Context-aware intelligent video analysis for the management of smart buildings

Roberto Enrique Marroquín Cortez

THÈSE DE DOCTORAT DE L'ÉTABLISSEMENT UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ

PRÉPARÉE À L'UNIVERSITÉ DE BOURGOGNE

École doctorale n°37

Sciences Pour l'Ingénieur et Microtechniques

Doctorat d'Informatique et Instrumentation de l'Image

par

ROBERTO ENRIQUE MARROQUÍN CORTEZ

Context-aware intelligent video analysis for the management of smart buildings

Thèse présentée et soutenue à Dijon, le 18 October 2019

Composition du Jury :

GESQUIÈRE GILLES	Professeur à l'Université Lumière Lyon 2	Président
BOUCHAFA-BRUNEAU SAMIA	Professeure à l'Université d'Evry Val d'Essonne	Rapportrice
SÈDES FLORENCE	Professeure à l'Université de Toulouse Paul Sabatier	Rapportrice
DUBOIS JULIEN	Maître de Conférences HDR à l'Université de Bourgogne	Directeur de thèse
NICOLLE CHRISTOPHE	Professeur à l'Université de Bourgogne	Codirecteur de thèse

N° | X | X | X |

école doctorale **sciences pour l'ingénieur et microtechniques**

Title: Context-aware intelligent video analysis for the management of smart buildings

Keywords: Smart camera, artificial intelligence (AI), building information modeling (BIM), ontology engineering, real-time knowledge fusion, people detection, people tracking, event detections, multi-camera multi-space dataset.

Abstract:

To date, computer vision systems are limited to extract digital data of what the cameras "see". However, the meaning of what they observe could be greatly enhanced by environment and human-skills knowledge.

In this work, we propose a new approach to cross-fertilize computer vision with contextual information, based on semantic modelization defined by an expert.

This approach extracts the knowledge from images and uses it to perform real-time reasoning according to the contextual information, events of interest and logic rules. The reasoning with image knowledge allows to overcome some problems of computer vision such as occlusion and missed detections and to offer services such as people guidance and people counting. The proposed approach is the first step to develop an "all-seeing" smart building that can automatically react according to its evolving information, i.e., a context-aware smart building.

The proposed framework, named WiseNET, is an artificial intelligence (AI) that is in charge of taking decisions in a smart building (which can be extended to a group of buildings or even a smart city). This AI enables the communication between the building itself and its users to be achieved by using a language understandable by humans.

Titre : Analyse vidéo en temps-réel intégrant les données contextuelles pour la gestion de bâtiments intelligents

Mots-clés : Caméra Intelligente, intelligence artificielle (IA), modélisation des informations sur le bâtiment (BIM), ingénierie d'Ontologies, fusion de connaissances en temps réel, detection de personne, suivi de personne, détection d'événement, multi-space multi-camera dataset.

Résumé :

Les systèmes de vision artificielle sont aujourd'hui limités à l'extraction de données issues de ce que les caméras « voient ». Cependant, la compréhension de ce qu'elles voient peut être enrichie en associant la connaissance du contexte et la connaissance d'interprétation d'un humain.

Dans ces travaux de thèse, nous proposons une approche associant des algorithmes de vision atificielle à une modélisation sémantique du contexte d'acquisition.

Cette approche permet de réaliser un raisonnement sur la connaissance extraite des images par les caméras en temps réel. Ce raisonnement offre une réponse aux problèmes d'occlusion et d'erreurs de détections inhérents aux algorithmes de vision artificielle. Le système complet permet d'offrir un ensemble de services intelligents (guidage, comptage...) tout en respectant la vie privée des personnes observées. Ces travaux forment la première étape du développement d'un bâtiment intelligent qui peut automatiquement réagir et évoluer en observant l'activité de ces usagers, i.e., un bâtiment intelligent qui prend en compte les informations contextuelles.

Le résultat, nommé WiseNET, est une intelligence artificielle en charge des décisions au niveau du bâtiment (qui pourrait être étendu à un groupe de bâtiments ou même a l'échelle d'un ville intelligente). Elle est aussi capable de dialoguer avec l'utilisateur ou l'administrateur humain de manière explicite.

UBFC UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ

Université Bourgogne Franche-Comté
32, avenue de l'Observatoire
25000 Besançon, France

ACKNOWLEDGEMENTS

I wish to thank the members of my jury for agreeing to read the manuscript and to participate in the defense of this thesis. I am very grateful to Samia Bouchafa-Bruneau et Florence Sèdes for generously offering their time and efforts for the review of this manuscript. I equally extend my gratitude to the other member of the jury Gilles Gesquière for his support and interest in this thesis.

Firstly, I would like express my immense gratitude to my supervisors Julien Dubois and Christophe Nicolle for their motivation and enthusiasm in my research, for their patience and understanding, as well as for always being available and for sharing all their knowledge in their respective domains. Moreover, I would like to thank them for all the professional and personal guidance they taught me.

I also would like to thank the Université de Bourgogne and the French Ministry of Education for providing me the opportunity and funding to pursue this research.

