PPE) and that the audible and vibrotactile alerts are active. That allows the remote user to immediately react to this situation, intervening on the devices or calling upon the people closest to the person at risk to alert or help him.

Figure 41 – Transmission of the denied safety status to the receivers.



Source: The author.

## B.3 EXPERIMENT

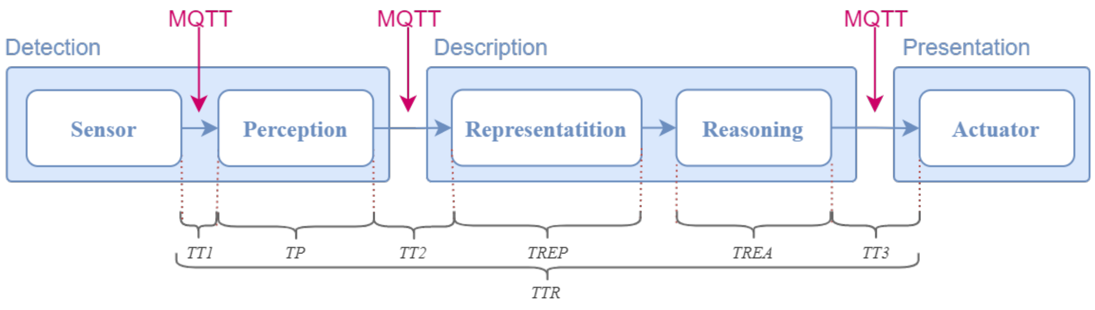
We performed some experiments that measure the reaction time of our system in the event of a safety risk. More specifically, we evaluate the time spent for the execution of each module in our system, considering the architecture presented in Fig. 31 (Section B.1) and the respective technologies presented. The following section describes the methodology adopted for conducting experiments, followed by a description of the results.

### B.3.1 Methodology

The experiment targets the measurement of the total reaction time of the proposed system. Fig. 42 offers insights into the system configuration and the periods that need to be measured. As a sensor, we simulated a camera using a process that takes a video signal and transmits this a frame at a time. It sends a total of 10000 frames, with 640x480 pixels resolution. The Perception module receives the raw images and processes each frame in order to determine: a) the human-robot distance and b) if the person is using PPE. This information is then fed into the Description module. This module uses the adopted Ontology to interpret it and passes its findings to the Presentation module. Reasoning determines risk presence according to the semantic definitions and rules in Tables 15 and 16, respectively. Next, the processed risk state definition is transmitted to the interface. The following defines the time consumed by each module in the multi-stage processing pipeline adopted in the present architecture:

- **TT1:** Transmission Time 1. Represents the time it takes to transmit between the

Figure 42 – System configuration and times to be measured for the experiment.



Source: The author.

sensor and the Perception module.

- **TP:** Perception module Time. It is the time the Perception module spends processing the data received from the sensors.
- **TT2:** Transmission Time 2. Reflects the time necessary to transmit between the Perception module and the Representation module.
- **TREP:** Representation module Time. This is the processing time required by the Representation module.
- **TREA:** Reasoning module Time. Measures how long the Reasoning module takes to process the data.
- **TT3:** Transmission Time 3. Tells the duration for passing the outcome of the Reasoning module to the Actuator module.
- **TTR:** Total Reaction Time. It determines how long our solution spends from the moment the sensors send the data to the moment the actuator receives a response. In other words, it is the sum of the previous time values.

The communication among these modules follows the publish/subscribe paradigm and uses the MQTT (Message Queuing Telemetry Transport) protocol with the Mosquitto server Light (2017). Note that we do not have network communication between the Representation and Reasoning modules, as these programs have been co-located on the same host to avoid transmission overhead in the present scenario.

This experiment considers a single machine where all the agents execute. This facilitates the measurements as all agents use the same clock. The specifications of the machine used are given in Table 17.

### B.3.2 Results

Table 18 and Fig. 43 lists the achieved timing results associated with each step part of the processing flow of our system. As the reader can see, the average overall TTR from

Table 17 – Machine specification.

| | |
|---|---|
| **OS** | Ubuntu 20.04.3 LTS |
| **Kernel** | 5.4.0-86-generic x86_64 |
| **Processor** | Intel Xeon e302xx |
| **GPU** | Geforce GTX 1060 |
| **RAM** | 8GB |

Source: The author.

perception to the system reaction is 1.052 seconds. In other words, if the system perceives a human-robot contact, the system reacts within 1.052 seconds. This is considered an adequate reaction time capable of mitigating most potential collisions. The reasoning step consumes most of this time, spending 88.44% of the total time. The overhead is due to this module having to carry out inconsistency verification and make inferences in the ontology for all new insertions. Furthermore, we observe that it cannot be parallelized in the GPU in an attempt to reduce its execution time. As part of future work, we plan to separate the verification processing and perform this less frequently and not for all the insertions, as is the case at present. In addition, inferences could be executed on-demand. Combining these strategies lowers the cost of reasoning and, consequently, TTR. Despite that, one may state that the current results are encouraging, especially how reasoners have evolved. Recall that when reasoners such as Pellet and Hermit became available for the first time, ontology reasoning could take hours. Currently, reasoning is performed in the order of few seconds only (Shearer, Motik e Horrocks (2008) and Glimm et al. (2014)).

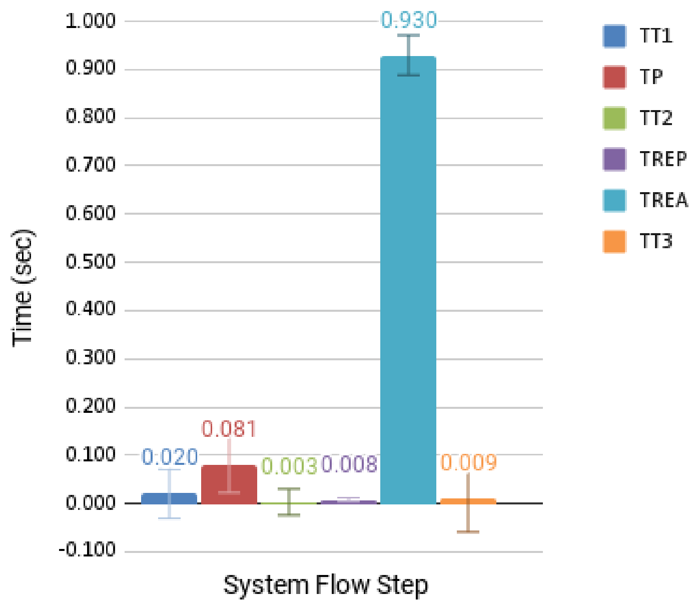
Table 18 – Average time (Time Avg) and standard deviation (Time Std) for 10000 frames transmission for each step in the system architecture. We also have the proportion (%) of TTR each step uses and the Processing Rate per Second (PRS - represents 1/Time Mean) for that each step processes or transmits the data.

| Test | TT1 | TP | TT2 | TREP | TREA | TT3 | TTR |
|---|---|---|---|---|---|---|---|
| Time Avg | 0.020 | 0.081 | 0.003 | 0.008 | 0.930 | 0.009 | 1.052 |
| Time Std | 0.051 | 0.059 | 0.027 | 0.004 | 0.041 | 0.068 | 0.116 |
| % TTR | 1.93% | 7.71% | 0.29% | 0.77% | 88.44% | 0.86% | 100% |
| PRS (1/Time) | 49.356 | 12.327 | 332.050 | 123.041 | 1.075 | 110.165 | 0.951 |

Source: The author.

Considering only the sum of the transmission times (TT1, TT2, and TT3) the solution suffers a total of 0.032 seconds of transmission time overhead, representing a mere 3.08% of the total time. Such low-level transmission time is expected once all the modules execute locally on the same computer leading to no packet loss. Observe the high standard deviation in this metric due to outliers resulting from the high transmission rates. The

Figure 43 – Average times execution for each step in the system architecture.



Source: The author.

largest share in transmission time was taken by TT1, responsible for the transmission of entire images. TT2 communicates information extracted from the image, including the human-robot distance information and whether or not a human user identified in an image is using PPE. This exchanged information is inherently much smaller than the previous one. Because of this, the TT1 is 6.73 times longer than the TT2. This shows us the importance of the detection module, mainly if we have more cameras, once more images data travels the network, and the system could resume this information. Observe also that both TT1 and TT2 overlap in their standard deviations. There is, therefore, a need to confirm that they follow different distributions. We run the Wilcoxon test because the samples do not follow a Gaussian distribution. The test resulted in a p-value of 0.028, which rejected the null hypothesis ($\alpha = 0.05$) and confirmed that they are statistically different.

The row in Table 18 lists the transmission or processing rate for each module. In other words, the number of frames it can deal with every second. The achieved processing time within the perception module, TP, reaches 12.327 FPS (Frames Per Second). In other words, if one uses cameras that generate images at a higher frame rate than this one, this module will be forced to drop some of these as it will be overwhelmed by the excess of frames. Note also that the insertion of additional cameras into the system must carefully consider the processing limitations of each module listed in Table 18. This is an important design consideration to ensure that a targeted low reaction time is achievable. For example, we could have 12 camera transmissions, each streaming at a 1 FPS rate or more cameras with lower frame rates. The effect on the overall reaction time should be the same.

Despite having a TREA being 0.930 seconds, the equivalent of 88.44% of the total time, the system can have more cameras and associated detection modules while maintaining a single Reasoning module. The reasoning module merely processes the data already instantiated in the ontology and does limit the module before it, namely the representation module. As a result, as the Representation module (TREP) has shown to be capable of processing at a 123 Hz rate (equivalent to 123 images per second), this module could, in fact, handle input from as many as 10 Detection modules. While the Reasoning module is processing, the representation module instantiates data in the ontology as new messages arrive. With ten detection modules and 12 cameras transmitting at 1FPS, the system may deploy up to 120 cameras. It could maintain approximately the same total response time, achieving 1.052 seconds, while considering cameras being directly connected to 10 PCs with the same configuration and without noise or packet loss in the network.

# APPENDIX C – FUZZY-BASED ATTENTION APPLIED TO OBJECT TRACKING AGENTS

This Chapter consider a multi-agent object tracking environment. We design and evaluate the effect of agent selection to optimize processing time while maintaining adequate accuracy of the raised diagnosis. More specifically, we consider a fuzzy-based attention, that is an increment of the proposed in Section 4.3.5. It orchestrates agents by controlling their utilization. At any given moment, it decides which agent can be activated and schedule when it can process or transmit its data. This chapter adopts and compares two distinct scheduling approaches. The first one relies only in the accuracy (as in the Section 4.3.5). In contrast, the second one is based on Fuzzy logic and considers both metrics: the accuracy and rate of Frames Processed per Second (FPS), as its main decision parameters.

We consider tracking algorithms as the main task of agent processing. Tracking algorithms are of particular interest to our experiment setup as they have a temporal dependence between the frames leveraged to obtain a better position prediction. But as the attention may prevent data processing at a given agent due to its low relevance feedback at a specific time, this could impact the tracking accuracy, leading to even cases of losing track of an object.

The agent selection approach improves FPS reduces processing time and costs while suffering minor accuracy loss. We also discovered that the higher the number of deployed agents, the greater our selection approach would reduce such cost.

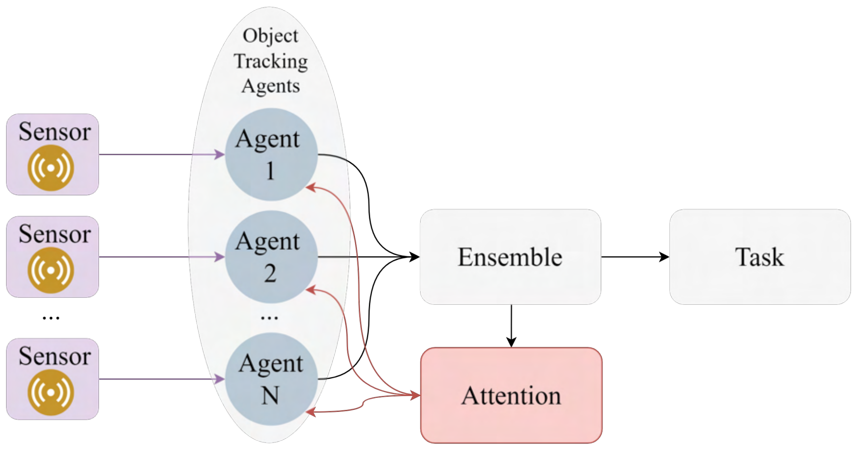
## C.1 METHOD

The architecture presented earlier in the thesis (Chapter 4) is simplified and adapted for the application in this specific use case. The architecture considered is illustrated in Fig. 44. It encompasses sensors, object tracking agents, an Ensemble, and Attention. Tracking agents can differ from each other in the algorithm used or the sensor connected. The Ensemble unifies the messages transmitted from the different agents and returns their results so that some tasks can be developed. The Attention module is responsible for selecting the most relevant agent each time.

The tracking agents are shown in Fig. 45. Each has a *Tracking* that represents the object tracking program with its specific algorithm. It also has a *Filter* that receives the Attention commands. The commands from the Attention determine whether or not the sensor data can be passed on to *Tracking*. There is also a module to make some measurements of *Tracking*; in the case of this work, the *Measurement* evaluates the Process Frames Per Second (FPS).

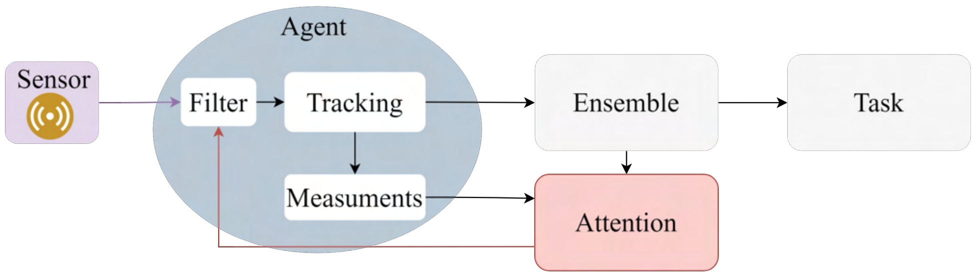
For the success of our evaluation, it is also necessary to consider an Ensemble that

Figure 44 – Architecture using object tracking agents.



Source: The author.

Figure 45 – Object tracking agent.



Source: The author.

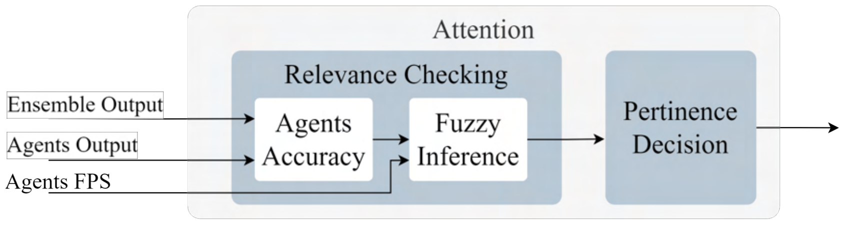
manages to improve accuracy when we increase the number of agents. Indeed we observed that some Ensembles could not achieve this. This is the case with Ensembles that prioritize different factors such as low standard deviation when using more agents. A case in point is Ensembles that produce the mean or median of their received numeric inputs.

For the present evaluation, we consider an Ensemble that receives a single suggested bounding box from each agent. The bounding box reflects the tracked object's position as inferred by a given agent. The Ensemble, then, selects from the different received bounding boxes the one which detected the object more accurately, considering the real position of the object in the image (ground trough). Note that this is not a real ensemble, it is only used for the evaluation purpose. The Ensemble always returns the box that is closest to the ideal (which is previously known), which is suitable for the purposes of evaluating the attention module that is done in this chapter. To assess which boxes at the ensemble's input are closer to the ground trough, we consider the one that returns the most significant Intersection over Union (IoU). Eq. C.1 refers to the adopted IoU metric, where $L$ is the position of the real object and $P$ is the input of the ensemble.

$$IoU = \frac{L \cap P}{L \cup P} \tag{C.1}$$

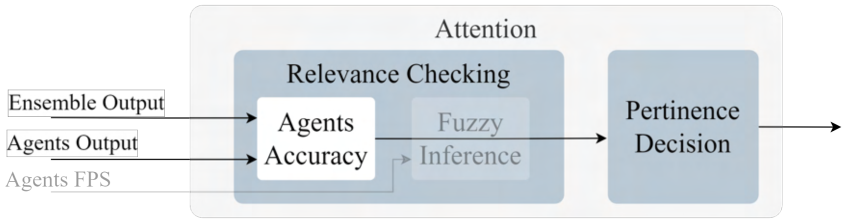
A fuzzy inference step is now added in the Relevance Checking, in the Attention module that was introduced in Section 4.3.5. This is illustrated in Fig. 46. In Section 4.3.5 the description only considers the accuracy of the agents related to the Ensemble output. The agents' accuracy is transported directly to the pertinence decision module, as is shown in Fig. 47. Therefore, to distinguish between both, we call the fuzzy inference Attention as A-Fuzzy, and the previous reference model in which only the accuracy is used as A-Ref.

Figure 46 – Object tracking Attention with fuzzy inference.



Source: The author.

Figure 47 – Object tracking Attention without fuzzy inference.

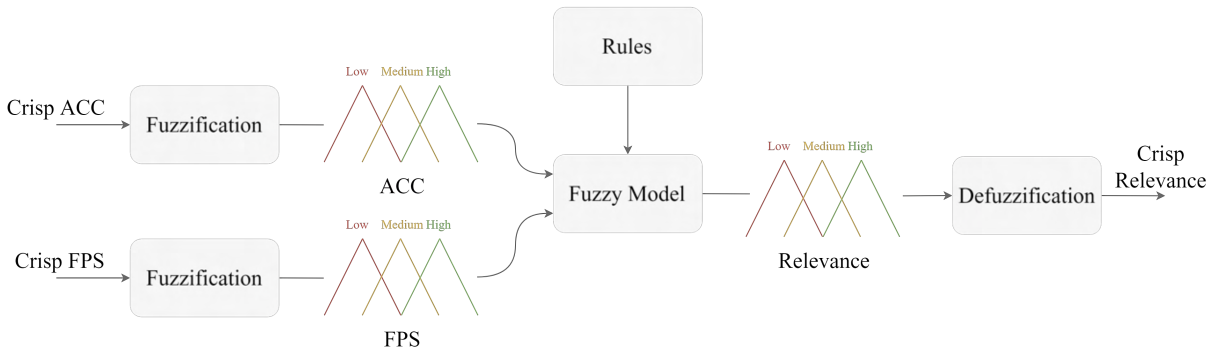


Source: The author.

The *Agents Accuracy* step is slightly different from the one presented in Section 4.3.5 as the system object tracking handles different types of data. The data type is a box represented as an array with four positions, which are the box coordinates in the Cartesian plane. *Agents Accuracy* returns a value between 0 and 1. It takes the agent's output and the ensemble's output as input. Thus, relevance to an agent is made by measuring its accuracy in relation to the ensemble's output. Therefore, a suitable measure of precision is the IoU (Eq. C.1).

The *Fuzzy Inference* step is included in the A-Fuzzy. It uses fuzzy logic as its underlying reasoning engine. The idea is to bring more compatibility to Attention, considering the different variables of the system that contribute to the decision-making about the relevance of an agent. More specifically, we consider two input variables, the agent's accuracy (ag-ACC) that comes from the previous *Agents Accuracy* step and the processing level measured as Frames Per Second processed (ag-FPS) in our application. These variables were chosen due to the context of the object tracking application, making it available to run online, so ag-ACC is needed to regulate the precision and ag-FPS to regulate the level of processing. Fuzzy brought them together for a unified result. Fig. 48 the fuzzy inference process.

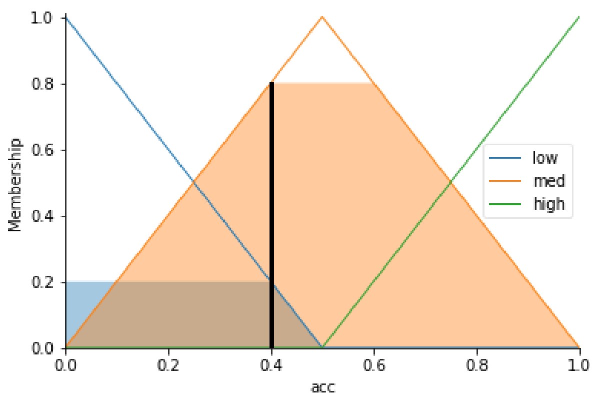
Figure 48 – Fuzzy inference.



Source: The author.

To summarize, the adopted fuzzy controller processes both ag-ACC and ag-FPS information at its input and delivers the relevance of an agent. ag-ACC fuzzification considers the fuzzy set illustrated in Fig. 49 whereas the Fuzzy set for ag-FPS is illustrated in Fig. 50. Both consider three Λ-shaped membership functions that are labeled as Low, Medium, or High.

Figure 49 – Fuzzy logic input with three membership functions for Accuracy.

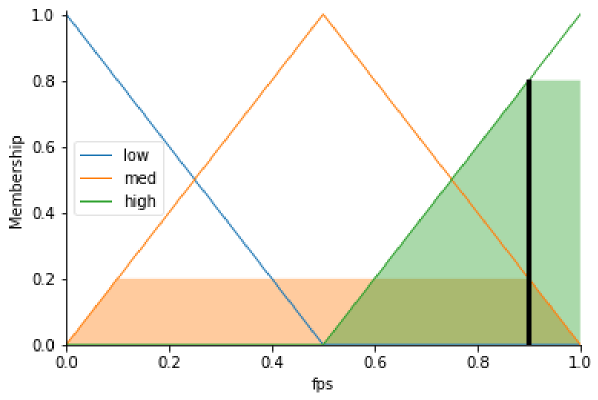


Source: The author.

The frame processing rate considered for the FPS parameter ranges from tens, to hundreds, or thousands depending on the tracking algorithm or the hardware used for processing. Prior to using this input parameter to the proposed fuzzy system, it is first normalized to the interval [0,1]. Note that this conversion is not needed for ag-ACC once it falls by nature into this range. To obtain ag-FPS, we consider the following min-max normalization:

$$fps'(x) = (fps(x) - min(fps))/(max(fps) - min(fps)) \tag{C.2}$$

Figure 50 – Fuzzy logic input with three membership functions for FPS.



Source: The author.