Moreover, I am grateful to my family (Salvadorian and French) for all the motivation words, their support, their love and for always being there. Also, I would like to thank all my friends (from the lab, la copro and from life in general) for their support, their happiness and their craziness. Finally, I would like to thank my love Chloé for her motivation, her love, her help, but most importantly for her tolerance and her courage for supporting me during my thesis.

Contents

1

INTRODUCTION

The intelligence deployed at buildings, has brought the concept of *smart building*. In the past few decades, this term has been used for referring to a building equipped with a network of devices to improve its energy efficiency [158, 119, 178, 70]. In general, the smart building consists of:

- Sensors: devices used for measuring parameters/features of the environment (e.g., a thermometer);
- Actuators: devices used for performing a physical action (e.g., opening a window);
- Controllers: set of programmed rules used for controlling the actuators (also known as system's response);
- Central unit: enables the programming of the different system's units;
- Interface: allows the user to communicate with the system;
- Network: enables the communication between the units.

Building managers (also known as facility managers) are concerned with the people using the space. Their main duties are: ensuring the safety of the building's occupants and visitors, oversee the security system and ensuring the maintenance and repair of building elements. Consequently, a smart building should provide services that assist the building managers to perform their tasks, as well as, make the life of the users easier [70, 118].

Smart buildings are not restricted to energy efficiently applications (even though is the most common one). Specifically, after the boom of the Internet of Things (IoT), new smart building applications have emerged (e.g., access control, surveillance, activity monitoring, smart lighting, localization), using multiple sensing devices (e.g., radio-frequency tags and readers, beacons, passive infra-red, smart phones, 2D/3D cameras) [22, 62, 96, 2].

Visual data coming from cameras is very rich, since multiple and heterogeneous information can be extracted from it. For example, from a single image, it is possible to know all the objects present in the scene, their status and the relations between them. In a smart building context, the integration of multiple data extracted from a Visual Sensor Network (VSN) is a challenge [51, 181]. The most widespread and well-known application of this type of networks is video surveillance. In this type of application, the videos captured by the VSN are usually analyzed in real-time by a human operator in a monitor room, using dozens of screens [37]. However, as the size of the network increases, it becomes

difficult, even impossible, for a human being to monitor all the video streams at the same time and to quickly identify events.

A solution to this problem is to rely on Intelligent Video Surveillance (IVS) systems, to detect “abnormal” or “interesting” behaviors in the observed scene [36, 145, 174]. This type of systems might take advantage of the great advances in computer vision, and in general the advances of Artificial Intelligence (AI), to automatically detect the pertinent information from the scene. For example, a Convolutional Neural Network (CNN) could be used for detecting a bag in a lobby or a person in a space [103, 142]. There exists two types of AI: (1) *symbolic IA*, based on the knowledge of common sense or human learning. This type of AI enables to check logical hypotheses, by using inferences, to identify new knowledge by causal relation. In this AI system, to find the balance between the level of expressivity and the level of decidability is a complex task and impact the performance. Moreover, the associated models, such as ontology are domain dedicated and lack of generalization [163]; (2) *non-symbolic AI*, which learns by example/data. This type of AI are high dimensional models, which present high generalization and gives good results in many applications. However, their decision processes are (almost) impossible to decrypt by humans, thus they are also called "black box AI". Some examples are deep-learning based model, and in general any Artificial Neural Networks (ANN) [89]. Most of recent (almost all) computer vision systems are based on non-symbolic AI [89, 103, 142].

The use of non-symbolic AI systems in a smart building (and in general), raises some open questions. Firstly, a question concerning the lack of contextual information: Is it possible to consider external and heterogeneous information during the decision process of an AI system? Secondly, some questions concerning the understanding, interpretability and communication with the non-symbolic AI system: Is there a way to better understand and control the decision process of AI systems? How can we interact with it? How can we insert human reasoning in it? As Professor Hawking said “*The rise of powerful AI will be either the best, or the worst thing, ever to happen to humanity. We do not yet know which*”.[1] Which for us refers to the lack of understanding and interaction with non-symbolic AIs. Finally, one question may arise: Is it possible two combine both types of AI in an application which requires, interpretability, performance and generalization?

In this context, we propose a semantic-based AI framework, that combines the information coming from a network of cameras with the contextual information of the environment, applied in a smart building application. The proposed framework is a “transparent” AI which allows to interact with it using a human understandable language, and allows the use of non-symbolic AI to extract information from the scene.

Finally, we consider that a real smart building should not be limited to connecting sensor information, but also it should be able to extract, interpret and understand the contextual information, to automatically adapt its services accordingly.

[1] “*The best or worst thing to happen to humanity*” - Stephen Hawking launches Centre for the Future of Intelligence, October 19, 2016, https://www.cam.ac.uk/research/news/the-best-or-worst-thing-to-happen-to-humanity-stephen-hawking-launches-centre-for-the-future-of

1.1/ MOTIVATION AND OBJECTIVES

The use of a multi-camera based system is not a trivial task and several limitations to these systems have been identified. In any system generating a large amount of heterogeneous information (such as visual), the selection of significant data remains a problem. Another limitation, is the integration of data coming from the different cameras, and the integration of non-visual data (e.g., context). Besides, the cameras should be able to adjust their configuration according to the tasks required. Finally, the use of multi-camera systems always poses the problem of invasion of privacy.