After receiving the input variables and performing the fuzzification, the fuzzy system performs rule evaluation based on the rule matrix detailed in Table 19 in order to produce the agent relevance level as the outcome. The matrix represents an $AND$ operation between the ag-FPS possibilities (Low, Med, and High) in the rows, and the ag-ACC possibilities (Low, Med, and High), listed in the columns. The result is in the cell that matches that row and column. For example, for $ag-FPS = High$ and $ag-ACC = Med$, the rule points to the result: $Relevance = High$. These rules were obtained based on a preliminary experimental test conducted with different ag-ACC and ag-FPS combinations. The combinations leading to more sensible results were retained. For defuzzification, we apply the center of gravity method. Fig. 51 illustrates the agent relevance fuzzy set. The membership function can have the labels Low, Medium, or High.

Table 19 – Fuzzy logic rule matrix.

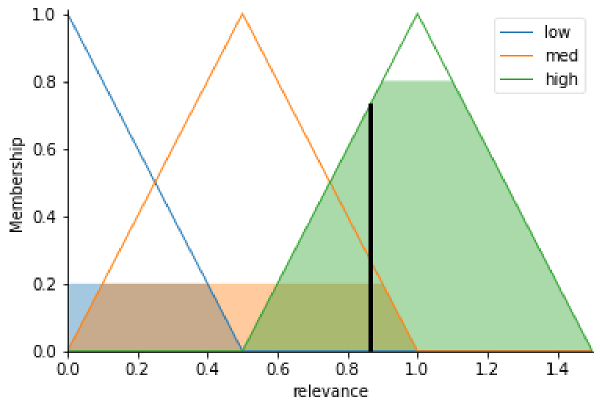
| **ag-FPS** | **ag-ACC** | | |
|---|---|---|---|
| | Low | Med | High |
| Low | Low | Low | Med |
| Med | Low | Med | High |
| High | Med | Med | High |

Source: The author.

For example, under the proposed fuzzy system when ag-ACC is 0.4 and ag-FPS is 0.9, the relevance output is 0.86. The Figures 49, 50 and 51 illustrate with a black line the position of the values for each respective variable. The membership functions are filled in the grade where the black line intersects with the membership function.

The *Pertinence Decision* is responsible for deciding whether or not to use the data produced by a specific agent in the future. Consequently, the agent processing could be paused when its relevancy is estimated as being low. This is why this module integrates

Figure 51 – Fuzzy logic output with three membership functions for Relevance.



Source: The author.

two critical variables. So as the tracking agents have to process frame stream, the first variable considers the number of sequential frames that the agent has to avoid processing, which we will simply call Pertinence Frame Skip (P-FS). This replaces the sleep time that is used in the Section 4.3.5, once for the case of this application is based on the frames flow, not in the time. On a first look, the P-FS can bring advantages when having sequential, repetitive frames that can be avoided. The other variable is the Pertinence Threshold (P-Th) that reflects the minimum relevance value that such an agent must achieve in order to remain active.

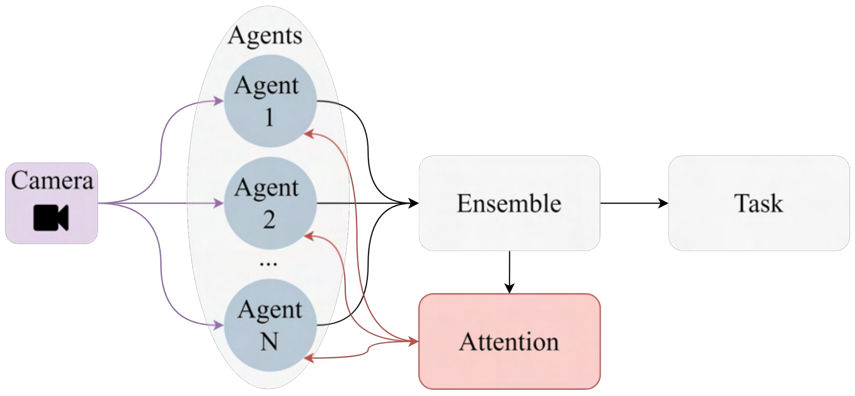
This module gathers the relevance from each agent and checks if it is above the user pre-established P-Th. If yes, that agent is allowed to keep processing video frames. Otherwise, it is instructed to skip a given number of frames as already defined by the P-FS parameter. Hence, the output of this module is referred to as an agent's pertinence decision.

## C.2 EXPERIMENTS AND RESULTS

This section describes the experiments performed to evaluate the method described above. Fig. 52 reflects the experimental scenario. In the scenario, all agents receive the same images for processing. The idea is to analyze the variability in the agent's output, regardless of the sensor input. The number of agents ranges from 1 to N, representing that, for experimental purposes, one can scale up or down the same agent type to identify when more or fewer instances of a given agent would improve the accuracy for some given input and whether including the Attention improves processing.

Agents return the bounding box to infer the position of an object in an image. Ensemble receives these boxes from all agents and returns a single box as an output. Attention receives the Ensemble and the agents' output and returns feedback to the agents, informing them if they should skip subsequent frames. At the beginning of the execution of each experiment, the initial position of the object to be tracked is pre-informed to the tracking agent.

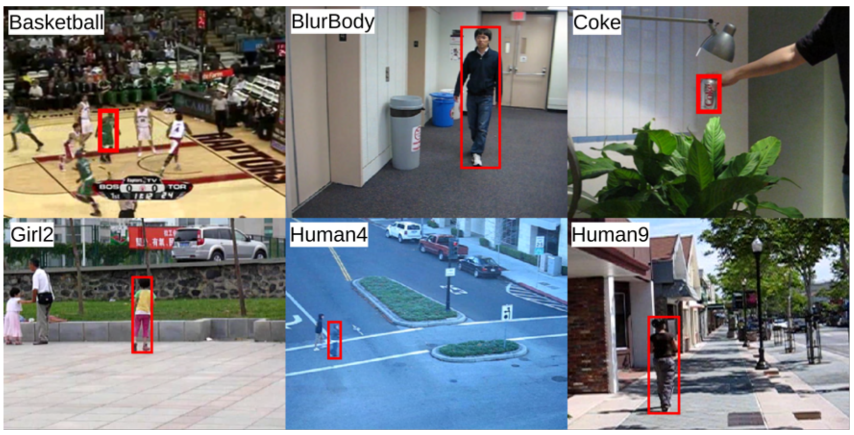
Figure 52 – Experimental scenario.



Source: The author.

The camera stream is emulated considering different datasets of (WU; LIM; YANG, 2013). Fig. 53 illustrates the frames of the datasets used and Table 20 has a description of each one. All frames in this dataset are streamed sequentially to each agent.

Figure 53 – Data sets used for experimental purposes.



Source: The author.

The metrics used to evaluate the results are accuracy (R-ACC) and Frames Per Second (R-FPS). They are different from the A-ACC and the A-FPS used in Attention. R-ACC first compares the Ensemble output to the ground truth for all inserted frames from the dataset. For this, R-ACC considers the IoU (Eq. C.1) between the two. It then evaluates the number of outputs above a predetermined detection threshold, to which a 0.3 overlap threshold ((ZHANG et al., 2017)) is applied.

The R-FPS represents the frame processing speed. To analyze the R-FPS, we consider the time for all agents to process an image given as input by the Camera. $ProcessingTime$ (as in the Fig. 52) is the time from when the first agent begins to process the image until the last agent produces the output. The FPS is calculated following the Eq. C.3:

Table 20 – Description of the data sets used in the experiments.

| Datasets | Tracked object | Number of frames | Frame dimension |
|---|---|---|---|
| Basketball | Human | 725 | 576x432 px |
| BlurBody | Human | 334 | 640x480 px |
| Coke | Soda can | 291 | 640x480 px |
| Girl2 | Human | 1500 | 640x480 px |
| Human4 | Human | 667 | 640x480 px |
| Human9 | Human | 305 | 320x240 px |

Source: The author.

$$R - FPS = \frac{1}{ProcessingTime} \tag{C.3}$$

Consequently, we obtain an R-ACC and an R-FPS values on a per frame basis. Next, we calculate the mean R-ACC and R-FPS for the entire data set.

The 21 table lists the hardware configuration of the machine running the experiments.

Table 21 – Specifications of the computer used for hosting the sets of experiments presented in this work.

| Device | Specification |
|---|---|
| Processor | Intel(R) Xeon(R) W-1250 CPU @ 3.30GHz |
| GPU | NVIDIA Quadro RTX 4000 8GB GDDR6 |
| RAM | 16GB |
| OS | Ubuntu 20.04.3 LTS |

Source: The author.

For comparison purposes, we evaluate the scheduling approach presented in section C.1, A-Ref and A-Fuzzy. Additionally, we also consider executions without Attention as baseline for our experiments, which we call A-None, as well as a simple Random scheduler, which is named as A-Rand.

We designed three sets of experiments. They seek to identify the impact of several important configuration parameters of our proposal. In the following sections, the tables that present the results are colored on a scale from red to green, following the color pattern established in Table 22. The more intense the red color, the lower the value it represents. Also, the more intense a green color, the higher the value it describes. Lighter, color-coded values close to white are closer to the average value.

Table 22 – Color scale pattern used to color code the results in the following tables.

| low | | | mean | | | high |
|---|---|---|---|---|---|---|
| 🟥 | 🟥 | 🟥 | ⬜ | 🟩 | 🟩 | 🟩 |

Source: The author.

### C.2.1 Pertinence Threshold (P-Th) and Frame Skip (P-FS)

Recall that the Attention uses a P-Th level below which we conclude that a given agent is not offering relevant information towards the Ensemble decision. An agent deemed to be of little relevance is then asked to skip the following frames. In other words, we assume that an agent that is not offering relevant information will continue doing so in the near future. So, in this experiment, one of the parameters is the P-Th.

The P-FS configuration parameter determines the number of frames this agent must skip. The first set of our experiments aims to understand the impact of P-Th and P-FS parameters on R-ACC and R-FPS metrics. For example, we looked to see if better tracking could be achieved by varying the P-FS. Likewise, we also seek insights to identify the optimal level(s) for P-Th. Ultimately, agents are expected to be subject to a well-calibrated pertinence decision module. In other words, we have to use the best P-FS and P-Th in a real online run.

Our preliminary tests showed that all datasets exhibited similar performance behavior when changing the P-Th and P-FS parameters. These initial tests were performed with all six data sets. Thus, to concisely demonstrate the results, we expose the results obtained for the BlurBody dataset. The experiments in Section C.2.2 uses the different remaining datasets.

Considering P-FS, we start its value at two and then gradually increase it to 10, using a step of 2. In other words, an agent can skip between 2 and 10 frames during experiments. The P-Th is initially set to 0.5 and increases to 0.9 using a step of 0.1. We set P-Th at the initial value of 0.5, which means that a given agent must achieve a minimum of 50% IoU accuracy. Intuitively, we estimate that below this minimum P-Th level, it is likely that many agents will always be selected, offering a small concrete contribution to the R-ACC while consuming considerable resources and possibly reducing R-FPS. We also observed that the results tend to stabilize after some maximum value for P-FS during the experiments. Consequently, we set a maximum level of 10 for P-FS. We ran each of these P-Th and P-FS combinations 30 times. As previously reported, each run returns R-ACC and R-FPS metrics. Then we took the mean and standard deviation of all 30 repetitions of the experiment.

We consider five agents. We empirically perceive this number of agents as sufficient to demonstrate the benefits of applying the proposed Attention, as well as to understand and highlight the role of an adequate selection of the P-Th and P-FS parameters. Furthermore, such cost incurred does not necessarily imply better R-ACC or R-FPS. In fact, the FPS will likely suffer more in the presence of slow agents. As part of the experimental setup, we used the MIL tracker, proposed by (BABENKO; YANG; BELONGIE, 2009). Our choice is motivated because its detection capacity varies with each execution or repetition of the experiments. Therefore, it offers more variability in results between multiple concurrent agents. The MIL tracker produces different responses for the same input. Experiments for different trackers are concentrated in Section C.2.3.

Table 23 summarizes the parameters used for configuring the experiment.

Table 23 – P-Th x P-FS experiment parameters.

| Parameter | Value |
|---|---|
| Executions | 30 |
| Qtd. agents | 5 |
| Data set | BlurBody |
| Tracker | MIL |
| Scheduler | A-None, A-Rand, A-Ref, A-Fuzzy |
| P-FS | 2, 4, 6, 8, 10 |
| P-Th | 0.5, 0.6, 0.7, 0.8, 0.9 |

Source: The author.

The Tables 24 and 25 list the average results of R-ACC and R-FPS obtained with the experiment. Observe from both tables that the A-Rand achieves the lowest levels of accuracy for all P-Th and P-FS configuration levels. This reinforces the need for a clever and well-crafted agent scheduling strategy other than doing this by random. For most cases, the A-Ref and A-Fuzzy Attentions cause an R-ACC level below that of A-None. This is expected behavior since both A-Ref and A-Fuzzy interrupt the processing flow by trackers, which can cause track loss. Despite that, there are scenarios where the mean R-ACC for the A-Ref and A-Fuzzy scheduling outperform the A-None. For example, this is the case when setting P-Th and P-FS to 0.9 and 2, respectively. Despite having overlapping standard deviations, these results show the potential of A-Ref and A-Fuzzy scheduling. We believe this occurs, for example, when there is some occlusion that disturbs the tracker. The A-Ref and A-Fuzzy could skip this disturbance, resulting in continuity in the object tracker.

Concerning the R-FPS metric (the result is in Table 25), the worst frame rate is achieved when no Attention is used (i.e., using strategy A-None) for all values of P-Th and P-FS. Unlike what we observed with the R-ACC metric, the impact of using a Attention on R-FPS is always positive. This has been the case even when using a Random Attention. This R-FPS gain is easily explained because the random, similarly to the others, avoid having all the agents active all the time. Instead, it selects a subset and hence allows the

Table 24 – R-ACC results for different P-Th and P-FS levels.

| Att. | P-FS | P-Th | | | | |
|---|---|---|---|---|---|---|
| | | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
| A-None | 2 | 0.83±0.09 | 0.86±0.08 | 0.8±0.09 | 0.84±0.09 | 0.82±0.09 |
| | 4 | 0.81±0.1 | 0.83±0.09 | 0.82±0.07 | 0.82±0.09 | 0.81±0.1 |
| | 6 | 0.82±0.1 | 0.84±0.09 | 0.83±0.09 | 0.85±0.1 | 0.85±0.07 |
| | 8 | 0.86±0.1 | 0.81±0.09 | 0.85±0.09 | 0.81±0.12 | 0.83±0.08 |
| | 10 | 0.82±0.08 | 0.81±0.09 | 0.83±0.09 | 0.81±0.09 | 0.84±0.09 |
| A-Rand | 2 | 0.52±0.08 | 0.49±0.06 | 0.51±0.06 | 0.48±0.1 | 0.53±0.11 |
| | 4 | 0.45±0.08 | 0.47±0.11 | 0.53±0.08 | 0.53±0.1 | 0.63±0.12 |
| | 6 | 0.46±0.11 | 0.48±0.1 | 0.51±0.08 | 0.58±0.09 | 0.62±0.11 |
| | 8 | 0.47±0.11 | 0.47±0.07 | 0.53±0.09 | 0.54±0.09 | 0.63±0.11 |
| | 10 | 0.45±0.1 | 0.51±0.08 | 0.51±0.09 | 0.56±0.09 | 0.62±0.1 |
| A-Ref | 2 | 0.88±0.07 | 0.82±0.13 | 0.83±0.1 | 0.89±0.04 | 0.89±0.04 |
| | 4 | 0.82±0.1 | 0.76±0.16 | 0.78±0.1 | 0.84±0.08 | 0.87±0.07 |
| | 6 | 0.8±0.08 | 0.75±0.13 | 0.77±0.14 | 0.83±0.06 | 0.84±0.05 |
| | 8 | 0.69±0.18 | 0.74±0.13 | 0.73±0.15 | 0.8±0.09 | 0.82±0.08 |
| | 10 | 0.7±0.15 | 0.71±0.1 | 0.71±0.14 | 0.78±0.09 | 0.79±0.09 |
| A-Fuzzy | 2 | 0.88±0.09 | 0.85±0.09 | 0.81±0.14 | 0.86±0.09 | 0.89±0.04 |
| | 4 | 0.86±0.08 | 0.81±0.12 | 0.76±0.18 | 0.83±0.07 | 0.82±0.12 |
| | 6 | 0.8±0.11 | 0.76±0.13 | 0.78±0.12 | 0.8±0.11 | 0.84±0.08 |
| | 8 | 0.75±0.12 | 0.75±0.13 | 0.7±0.15 | 0.76±0.11 | 0.79±0.09 |
| | 10 | 0.72±0.16 | 0.71±0.15 | 0.71±0.15 | 0.79±0.09 | 0.79±0.1 |

Source: The author.

processing of more frames, reflecting in a higher R-FPS. However, the A-Ref and A-Fuzzy reach the best results under most P-Th and P-FS setups. This is particularly evident with some high values of the parameters P-Th and P-FS. Skipping more frames (i.e., higher P-FS) and having stringer conditions for selecting agents (higher P-Th) offers better results up to a certain degree. To keep increasing P-FS ultimately is prone to reduce R-ACC. This happens since the tracking application has a temporal dependency between frames. When some frames are skipped, a lack of information leads to an input hole comparable to an occlusion. But, in some cases, it could be interesting to use higher values for P-Th and P-FS. This may be the case, for example, when prioritizing R-FPS over R-ACC. With A-Rand, R-ACC increases, whereas R-FPS decreases when augmenting P-Th. Here for a higher P-Th, such as 0.9, each agent has a probability of 10% to be chosen. As a result, there could be many instances where no agent is selected at all. When this is the case, the system reacts by activating all of its agents (i.e., no restrictions imposed by agent selection). We will then have more issues where all agents are on, which results in less

Table 25 – R- FPS results for different P-Th and P-FS levels.

| Att. | P-FS | P-Th | | | | |
|---|---|---|---|---|---|---|
| | | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
| A-None | 2 | 7.29±0.19 | 7.17±0.16 | 7.26±0.1 | 7.32±0.12 | 7.39±0.1 |
| | 4 | 7.37±0.11 | 7.32±0.11 | 7.32±0.11 | 7.33±0.11 | 7.39±0.12 |
| | 6 | 7.32±0.15 | 7.27±0.13 | 7.36±0.12 | 7.32±0.13 | 7.32±0.11 |
| | 8 | 7.27±0.12 | 7.28±0.07 | 7.3±0.12 | 7.3±0.1 | 7.33±0.1 |
| | 10 | 7.32±0.14 | 7.33±0.11 | 7.31±0.1 | 7.36±0.11 | 7.3±0.11 |
| A-Rand | 2 | 17.81±0.57 | 20.18±0.7 | 21.25±0.88 | 21.98±1.01 | 19.65±1.87 |
| | 4 | 22.52±1.15 | 22.65±0.76 | 21.1±0.79 | 19.87±0.69 | 16.49±0.7 |
| | 6 | 22.71±0.8 | 22.05±1.12 | 21.13±0.77 | 19.81±0.71 | 16.38±0.8 |
| | 8 | 22.78±0.82 | 22.41±0.75 | 21.47±1.08 | 19.59±0.79 | 16.49±0.91 |
| | 10 | 22.9±0.94 | 22.08±0.94 | 21.3±0.77 | 19.62±0.69 | 16.77±0.62 |
| A-Ref | 2 | 15.31±1.53 | 17.23±1.36 | 17.4±1.65 | 19.82±2.39 | 20.96±2.93 |
| | 4 | 21.08±2.86 | 21.97±1.58 | 24.17±1.8 | 24.9±2.09 | 24.43±1.58 |
| | 6 | 22.68±2.19 | 24.74±1.59 | 26.43±1.58 | 27.51±2.04 | 28.03±1.28 |
| | 8 | 25.58±1.84 | 27.38±1.43 | 28.07±1.78 | 28.41±1.23 | 29.52±1.32 |
| | 10 | 27.57±1.72 | 28.36±1.13 | 28.74±1.4 | 29.61±1.58 | 31.13±1.4 |
| A-Fuzzy | 2 | 12.14±0.95 | 16.36±1.28 | 18.1±1.69 | 20.12±2.24 | 20.28±3 |
| | 4 | 15.85±1.49 | 22.42±2.12 | 23.19±1.57 | 25.18±2.53 | 24.92±1.98 |
| | 6 | 17.11±1.31 | 24.2±1.11 | 26.21±1.66 | 26.97±1.97 | 28.13±1.72 |
| | 8 | 19.91±1.98 | 26.77±2.15 | 28.2±1.87 | 29.16±1.33 | 29.89±1.27 |
| | 10 | 21.27±2.31 | 28.48±1.45 | 28.88±2.06 | 29.52±1.74 | 31.1±1.12 |

Source: The author.

R-FPS and higher R-ACC for higher P-Th in the A-Rand.

### C.2.2 Scalability

As part of the experiment, we also evaluate our Attention's scalability. In other words, we want to understand its behavior and impact on Attention performance as we increase the number of agents. The scalability of all four Attentions, namely, A-None, A-Rand, A-Ref and A-Fuzzy, is examined. Recall that we use MIL tracking to ensure that although the different agents receive the same input information stream, they nonetheless produce different responses.

Studying the impact of P-Th and P-FS was the goal defined in the previous C.2.1 experiment. Therefore, in this section, they are used with a fixed value, defined as P-Th= 0.9 and P-FS=10, considering that this configuration surpassed the others.

With regard to the number of input agents, it takes the values 1, 5, 10, 15, and 20.

Unlike the previous experiment, this one processes video streams from six different data sets also used in the work (WU; LIM; YANG, 2013). These are Basketball, BlurBody, Coke, Girl2, Human4, and Human9. Each experiment is repeated 30 times to measure the mean and standard deviation for the two performance metrics, ACC and FPS.

Table 26 lists the main parameters applied for this experiment.

Table 26 – Parameters applied for scalability experiment.

| Parameter | Value |
|---|---|
| Executions | 30 |
| Qtd. agents | 1, 5, 10, 15, 20 |
| Data set | Basketball, BlurBody, Coke, Girl2, Human4, Human9 |
| Tracker | MIL |
| Attention | A-None, A-Rand, A-Ref, A-Fuzzy |
| P-FS | 10 |
| P-Th | 0.9 |

Source: The author.

Tables 27 and 28 show respectively the R-ACC and R-FPS results for the four different Attentions and datasets while varying the number of agents.

A close examination of Table 27 shows that independently of the used dataset, R-ACC grows according to the increase in the number of used agents. Understandably, the worst results are observed when there is a single input agent. Accuracy increase is mainly evidenced as more agents are inserted: starting with one agent and moving to have five agents. Such gain in accuracy rate does not persist as more agents are added. When conducting experiments where the number of agents ranged between 15 and 20, the gain in R-ACC is slowed down.