Many efforts have being devoted to deal with some of the aforesaid limitations. The most prominent one is to rely on smart cameras to extract, in a semi-autonomously manner (with minimal human interaction), the pertinent information from a scene, by executing computer vision algorithms. Furthermore, smart cameras are able to execute (and reconfigure) multiple computer vision algorithms according the task of interest.

However, the information extracted from visual data lacks of contextual information. Context is everything that helps the understanding of an action/situation, thus, it is an essential factor in decision-making. In a built environment, context is very heterogeneous, and it refers to information about the structure of the building, information about the events that have occurred and human-skill knowledge that facilitate the analyze of a situation.

Beyond the work of extracting information from the cameras, we are interested in the deduction of new knowledge by aggregating information from multiple heterogeneous sources. In other words, our ambition is to combine the information from a smart camera network with contextual information to automatically understand what is happening in the environment.

In that regard, we propose the creation of a semantic-based framework, called WiseNET, that enables the interoperability between the heterogeneous sources of information, and reasons over the aggregation of knowledge. The main objectives of WiseNET are: (1) to enable interoperability between the heterogeneous sources of information, (2) to perform real-time reasoning over the aggregation of information, (3) to enhance classical computer vision by considering the contextual information of the environment, and (4) to provide a set of innovative services to building managers, to ease their work. As a result, WiseNET overcomes some limitations of computer vision—such as occlusions—as well as the previously presented limitations of multi-camera based systems. Furthermore, the WiseNET system enables building managers to perform queries, in real-time, to have information related to the environment and its usage.

In a few words, the proposed semantic-based system allows a smart building, equipped with a camera network, to *know beyond seeing*.

1.2/ DISSERTATION OUTLINE

This work is positionned in the intersection of two vasts domains: computer vision and knwoledge representation and reasonning. Therefore, the thesis starts by an introduction and state-of-the-art of each of these domains, presented in Chapter 2 and Chapter 3, respectively. Follows by, our proposition on how to combine the domains, presented in Chapter 4 and Chapter 5. And finishes by, presenting the advantages of the combination,

applied in a smart building.

Chapter 2 starts by introducing some important concepts that will be used along all the thesis: Visual Sensor Networks (VSN) and Intelligent Video Surveillance (IVS) systems. Followed by a discussion about the different types of smart cameras, specifically about their hardware. Finally, computer vision is introduced, where we focus on the image recognition task. Specifically, a state-of-the-art of feature-based and deep-learning-based people detection methods are presented. As well as, a state-of-the-art of people tracking methods. The chapter finishes with an example showing that classical computer vision methods lacks of contextual information, which can be useful for resolving ambiguities and for overcoming some visual limitations. At the end of this chapter, the reader will know about, the role of an smart camera in an IVS system, how it can extract information by using computer vision algorithms, and how contextual information could improve the results.

In Chapter 3, the concepts of context, context-aware system, and semantic model (ontology) are introduced. The chapter starts by a discussions on, *what* is context? *why* is important? and, *which* elements compose it? This follows by a comparison between different ways of representing the built environment information, making emphasis in the Building Information Modelling (BIM) and the Industry Foundation Classes (IFC). This lead us to a machine-understandable representation of a built information using a semantic model, i.e., an ontology. The rest of the chapter introduces the ontology world, their formalism, their implementation, and technologies that ease their creation, extension and interaction. At the end of this chapter, the reader will know, what is contextual information in a built environment, the different ways of obtaining it, and the advantages of using an ontology model to represent it, specially the inherent semantic interoperability. As well as, what is an ontology and the mathematical formalism behind it.

Chapter 4 starts by giving and overview of the proposed semantic-based framework (named WiseNET), and presenting the different modules that compose it. Afterwards, the chapter focuses on the definition of the core module of the system, the ontology. This is done by following an ontology-development methodology, where each step is presented and explained in details. The steps goes from defining a set of questions that the ontology should be able to answer, to selecting a set of ontologies from which some knowledge could be re-used, until defining a set of concepts, constrains and inserting individuals. Furthermore, the chapter finishes by presenting a way of inserting human-skill (common sense) rules, that extends the ontology knowledge. At the end of this chapter, the reader will know, how to develop an ontology from A to Z, how the WiseNET ontology semantically links the knowledge from sensors with the environment knowledge, and how to extend the ontology knowledge by defining human-skill logic rules.

Chapter 5, presents the different procedures developed to automatically populate data into the semantic model. The chapter starts by presenting the central API, which is an intermediary software that enables the communication between the different data sources and the ontology. Followed by the static population procedure, that allows to automatically extract the pertinent environment data from an IFC file and inserts it into the ontology. The static population procedure also includes the insertion of the camera-calibration information. Finally, the dynamic population procedure is presented. This procedure consists in automatically extracting, structuring and inserting the data observed by a camera node, into the ontology. At the end of this chapter, the reader will know, how to extract data from an IFC file, how to constantly insert data into an ontology, and how to fusion the

knowledge coming from a smart camera with contextual knowledge.

In Chapter 6, the a dataset used for evaluating the proposed semantic-based system, as well as the different evaluation procedures are presented. The first part of the chapter deals with the choice of dataset, starting by presenting a state-of-the-art of multi-camera multi-space datasets, following by a detail presentation of the WiseNET dataset, which is composed of video sets, information of the environment and annotations for people detection and tracking. The second part of the chapter deals with evaluating the system. Firstly, the ontology is evaluated by checking if it is able to answered some important questions concerning, the structure of the building, the sensors deploy on it and information about the building usage. Secondly, the performance of the semantic-based in tracking people is evaluated. Furthermore, an monitor unit interfaces is presented, which agglomerates a set of services to help building managers. Finally, a comparison between different state-of-the-art people detectors is presented, as well as their influence on the semantic-based framework. At the end of this chapter, the reader will know, the dataset used for evaluating the proposed framework, as well as, the different evaluation procedures, the results obtain by the framework and some of its limitations.

Finally, conclusions, limitations and future research are presented in Chapter 7, where we summarized our contributions, their utility in a smart building and the possible use in different scenarios.

2

VISUAL SENSOR NETWORK

> The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.
>
> *Mark Weiser, 1991*

2.1/ FROM VSN TO IVS

Currently, we are living in a world of data, where smart and connected objects are almost constantly acquiring and analyzing information about our daily life, without we even notice it! This goes from data taken by our smart phones, for example our current location and how many steps we did in a day; to data taken by the different types of sensors deployed in the environments we visit, for example occupancy sensor, thermal sensors and visual sensors.

In this study, we focus our interest in the visual sensors due to the richness of the information that can be extracted from them. For instance, from a simple image it can be extracted information about the environment, the objects present on it, their characteristics and relations between the different objects. To exemplify this, consider the image shown in Fig. 2.1, some information that can be extracted from the image is: that the image was taken in an indoor environment; that there are three doors and one person; that the doors are closed; that the person is standing and is dressed in blue; that there is a person which is around a closed door (relation between objects), etc.

The most widespread and well-known use case of visual sensors is in video surveillance systems [181], which (normally) is composed of a network of sensors, better known as Visual Sensor Network (VSN). In most current VSN systems, the captured videos are analyzed in real time by a human operator in a monitor room using dozens of screens [37] (see Fig. 2.2 for an example). This type of monitoring is not a trivial task, especially as the size of the network increases the richness of visual information leads to a high *cognitive load*[1], making it difficult, or even impossible, for a human being to monitor all the video streams at the same time and to extract useful information from them.

Many efforts have being devoted to ease the monitoring of a VSN system. One solution is to pre-analyze the generated videos thus only monitoring the pertinent data, this type of

[1] In cognitive psychology, *cognitive load* refers to the total amount of mental activity imposed on the working memory at any instant [165].

Figure 2.1 – A picture is worth thousand information. Many information can be obtained from this image, for example, that the scene happens in an indoor environment, that there are some doors in the space which are closed, that there is a single person with a blue shirt which seems happy (by looking its facial expression), that the person is around a closed door, etc.

Figure 2.2 – Classical monitoring room.

systems are called Intelligent Video Surveillance (IVS) and will be our main research application [86]. According to Wang [174], the goal of an IVS system is to " efficiently extract useful information from a huge amount of videos collected by surveillance cameras by automatically detecting, tracking and recognizing objects of interest, and understanding and analyzing their activities". From this statement we conclude that IVS is a multidisciplinary field that relates computer vision, embedded computing and image sensors. This chapter presents some works that allows to achieve the IVS vision; more precisely, the next sections will introduce the smart cameras which are image sensors with embedded computing capabilities, followed by a state-of-the-art of the computer vision algorithms required in a surveillance application, such as person detection and tracking.

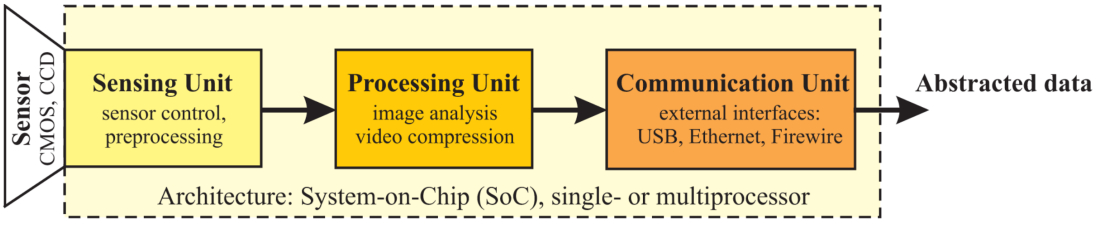


Figure 2.3 – Generic architecture of a smart camera. Figure taken from [145].