Furthermore, the results exhibited a considerable overlap in the standard deviation. We conclude that the increase in R-ACC tends to stabilize after a given number of agents, and to continue doing so is fruitless. Meanwhile, we observe that FPS is highly impacted by increasing the number of agents. For example, when using the None Attention, i.e., no actual agent scheduling or selection is performed, and while ranging the number of agents from 1 to 5 agents, R-FPS falls by 82%. Under A-Ref or A-Fuzzy Attention, we observe less R-FPS reduction, in the range of 26%. Note that even the most straightforward agent scheduling strategy, such as A-Rand, outperforms A-None in terms of achieved R-FPS despite remaining below the FPS levels achieved by A-Ref or A-Fuzzy.

Except for the Human4 dataset, the A-Rand always fell short in terms of R-ACC compared to the others. There was no significant difference when changing the number of agents or the Attentions when using this dataset. Even when adding more agents, R-ACC

Table 27 – R-ACC results for different datasets and an increasing number of agents.

| Dataset | Att. | Qtd Agents | | | | |
|---|---|---|---|---|---|---|
| | | 1 | 5 | 10 | 15 | 20 |
| Basket. | A-None | 0.37±0.3 | 0.85±0.19 | 0.95±0.09 | 0.98±0.03 | 0.99±0.03 |
| | A-Rand | 0.37±0.3 | 0.54±0.18 | 0.58±0.15 | 0.64±0.09 | 0.63±0.08 |
| | A-Ref | 0.37±0.3 | 0.73±0.17 | 0.84±0.08 | 0.91±0.04 | 0.94±0.05 |
| | A-Fuzzy | 0.37±0.3 | 0.73±0.19 | 0.86±0.1 | 0.91±0.06 | 0.93±0.04 |
| BlurB. | A-None | 0.58±0.18 | 0.83±0.09 | 0.9±0.08 | 0.93±0.06 | 0.94±0.05 |
| | A-Rand | 0.58±0.18 | 0.6±0.13 | 0.66±0.06 | 0.67±0.04 | 0.69±0.04 |
| | A-Ref | 0.58±0.18 | 0.81±0.06 | 0.86±0.04 | 0.88±0.04 | 0.9±0.03 |
| | A-Fuzzy | 0.58±0.18 | 0.78±0.1 | 0.87±0.04 | 0.88±0.04 | 0.9±0.02 |
| Coke | A-None | 0.85±0.09 | 0.95±0.04 | 0.97±0.02 | 0.98±0.02 | 0.99±0.01 |
| | A-Rand | 0.85±0.09 | 0.89±0.05 | 0.89±0.04 | 0.89±0.03 | 0.88±0.04 |
| | A-Ref | 0.85±0.09 | 0.87±0.08 | 0.91±0.05 | 0.93±0.03 | 0.93±0.03 |
| | A-Fuzzy | 0.85±0.09 | 0.89±0.07 | 0.9±0.04 | 0.92±0.04 | 0.93±0.04 |
| Girl2 | A-None | 0.37±0.12 | 0.61±0.09 | 0.69±0.09 | 0.74±0.08 | 0.76±0.05 |
| | A-Rand | 0.37±0.12 | 0.49±0.09 | 0.49±0.1 | 0.48±0.08 | 0.48±0.09 |
| | A-Ref | 0.37±0.12 | 0.64±0.1 | 0.73±0.11 | 0.79±0.11 | 0.79±0.1 |
| | A-Fuzzy | 0.37±0.12 | 0.67±0.12 | 0.73±0.11 | 0.8±0.11 | 0.81±0.11 |
| Human4 | A-None | 0.19±0.02 | 0.21±0 | 0.21±0 | 0.21±0 | 0.21±0 |
| | A-Rand | 0.19±0.02 | 0.2±0.01 | 0.2±0 | 0.2±0 | 0.2±0 |
| | A-Ref | 0.19±0.02 | 0.2±0 | 0.2±0.01 | 0.2±0.01 | 0.2±0 |
| | A-Fuzzy | 0.19±0.02 | 0.19±0.01 | 0.2±0.01 | 0.2±0.01 | 0.2±0.01 |
| Human9 | A-None | 0.31±0.16 | 0.45±0.07 | 0.47±0.04 | 0.48±0.04 | 0.5±0.04 |
| | A-Rand | 0.31±0.16 | 0.3±0.08 | 0.28±0.05 | 0.29±0.05 | 0.29±0.05 |
| | A-Ref | 0.31±0.16 | 0.51±0.02 | 0.52±0.01 | 0.53±0.01 | 0.53±0.01 |
| | A-Fuzzy | 0.31±0.16 | 0.5±0.04 | 0.52±0.02 | 0.53±0.02 | 0.53±0.01 |

Source: The author.

remained constant, situated about 2% below the case when no Attention is used (i.e., when A-None is used). This probably happens due to this dataset's low tracker accuracy level, which can lose track at some point, stabilizing independently of the number of used agents for input stream processing.

Comparing the inclusion of advanced Attentions with None, one expects to decrease in R-ACC once we are causing restrictions in the detection process when selecting only part of the agents and skipping some frames (as determined by the parameter P-FS). But in some cases, like in the Girl2 and Human9 datasets, the A-Ref or A-Fuzzy Attentions improved the R-ACC when using more than one agent. This could happen when the Attention skips occlusions that disturb the tracker detection.

Table 28 – R-FPS results for different datasets and an increasing number of agents.

| Dataset | Att. | Qtd Agents | | | | |
|---|---|---|---|---|---|---|
| | | 1 | 5 | 10 | 15 | 20 |
| Basket. | A-None | 38.34±1.79 | 6.85±0.15 | 3.43±0.08 | 2.31±0.05 | 1.74±0.04 |
| | A-Rand | 38.34±1.79 | 14.53±0.53 | 14.59±0.32 | 13.76±0.29 | 13.03±0.33 |
| | A-Ref | 38.34±1.79 | 29.46±1.06 | 25.05±1.5 | 22.53±1.64 | 19.69±1.9 |
| | A-Fuzzy | 38.34±1.79 | 29.18±1.09 | 25.59±1.44 | 23.09±1.44 | 21.75±1.5 |
| BlurB. | A-None | 36.21±1.24 | 6.71±0.22 | 3.32±0.04 | 2.26±0.04 | 1.71±0.03 |
| | A-Rand | 36.21±1.24 | 15.17±0.59 | 14.78±0.61 | 14.04±0.66 | 12.87±0.55 |
| | A-Ref | 36.21±1.24 | 28.08±1.3 | 24.18±1.43 | 20.4±1.73 | 18.99±1.64 |
| | A-Fuzzy | 36.21±1.24 | 28.51±1.22 | 23.48±1.43 | 20.27±1.49 | 18.89±2.48 |
| Coke | A-None | 37.92±0.23 | 6.89±0.02 | 3.46±0.01 | 2.3±0.01 | 1.72±0.01 |
| | A-Rand | 37.92±0.23 | 14.59±0.5 | 14.15±0.44 | 13.36±0.44 | 12.5±0.65 |
| | A-Ref | 37.92±0.23 | 29.4±1.05 | 26.43±1.52 | 23.4±1.18 | 22.08±2.09 |
| | A-Fuzzy | 37.92±0.23 | 29.95±0.96 | 28.04±1.41 | 26.08±1.61 | 24.95±1.65 |
| Girl2 | A-None | 39.33±2 | 7.27±0.11 | 3.64±0.04 | 2.45±0.04 | 1.85±0.02 |
| | A-Rand | 39.33±2 | 15.36±0.35 | 14.99±0.43 | 14.47±0.26 | 13.66±0.44 |
| | A-Ref | 39.33±2 | 29.18±0.98 | 24.24±1.17 | 19.77±0.89 | 17.01±0.94 |
| | A-Fuzzy | 39.33±2 | 29±0.89 | 24.11±1.44 | 19.77±0.97 | 16.97±1.56 |
| Human4 | A-None | 41.11±1.01 | 7.98±0.06 | 4.09±0.02 | 2.78±0.05 | 2.11±0.04 |
| | A-Rand | 41.11±1.01 | 16.83±0.37 | 16.37±0.41 | 15.88±0.41 | 14.16±1.82 |
| | A-Ref | 41.11±1.01 | 33.54±0.99 | 28.85±1.26 | 24.9±1.9 | 22.92±2.84 |
| | A-Fuzzy | 41.11±1.01 | 34±1.35 | 30.1±1.64 | 28.17±1.79 | 26.35±1.39 |
| Human9 | A-None | 35.06±1.55 | 6.82±0.09 | 3.53±0.02 | 2.37±0.05 | 1.87±0.01 |
| | A-Rand | 35.06±1.55 | 14.66±0.58 | 15.04±0.57 | 14.29±0.68 | 14.17±0.56 |
| | A-Ref | 35.06±1.55 | 27.9±1.08 | 23.52±1.62 | 21.06±1.92 | 19.7±2.3 |
| | A-Fuzzy | 35.06±1.55 | 27.85±0.91 | 24.41±1.56 | 23.38±2.61 | 22.91±2.8 |

Source: The author.

R-FPS falls with the increase in the number of agents due to the overhead of managing more agents and waiting for them to execute. Attentions can reduce such R-FPS loss as they primarily activate a subset of the total agents. For example, when using the Basketball dataset with the None Attention and varying the number of agents from 1 to 20, we manage 95% reduction in R-FPS while increasing the R-ACC 4.5 times. With A-Fuzzy, the achieved R-FPS reduction is 43% while the R-ACC increases 4.3 times. So comparing the None with and the A-Fuzzy with 20 agents, the R-ACC is 6% lower, but the R-FPS is 12.5 faster. This is the case of the Basketball dataset that has the most R-ACC discrepancy for A-None and the A-Fuzzy Attentions. In the other datasets, the gain for Attention use is even better in the R-ACC.

In other words, these Attentions managed R-ACC levels below the None Attention but benefit situations with more than one agent and can reduce the FPS level compared to None and the more simple A-Rand. In addition, in some cases, it could be beneficial to force the inclusion of more agents when the application requires a certain level of R-ACC.

### C.2.3 Different agents

In this next experiment, we seek to understand and highlight the impact of the A-Fuzzy when using agents with different processing rates. As a result, we consider the following five different trackers: MIL ((BABENKO; YANG; BELONGIE, 2009)), CSRT ((LUKEZIC et al., 2017)), KFC ((HENRIQUES et al., 2014)), TLD ((KALAL; MIKOLAJCZYK; MATAS, 2011)), and MedianFlow (MF - (KALAL; MIKOLAJCZYK; MATAS, 2010)). They have different R-ACC and R-FPS. The P-FS is fixed to 10 and the P-Th is set to 0.9. This experiment uses, five agents where each one executes a different tracker algorithm. We compare the R-ACC and R-FPS metrics obtained for all four Attentions: A-None, A-Rand, A-Ref, and A-Fuzzy. The number of repetitions of each experimental setup is set to 30.

Table 29 – Parameters applied for experiment with different trackers.

| Parameter | Value |
|---|---|
| Executions | 30 |
| Qtd. agents | 5 |
| Data set | BlurBody |
| Tracker | MIL, CSRT, KFC, TLD, MF |
| Attention | A-None, A-Rand, A-Ref, A-Fuzzy |
| P-FS | 10 |
| P-Th | 0.9 |

Source: The author.