2.2/ SMART CAMERAS

The most prominent way to efficiently filtering/selecting pertinent data from the huge amount of visual information, is to rely on Smart Cameras (SCs) to perform visual tasks semi-autonomously (with minimal human interaction) [67, 18, 145, 174]. Smart cameras are sensors which are capable of acquiring visual information and filtering/extracting the pertinent information of the scene. Thus, allowing to better exploit visual data and to optimize/minimize data storage and transmission. Therefore, the SCs *not only* contain the image sensor, but also a processing unit and communication interfaces, i.e., SCs are self-contained vision systems. Figure 2.3 depicts the smart camera pipeline presented by [145]. The image sensor, which is commonly implemented either in Complementary Metal–Oxide–Semiconductor (CMOS) or Charge-Coupled Device (CCD) technology, is the one that captures the raw data in the form of an image. The sensing unit reads the raw image and performs some preprocessing to enhance it, like color transformation or white balance. The processing unit receives the image, performs some computer vision algorithm to analyze it, and then transfers the abstracted data to the communication unit. This unit enables communication with the external world by providing various interfaces such as the standard USB and Ethernet, or even wireless ones like Wi-Fi or Bluetooth.

The processing unit is the central part of a SC. Thus, the type of processing unit will determine the SC's computing power, development time (i.e., utilization of high-level programming languages, development tools, availability of libraries, etc, that help to reduce the development time) and versatility (i.e., the ability to manage and modify the processing tasks performed). For implementing complex image analysis in real-time, the choice of an SC is firstly a choice of its processing unit and secondly a choice of the image sensor and the communication interfaces. Table 2.1 presents a comparison of different processing units, based on our experience on SC design [120, 168, 33, 151, 14]. *Multi-core processors* are generic purpose processors that can be adapted in embedded systems. Currently, Multi-core enables a quick and efficient implementation of image processing algorithms, due to their architecture and the vast availability of development tools and libraries. Therefore, the trade-off between computing performance and development time is significant. Moreover, its versatility is optimal—compared to the other processing target—thanks to its generic architecture. Graphics Processing Unit (GPU) processors, which originally were designed and specialized for graphical tasks, are recently used for embedded image analysis. GPUs are particularly efficient in the implementation of Convolutional Neural Networks (CNNs)—used for object detection—due to its highly parallel structure. However, the achievement of high performance might involve the optimization/-parallelization of algorithms, which has an impact on the development time and versatility.

Table 2.1 – Comparison of processing units that can be found in smart cameras. The best value of each characteristic is shown in bold.

	Computing power	Development time	Versatility
Multi-core	medium	**low**	**very high**
GPU	high	medium	high
FPGA	**very high**	very high	low
SoC	**very high**	high	medium

The Field-Programmable Gate Array (FPGA) is equivalent to very large number of logical gates associated to a large set of arithmetic operators (i.e., up to several thousand of multipliers) and embedded memory. These resources and their reconfigurable architecture enable high processing performance to be reached using the intrinsic parallelism of the applications. Nevertheless, even if some High Level Synthesis (HLS) strategy can be used to decrease the development time [167], it is still significantly higher compared to the Multi-core and GPU. A System on Chip (SoC) regroups basically a FPGA with others hardware cores (e.g., processors, interface controllers and GPUs). The SoC's computing power is very high. Moreover, its embedded processor reduces the development time and increases its versatility, compared to FPGA.

Based on our experience and on the video surveillance application—where multiple SCs need to be deployed and multiple algorithms needs to be implemented on them (e.g., person detection, face detection and motion detection)—we favoured SCs with: (1) high–computing power, to enable real-time analysis; (2) low-development time, to facilitate the implementation of algorithms; (3) and high-versatility, to allows the management of different processing tasks. Leaving us with the choice of either Multi-core- or GPU- based SCs.

The output of a SC is some abstracted data of the observed scene (see Fig. 2.3). The delivered abstraction depends on the performed computer vision algorithm, which itself depends on the specific application of the SC. Computer vision is the science that deals with how computers/software systems can recognize and understand scenes, in other words, computer vision seeks to automate tasks that the human vision does. Some examples of computer vision algorithms are motion detection, face detection, 3D reconstruction, contour detection, image segmentation, etc. [17, 30]. However, since the most important task in video surveillance is the tracking of humans through a network of cameras then, our study will be focused on the *people detection* and *people tracking* algorithms.

Moreover, by being able to perform onboard image analysis and hence to avoid transferring raw data (videos), SCs have a great potential for increasing privacy, security and reducing the required network bandwidth.

2.3/ COMPUTER VISION - PEOPLE DETECTION AND TRACKING

The ultimate goal of computer vision is to allow computers (or SCs in our case) to emulate human vision by performing tasks like learning, making inferences and performing actions based on those inferences [17, 30]. Some typical computer vision tasks are:

- Image enhancement, which consists in processing an image to be more suitable for

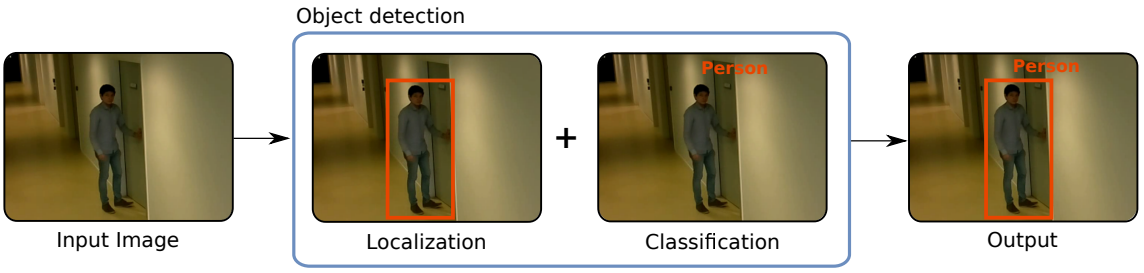


Figure 2.4 – Object detection overview, example for people detection. The output of the detector is a bounding-box containing the object and a label stating its class.

a further analysis. For example, remove noise from an image, sharpen it, blur it, brighten it, detect edges, etc.

- Scene reconstruction, which consists on generating a 2D/3D model of a scene, from a given set of images. The model could be a simple set of points or a surface model. Some examples are stitching 2D images to create a panoramic view and the structure from motion algorithm which computes a 3D model of a scene from a sequence of 2D images.

- Image recognition, which consists in classifying, identifying and detecting objects or actions in an image. For example, identifying a person's face, identifying handwritten digits, detecting cats or people in an image, detecting human activities such as dancing or playing soccer, etc.

We will focus our study in the image recognition aspect, specifically in the different ways to detect people in an image and track them while they appear in different cameras.

2.3.1/ OBJECT DETECTION → PEOPLE DETECTION

Object detection consists in identifying the location of an object in an image—localization—and predicting its class (i.e., type of object)—classification. Thus, the output of an object detector are: (1) the rectangle containing the object, called bounding box (Bbox), and (2) its class name. Figure 2.4 illustrates an example where a person-object is being detected.

People detection (also known as pedestrian detection) is a particular case of object detection, which has interest researchers over time, due to its challenges and its many applications including robotics, surveillance, entertainment, and driverless cars [48, 60, 152].

People detection methods can be grouped into two categories according to their nature: (1) feature-descriptor based, or (2) deep-learning based. We will now review some works on both categories. Please note that it is outside the scope of this manuscript to give precise details on the presented methods, and we refer interested readers to their papers.

Feature-descriptor based methods A feature descriptor is a representation of an image (or image patch) that simplifies it by extracting useful information and removing unnecessary information (i.e., selecting), according to a hand-crafted model. The goal of

a feature descriptor is to generalize a class of objects in such a way that different objects of the same class will produce similar feature descriptors, like this simplifying the classification task.

Generally, all the methods using feature descriptors follow a sliding-window paradigm in order to detect people at different scales [48, 10]. This paradigm entails acquiring image patches from the complete image, passing each patch through the person detection pipeline (see Fig. 2.5, which will be explained afterwards) and finally applying a Non-Maximum Suppression (NMS) algorithm to eliminate multiple nearby detections. The image patches are obtained by sliding a fixed size window from left to right, and from up to down and cutting out the corresponding patch in the image. To achieve multi-scale people detection, the image should be iteratively resized and each image size should be process using the same fixed sized window [128].

A classical person detector pipeline based on feature descriptor is shown in Figure 2.5. The input of the pipeline is an image patch, obtained from sliding window in the image. This image patch is then preprocessed to adjust it for the feature descriptor extraction. Some examples of preprocessing algorithms are image resizing, filtering (e.g., sharpening, blurring, reducing noise), color normalization (i.e., reduce color variations), color transformation (e.g., passing from Red-Green-Blue (RGB) to grayscale), etc.

After preprocessing, the image patch is passed through a feature descriptor model that extracts and selects the useful features from it. Some examples of feature descriptors are Histogram of Oriented Gradients (HOG) [42], Scale-Invariant Feature Transform (SIFT) [105], haar-like features [128, 171], Local Binary Pattern (LBP) [126] and color histograms like RGB or HSV [179, 130]. The color features can be obtain in different spaces, such as: RGB color space which stores individual values for red, green and blue, and describes what kind of light needs to be added/subtracted to produce a given color; HSV color space which stores values of hue, saturation and value/brightness, thus allowing to remove the influence of light changes by only considering the HS channels; CIELAB which is a perceptually linear space (i.e., a change in a color value should produce a visual change of the same importance) expresses color as three values: L^* for the lightness that goes from black to white, a^* that goes from green to red, and b^* that goes from blue to yellow; and the YCbCr space which is widely is in video and image compression, where Y is the light component and Cb and Cr are the blue-difference and red-difference chroma components.

Afterwards, the resulting features are passed through a binary classifier, that will determine the presence or not of a person in the input data. The most popular choices of classifiers, due to their theoretical guarantees, extensibility, and good performance, are Support Vector Machines (SVMs) [39, 23], Adaptive Boosting (AdaBoost) [59] and decision forest [138]. Finally, if a person is detected, then the coordinates of the Bbox will correspond to the sliding window used to extract the image patch (the input of the pipeline).