This last set of experiments explores the advantage of using the fuzzy method for scheduling agents with different frame processing rates. Table 30 shows the results obtained from the individual execution of 5 different trackers initially without the application of the Attention, whereas Table 31 lists the results when using this A-Fuzzy Attention.

Let us compare the results of Table 31 where agents execute different tracker algorithms (hence have different FPS rates) with those representing a setup with for 5 agents using the same MIL tracker presented in Tables 27 28 (i.e. a scenario where all agents have the same FPS) for the BlurBody dataset. No Attention is considered at first. We observe that the diversified tracker scenario improves R-ACC from 0.83 to 1.0 and R-FPS from 6.71 to 16.06. This occurs due to the ensemble having more trackers to choose from. As a result, the ensemble module can choose trackers with better R-ACC in the heterogeneous tracker setup, such as the CSRT tracker agent. This is unlike the homogeneous

Table 30 – Results for the execution of different trackers individually.

| Tracker | R-ACC | | R-FPS | |
|---|---|---|---|---|
| | Mean | Std | Mean | Std |
| TLD | 0.81 | 0.11 | 81.48 | 3.53 |
| MIL | 0.57 | 0.17 | 34.01 | 1.40 |
| CSRT | 1.00 | 0.00 | 59.48 | 0.49 |
| KCF | 0.45 | 0.00 | 393.30 | 6.60 |
| MF | 0.60 | 0.00 | 2243.22 | 24.62 |

Source: The author.

Table 31 – Results for the execution of five different trackers at once and applying the Attentions.

| Att. | R-ACC | | R-FPS | |
|---|---|---|---|---|
| | Mean | Std | Mean | Std |
| A-None | 1.00 | 0.00 | 16.06 | 0.28 |
| A-Rand | 0.87 | 0.03 | 156.21 | 25.07 |
| A-Ref | 0.98 | 0.01 | 102.68 | 24.63 |
| A-Fuzzy | 0.94 | 0.06 | 383.27 | 127.96 |

Source: The author.

experiment where the Attention is limited by the FPS and ACC output from the MIL tracker. In other words, the use of different types of trackers offers a diversity of results to the Attention to choose among them, which results in better R-FPS performance.

Analyzing Table 31 we see that the A-Fuzzy method outperforms the others and achieves the best R-FPS at the cost of reduced R-ACC. In fact, it suffers around a 6% R-ACC loss as compared to when using the A-None. Nonetheless, the A-Fuzzy Attention could significantly improve R-FPS by as much as 3.7, 2.4 and 23.9 times against A-Ref, A-Rand and A-None methods, respectively. This is mainly due to having agents with diverse frame processing rates. For example, The MF Tracker exhibits a considerable FPS gain relative to the others. As a result, the A-Fuzzy leverages such difference and selects this tracker when it may strike a good tradeoff between the achieved R-FPS gain and the R-ACC level obtained. Note that the A-Ref Attention does not establish such a tradeoff as it selects agents based only on the ACC metric.

# APPENDIX D – RASPBERRY PI CONFIGURATION

This section describes the raspberry configuration to run the proposed architecture. First, installing the Operational System (OS) in the Raspberry Pi is necessary. Therefore, this is done by following the step proposed in the Raspberry Pi Foundation[1]. A Debian-based SO (Xubuntu SO) is used as a host PC to make the installation process. Also, to run the complete system proposed in Chapter 5, it is necessary to configure the network interfaces and install the required libraries.

## D.1 WRITE THE OS IN THE SD CARD

1. Download the Raspberry Pi Imager.

    - https://www.raspberrypi.com/software/

2. Install the Raspberry Pi Imager in the host machine.

    - In an linux terminal go to the folder where the Imager was downloaded and type:

```
$ sudo apt install rpi-imager
```

3. Insert the memory card in the host machine.

    - It is recommended that the memory card has at least 16GB (Class 10 or 4).

4. Open the Pi Image.

    - Figure 54 illustrates the Pi Image interface.

5. Select the OS to install in the Raspberry.

    - This work consider the first option in the list Raspberry Pi OS (32-bit) (Released: 2021-1030)

6. Choose the Storage.

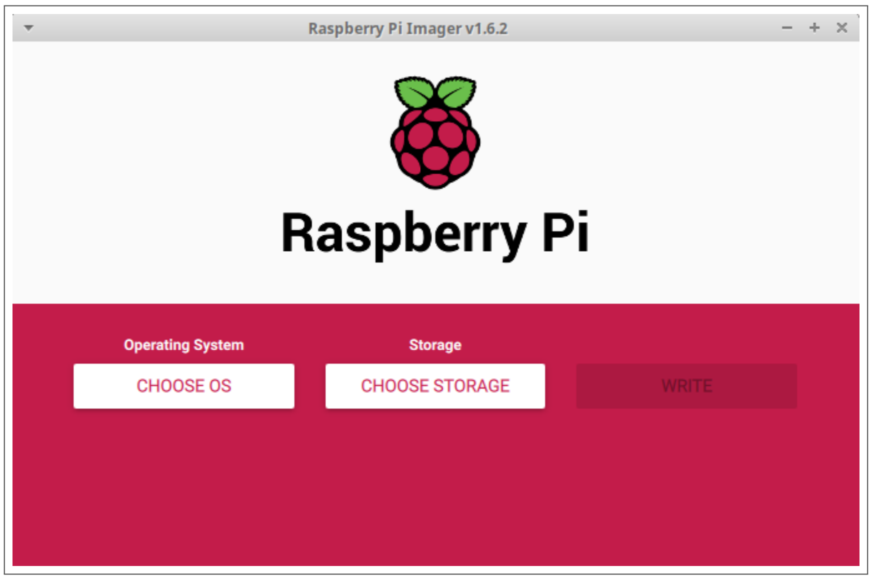
    - In this case, choose the memory card that was inserted in the configuration PC in step 3.

7. Start SD Card writing.

    - Click on the WRITE button on Raspberry Pi Imager.

[1] https://www.raspberrypi.com/software/

Figure 54 – Raspberry Pi Imager interface.



Source: The author.

- It alerts that all existing data on the device will be erased. Click YES (Assuming that you already considered using a memory card without important data).
- The writing process can take about 15 minutes.

8. End of SD card writing.

   - After the writing process, a successful message appears. Click on CONTINUE and remove the SD card.

## D.2 INSTALL AND CONFIGURATION

1. Initial steps.

   - Connect a screen, mouse, and keyboard to use in the installation.
   - Insert the SD Card in the Raspberry.
   - Connect the power cable.
   - It will initiate the SO and start the install routine.

2. If raspberry request the user to log in, use the following default parameters:

   - user: pi
   - keyword: raspberry

3. Initial configuration window.

- There is an alert with a welcome message window, from which one can make the initial configuration.

4. Set the country, language, and timezone
5. Enter a new password. It is highly recommended to change the default password to avoid security issues.
6. Setup screen.
7. Select WiFi to connect the raspberry
8. Update the software. This step could take a while.
9. Finally, a message Setup Complete appears, so just click in done.

## D.3 CONFIGURE REMOTELY CONTROL

If the user wants to control the raspberry from another computer using a user graphical interface visualization, it is recommended to use the VNC software. Alternatively, the user can control using the SSH connection. For that, it is required to allow its use in the Raspberry.

1. Click in the raspberry icon in the up left side corner.
2. Click in Preferences.
3. Click in Raspberry Pi Configuration.
4. Go to the tab Interfaces.
5. Enable SSH e/or VNC.

The user needs to have the raspberry IP address to visualize the raspberry interface in a remote machine. It can be taken by clicking on the VNC icon beside the clock in the upright corner (Fig. 55) or using one of the following terminal commands:

```
$ ifconfig
```

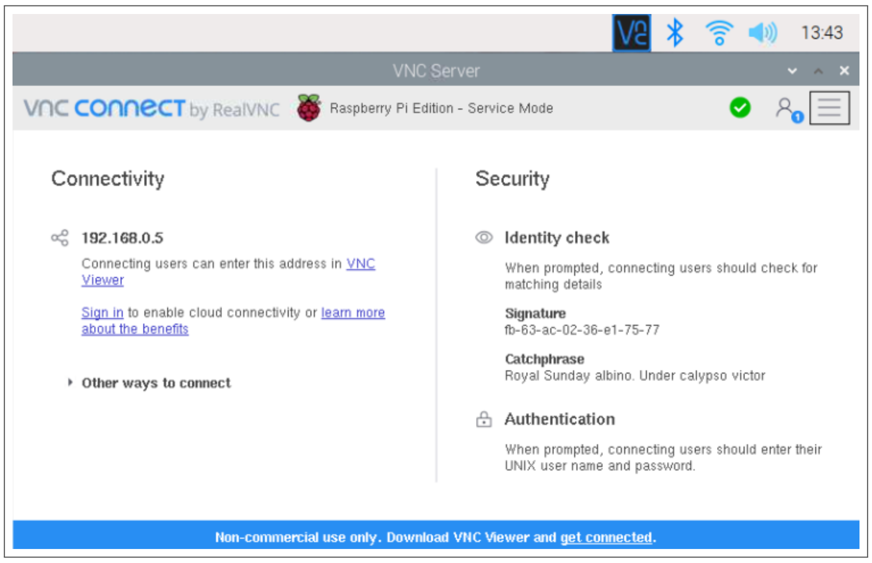
```
$ ip addr
```

Fig. 56 shows the result of the *ifconfig* command.

In the host machine, the user has to follow the following steps, considering a Debian/Linux based host machine:

1. Open the VNC server software.
2. Set the IP address of the raspberry pi, remind that the machines have to be in the same network.

Figure 55 – VNC connect interface.



13:43

VNC Server

vnc connect by RealVNC Raspberry Pi Edition - Service Mode

Connectivity

192.168.0.5

Connecting users can enter this address in VNC Viewer

Sign in to enable cloud connectivity or learn more about the benefits

Other ways to connect

Security

Identity check

When prompted, connecting users should check for matching details

**Signature**
fb-63-ac-02-36-e1-75-77

**Catchphrase**
Royal Sunday albino. Under calypso victor

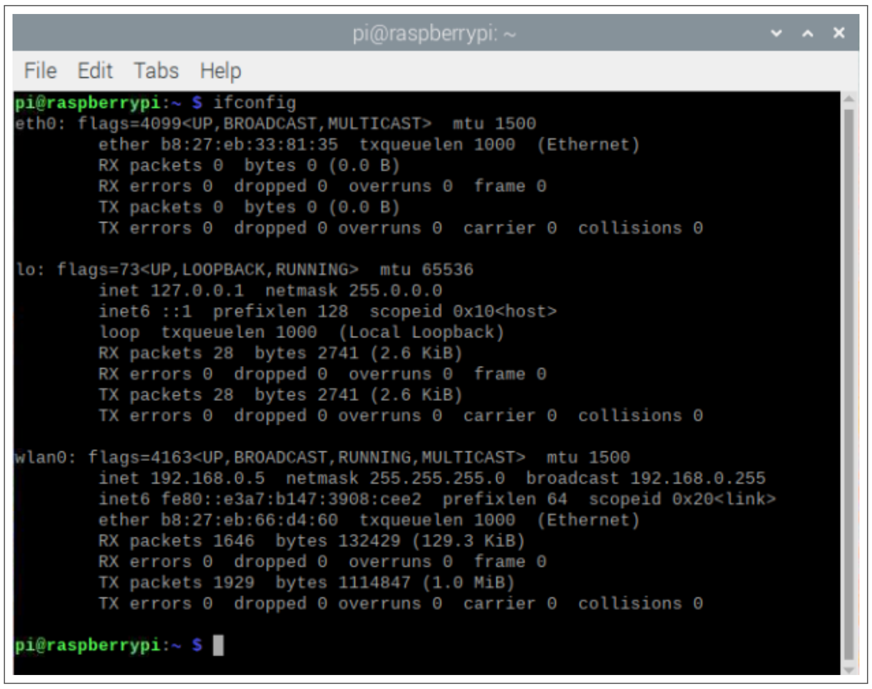
Authentication

When prompted, connecting users should enter their UNIX user name and password.