Papageorgiou and Poggio [128] were one of the firsts to propose a sliding window detector using haar-like features and SVM for classification. Viola et al. [172] based on those ideas, introduced the integral images for fast feature computation and used a cascade of AdaBoost classifiers for feature selection and classification. Inspired by shape context and SIFT features, Dalal and Trigss [42] introduced the popular HOG features and test them using a SVM classifier. Since their introduction, the HOG features have inspired many researchers. In 2014, Benenson et al. [10] reviewed 44 people-detectors,

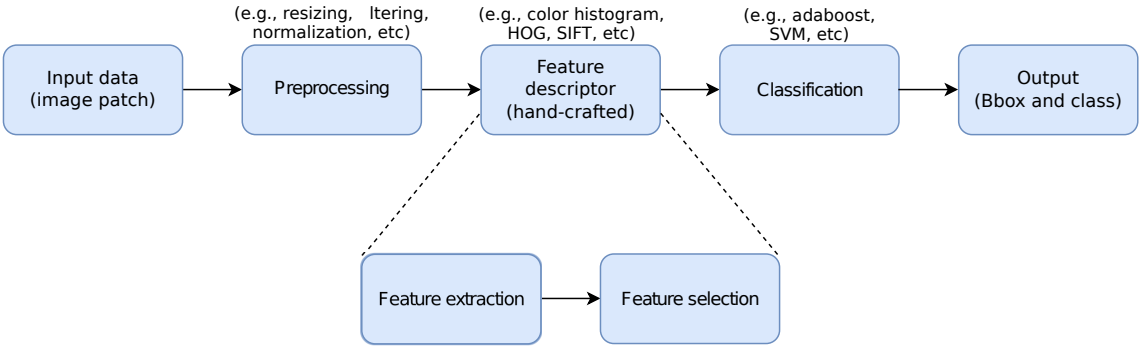


Figure 2.5 – Pipeline of a person detector based on feature descriptors.

from which 40 used HOG plus other type of features; for example, Wang et al. [175] combined HOG and LBP features to deal with partial occlusions, and Bar-Hillel et al. [8] combined HOG, SIFT and other features in a part-based method using an SVM classifier. Felzenszwalb et al. [56] proposed a Deformable Part Model (DPM) that handled view and pose variations; their model combined HOG features plus a set of part filters and deformation models, and used a modified version of SVM called Latent-SVM. Dollar et al. [47, 46] proposed the Integral Channel Features (ICF) and the Aggregated Channel Features (ACF) detectors, which combined HOG and color channels features, selected by a boosted decision forest; they also proposed to increased the computational efficiency by approximating features at multiple scales using a sparsely sampled image pyramids. The ICF/ACF methods have inspired many detectors as presented by Zhang et al. [189]; one method worth to mention due to its high performance, is the Checkboard detector presented by Zhang et al. [190] which used HOG, motion and color features with a set of filters with “checker-board like” patterns and a boosted decision forest.

Even-though, some feature-descriptor based methods have achieved competitive results at low computational complexity (as presented in [189]), they still present many disadvantages, such as:

- **Hand-crafted features** The main focus of the methods is on designing features. The detectors accuracy is largely determined by the ability of the designer to come up with and appropriate set of features [93]. Moreover, these features are mainly designed based on the people shape, which limits their discriminative ability against similar-looking background objects.
- **Low-level features** The basis of the descriptors are low-level features (local) such as edges, colors and textures, which lack of higher level of representation.
- **Sliding window paradigm** The sliding window paradigm is a computational bottleneck for most of the detectors [46].
- **Disjointness between features and classifier** One main problem with these methods is that the classification part has no influence on the feature descriptor, in contrast to the methods based on Convolutional Neural Networks (CNNs). Meaning that the feature representations and the classifiers cannot be jointly optimized to improve performance.

Remark. *For additional details on the detectors, their evaluations and comparisons we refer the reader to the original publications and to the following surveys: [48, 10, 189].*