Non-commercial use only. Download VNC Viewer and get connected.

Source: The author.

Figure 56 – Response in the terminal for ifconfig command.



pi@raspberrypi: ~

File Edit Tabs Help

```
pi@raspberrypi:~ $ ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether b8:27:eb:33:81:35  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 28  bytes 2741 (2.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 28  bytes 2741 (2.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.5  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::e3a7:b147:3908:cee2  prefixlen 64  scopeid 0x20<link>
        ether b8:27:eb:66:d4:60  txqueuelen 1000  (Ethernet)
        RX packets 1646  bytes 132429 (129.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1929  bytes 1114847 (1.0 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

pi@raspberrypi:~ $ 
```

Source: The author.

3. Then, it will request the user and password of the Raspberry.

4. Now, the user can control the Raspberry with a graphical interface.

To connect using the SSH in the host machine, the user needs to open a terminal and write the command:

```
$ login@ip
```

Considering the IP address taken in previous steps (Fig. 55 or Fig. 56) the command would be:

```
 $ ssh pi@192.168.0.5
```

After that, the user is asked to write the raspberry password, and then it is done.

## D.4 CONFIGURE NETWORK

The Raspberry network configuration is required due to the use of networked cameras and communication with the server (Chapter 5). For the proposed configurations in the scenario, the Raspberry must deal with two network interfaces, one for communicating with the cameras other for communicating with the server network. The Rasberry Pi 3 B+ (Table 3) offers the required two interface. The ethernet is used to communicate with the cameras, and the WiFi is used with the Router (Table 3). Choosing the edge device carefully is an important task, as old versions of the Raspberry (like versions 1 and 2) do not offer an integrated WiFi interface, requiring an external USB WiFi adapter.

For the Raspberry to receive the camera stream and simultaneously communicate with the router, the ethernet interface must be in a different subnetwork than the WiFi communication, or it can occur that it only uses one of the interfaces. For that, the user must configure an Ethernet interface's static IP. Also, it must define the interface with the highest priority, in our case, the WiFi.

To make the IP static edit the file */etc/dhcpcd.conf*.

```
$ sudo nano /etc/dhcpcd.conf
```

The user can use the configuration in the Listing 16. The interface with lower values in the tag *metric* has higher priority. The *eth0* is configured to have a static IP in the subnetwork *192.169.1.x*, and the *wlan0* interface keeps a dynamic IP, in the defalt network textit192.169.0.x.

Listing 16 – Network configuration file (/etc/dhcpcd.conf).

```
interface eth0
metric 302
static ip\_address=192.168.1.100
static routers=192.168.1.254
static domain\_name\_servers=192.168.1.254 8.8.8.8

interface wlan0
metric 202
```

Then the user must to restart the network interfaces:

```
$ sudo /etc/init.d/networking restart
```

Finally, the camera can be plugged into the ethernet interface. The user can test if raspberry can receive the stream by following the steps:

1. Open the VLC Media Player.

    - It comes pre-installed on the Raspberry PI.

2. In the menu bar go to Media -> Open Network Stream
3. In the URL space you have to write the stream link.
    - Depending on the camera model, the values can change.
    - The required parameters are: camera IP and Port, login, and password.
    - Example: rtsp://login:password@192.168.0.7:88/videoMain
4. Press Play. The stream must start in the VLC interface.

## D.5 INSTALL LIBRARIES

It is recommended to create a virtual environment to be safe from a complete OS reinstall in case of problems in some libraries. With a virtual environment, in case of errors, it is just necessary to create a new environment and reinstall the libraries.

The miniconda [2] virtual environment is used in this work. It is optimized for the raspberry pi arm processor.

1. Download start the miniconda install process:

```
$ wget http://repo.continuum.io/miniconda/Miniconda3-latest-Linux-armv7l.sh
$ sudo md5sum Miniconda3-latest-Linux-armv7l.sh
$ sudo /bin/bash Miniconda3-latest-Linux-armv7l.sh
```

2. Installation parameters:
    - In order to continue the installation process, please review the license agreement.
        - Press Enter
    - Do you approve the license terms? [yes|no]
        - Write yes, then press ENTER
    - Miniconda3 will now be installed into this location: [/root/miniconda3]:
        - Write /home/pi/miniconda3, them press ENTER
    - Do you wish the installer to prepend the Miniconda3 install location to PATH in your /root/.bashrc ? [yes|no]
        - Write yes, then press ENTER
3. Open the .bashrc file to ddd the install path to the PATH variable.

```
$ sudo nano /home/pi/.bashrc
```

[2] https://www.anegron.site/2020/06/18/how-to-install-conda-and-docker-on-your-raspberry-pi/

4. Go to the end of the file .bashrc and add the following line:

```
export PATH="/home/pi/miniconda3/bin:$PATH"
```

5. Save the .bashrc file and exit.

6. Close the terminal.

7. To test if the installation was successful, open a new terminal, write the command *$ source* and press enter.

Install the python 3.6 and create an conda virtual environment.

1. Install the python 3.6

```
$ sudo su
$ conda config --add channels rpi
$ conda install python=3.6
```

2. Close the super user sectio press crtl + d.

3. Create the conda enviroment.

```
$ conda create --name safety python=3.6
```

4. Activate the environment.

```
$ source activate safety
```

Install required libraries:

```
$ pip install paho-mqtt
$ pip install numpy
$ pip install imagezmq
```

Install opencv:

1. Select OpenCV version to install and prepare the system for the installation. The version used in the project was opencv 4.5.4

   - Clean unnecessary packages in the OS.

   ```
   $ sudo apt-get -y purge wolfram-engine
   $ sudo apt-get -y purge libreoffice*
   $ sudo apt-get -y clean
   $ sudo apt-get -y autoremove
   $ sudo apt-get -y remove x264 libx264-dev
   ```

2. Update Packages.

```
sudo apt -y update
sudo apt -y upgrade
```

3. Install required OS dependencies.

```
sudo apt-get -y install build-essential checkinstall cmake pkg-config yasm
sudo apt-get -y install git gfortran
sudo apt-get -y install libjasper-dev
sudo apt-get -y install libtiff5-dev
sudo apt-get -y install libtiff-dev
sudo apt-get -y install libavcodec-dev libavformat-dev libswscale-dev
    libdc1394-22-dev
sudo apt-get -y install libxine2-dev libv4l-dev
cd /usr/include/linux
sudo ln -s -f ../libv4l1-videodev.h videodev.h
cd $cwd
sudo apt-get -y install libgtk2.0-dev libtbb-dev
sudo apt-get -y install libatlas-base-dev
sudo apt-get -y install libmp3lame-dev libtheora-dev
sudo apt-get -y install libvorbis-dev libxvidcore-dev libx264-dev
sudo apt-get -y install libopencore-amrnb-dev libopencore-amrwb-dev
sudo apt-get -y install libavresample-dev
sudo apt-get -y install x264 v4l-utils
```

4. Install Optional OS dependencies.

```
sudo apt-get -y install libprotobuf-dev protobuf-compiler
sudo apt-get -y install libgoogle-glog-dev libgflags-dev
sudo apt-get -y install libgphoto2-dev libeigen3-dev libhdf5-dev doxygen
```

5. Install Python Libraries

```
sudo apt-get -y install python3-dev python3-pip
sudo -H pip3 install -U pip numpy
sudo apt-get -y install python3-testresources
```

6. Increasing SWAP memory space.

   - Open up the /etc/dphys-swapfile file:

     ```
     sudo nano /etc/dphys-swapfile
     ```

   - Set the size SWAP size to 2048 in the file: CONF_SWAPSIZE=2048
   - restart the SWAP memory:

     ```
     sudo /etc/init.d/dphys-swapfile stop
     sudo /etc/init.d/dphys-swapfile start
     ```

7. Download opencv and opencv_contrib

```
git clone https://github.com/opencv/opencv_contrib.git
cd opencv_contrib
git checkout $cvVersion
cd ..

git clone https://github.com/opencv/opencv.git
cd opencv
```

```
git checkout $cvVersion
mkdir build
cd build
```

8. Compile and install OpenCV with contrib modules. First navigate to the build directory. Next, start the compilation and installation process.

```
cmake -D CMAKE_BUILD_TYPE=RELEASE \
    -D CMAKE_INSTALL_PREFIX=/usr/local \
    -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib/modules \
    -D ENABLE_NEON=ON \
    -D ENABLE_VFPV3=ON \
    -D BUILD_TESTS=OFF \
    -D OPENCV_ENABLE_NONFREE=ON \
    -D INSTALL_PYTHON_EXAMPLES=OFF \
    -D HAVE_opencv_python3=ON \
    -D PYTHON3_EXECUTABLE=/home/pi/.conda/envs/safety/bin/python \
    -D PYTHON_DEFAULT_EXECUTABLE=/home/pi/.conda/envs/safety/bin/python3.6 \
    -D BUILD_EXAMPLES=OFF ..
make -j$(nproc)
make install
```

9. Reset SWAP memory size to the previous value (CONF_SWAPSIZE=100), as in step 6.

    - Once have finished the install of heavy Python modules like Numpy, it's time to reset the swap file.

10. Sym-link your OpenCV on the Raspberry Pi

```
cd /usr/local/lib/python3.6/site-packages/cv2/python-3.6/
sudo mv cv2.cpython-36m-arm-linux-gnueabihf.so cv2.so
cd ~/.conda/envs/safety/lib/python3.6/site-packages/
ln -s /usr/local/lib/python3.6/site-packages/cv2/python-3.6/cv2.so cv2.so
```

# APPENDIX E – CAMERA CONFIGURATION

This tutorial reports the configuration process to the Foscam R2M camera (described in Table 3) in the scenario presented in Chapter 5. When this guide was written, the FOSCAM software [1] had support for OSs Windows, Mac, and Android. The configuration of cameras required communication with the Foscam server. However, our application required that we have only internal communication in the local network for security requirements. This could be done only using the Windows OS Foscam software. If the user application does not have this restriction, you can only download the android version [2] and follow the steps in the app.

1. Download the software.
2. Install it.
3. Configure Foscam software login and password.
4. Log on Foscam software.
5. For the first-time configuration, a camera restart could be necessary. For that, press and hold the reset button in the camera for about 5 seconds.
6. To add the camera, press the green button with an "+" (Fig. 57).
7. Choose the camera and click in Add (Fig. 58).
8. Set the camera username and password. Then click in Create (Fig. 59).
9. The camera is added. Now the user can see the stream on the screen.

Some camera settings (Fig. 60) adaptations can be made using the same software to attend to the specification used in Chapter 5).

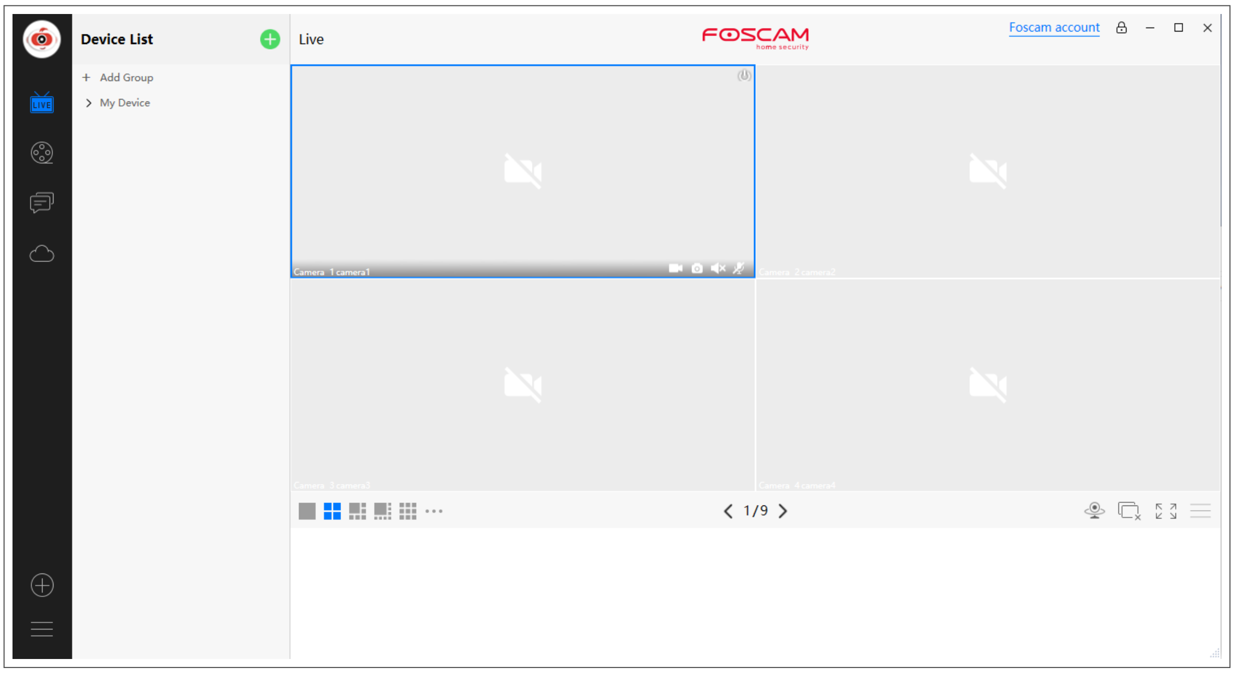
- The user can change the device name in:
  - .Basic -> Device name.
- The user can change the video stream configuration, as resolution, bit rate and frame rate in:
  - .Video -> Video encode
- The user can Remove the display timestamp and Display device name in:

[1] https://www.foscam.com/downloads/app_software.html
[2] https://play.google.com/store/search?q=foscam&c=apps&hl=en&gl=US

  - .Video -> On screen display

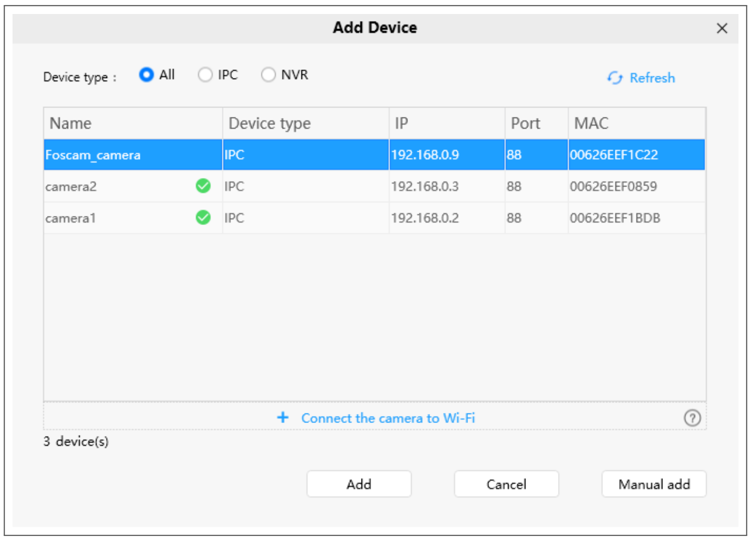
- The user has to define a static IP for the camera to communicate with the Raspberry PI using the pre-defined range for subnetwork in:
  - .Network -> IP (Fig. 61).
  - Define the item "Obtain IP from DHCP" to No.
  - Define the item IP Address, Gateway, and Primary DNS Server.
  - Remember that such subnetwork must be the same that the raspberry pi can access using the ethernet interface, as described in Section D.4.
  - For the last press Save. After that, if the camera is defined as a subnetwork different from the router network, the camera will no longer be connected to the router network. Therefore, the network configuration step must be the last one after all camera configurations.

Figure 57 – Foscam Software interface.



Source: The author.

Figure 58 – Foscam Software interface - Camera options.



Add Device

Device type : [✓ All] IPC NVR

Refresh

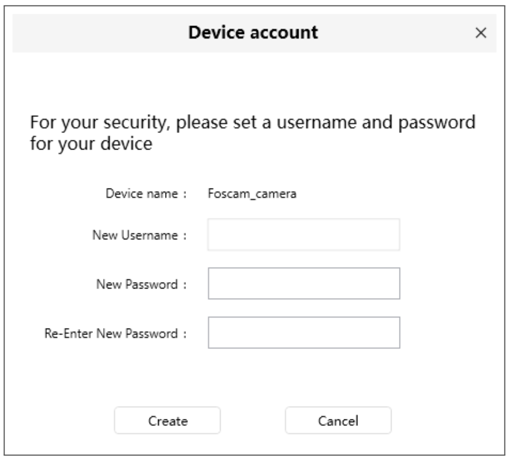
| Name | Device type | IP | Port | MAC |
|---|---|---|---|---|
| Foscam_camera | IPC | 192.168.0.9 | 88 | 00626EEF1C22 |
| camera2 ✓ | IPC | 192.168.0.3 | 88 | 00626EEF0859 |
| camera1 ✓ | IPC | 192.168.0.2 | 88 | 00626EEF1BDB |

+ Connect the camera to Wi-Fi

3 device(s)

Source: The author.

Figure 59 – Foscam Software interface - Set Username and Password.



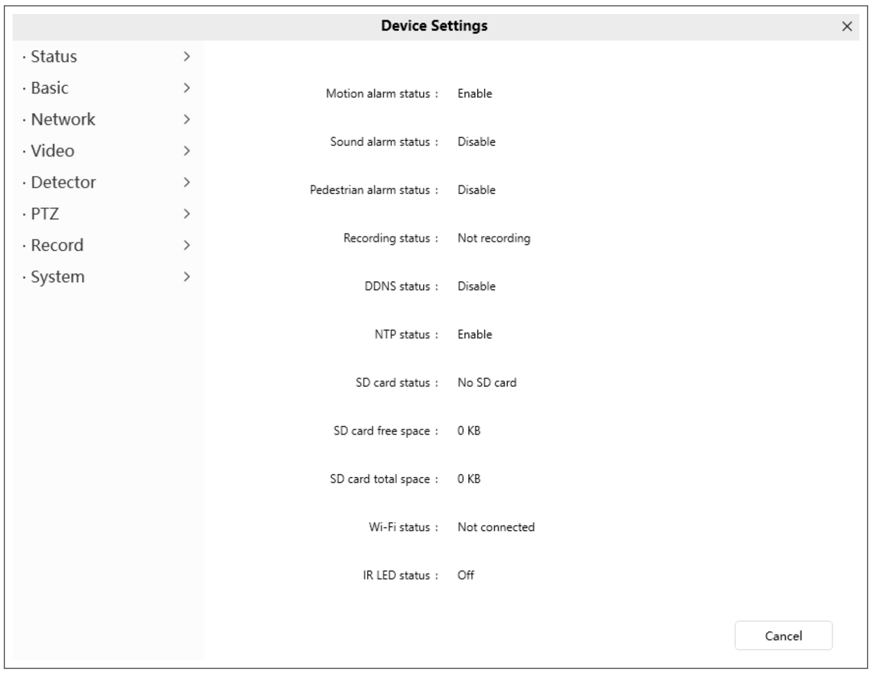
Device account

For your security, please set a username and password for your device

| | |
|---|---|
| Device name : | Foscam_camera |
| New Username : | |
| New Password : | |
| Re-Enter New Password : | |

Source: The author.

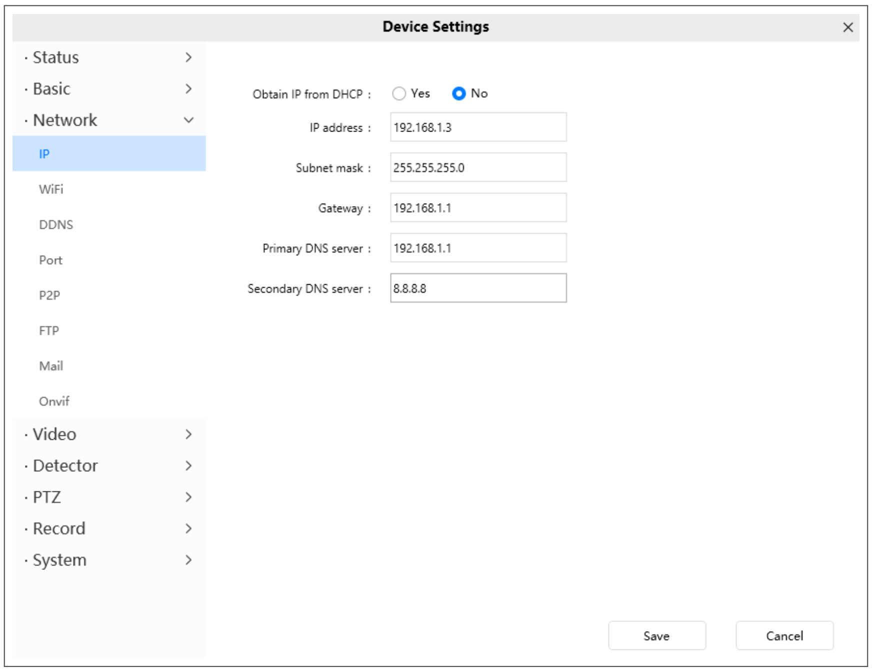
Figure 60 – Foscam Software interface - Camera Settings.



| | |
|---|---|
| Motion alarm status : | Enable |
| Sound alarm status : | Disable |
| Pedestrian alarm status : | Disable |
| Recording status : | Not recording |
| DDNS status : | Disable |
| NTP status : | Enable |
| SD card status : | No SD card |
| SD card free space : | 0 KB |
| SD card total space : | 0 KB |
| Wi-Fi status : | Not connected |
| IR LED status : | Off |

Source: The author.

Figure 61 – Foscam Software interface - Network Settings.



| | |
|---|---|
| Obtain IP from DHCP : | Yes [✓ No] |
| IP address : | 192.168.1.3 |
| Subnet mask : | 255.255.255.0 |
| Gateway : | 192.168.1.1 |
| Primary DNS server : | 192.168.1.1 |
| Secondary DNS server : | 8.8.8.8 |

Source: The author.