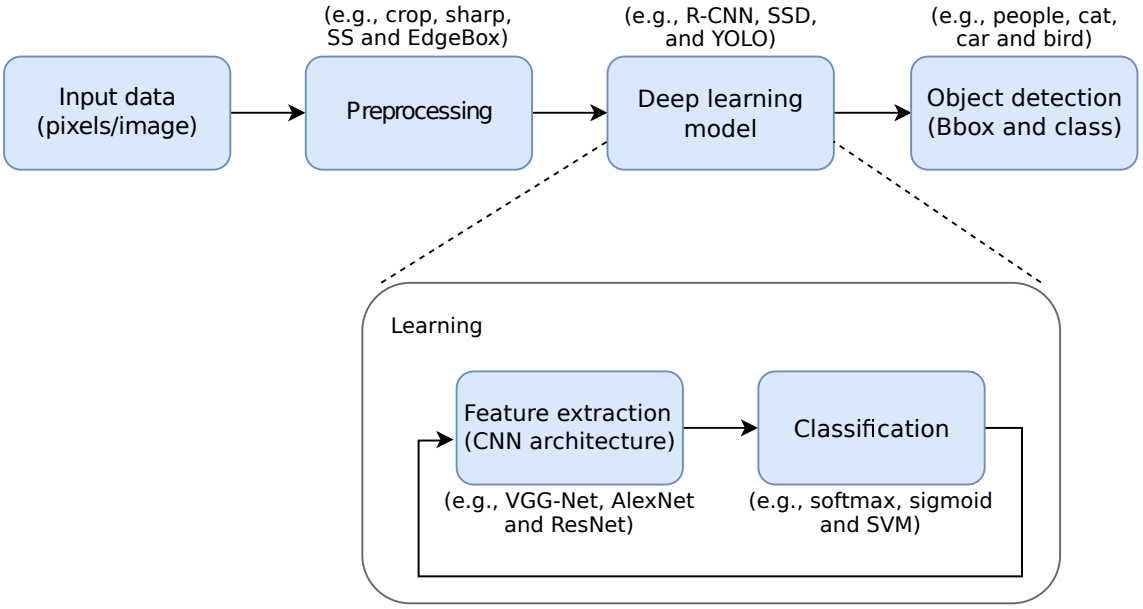


Figure 2.6 – Pipeline of an object detector based on a deep learning model.

Deep-learning based methods Deep learning is a class of machine learning technique that teaches computers to learn by examples. CNN is a class of deep learning commonly applied to recognize visual patterns directly from pixel/images with minimal preprocessing. CNNs learn higher-order features of an image via performing a sequence of convolution, which are trained using the back-propagation algorithm [93, 89, 97, 104]. Some advantages of CNNs are that they can discover features from raw pixel values i.e., no need of human effort in feature design; that they can recognize patterns (i.e., objects) with extreme variability and; that the classification step has an influence in the feature extraction–due to the back-propagation. Another important feature of CNN-based object detectors is that they are capable of detecting multiple classes of objects at once. Thus, state-of-the-art person detector can also accurately detect cats, dogs, chairs, cars, etc.

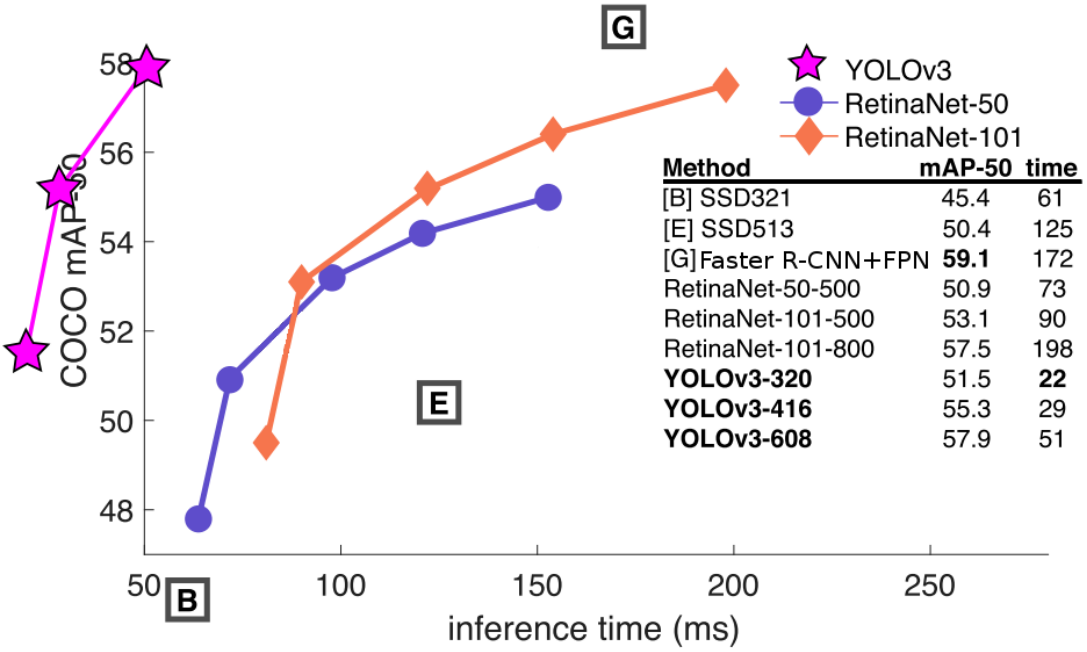
The rapid adoption of CNN in the last few years—and more generally of deep learning—has brought into development highly-accurate deep-learning based object detectors such as: Regions with CNN features (R-CNN) [64]; Fast R-CNN [63]: Faster R-CNN [143]; Single Shot Detector (SSD) [103]; RetinaNet [99]; and You-Only-Look-Once (YOLO) [142]. Figure 2.6 presents a general pipeline of this type of object detectors. These detectors use as feature extractors some of the most well-known CNN architectures (also known as backbone network) such as: AlexNet [89], VGG-Net [153], ResNet [69] and Darknet-59 [141]. The CNN architectures differ by their layers configurations (i.e., their design), for example they present different number of layers, type and sequence of layers. The deep-learning models may also use some well-known classifiers like SVM [39, 23], sigmoid binary-classifier [68] and its multi-class generalization called softmax [147]; as well as some region proposal algorithms like Selective Search (SS) [170], EdgeBox [196] and Region Proposal Network (RPN) [143].

Deep-learning based object detectors can be divided into two classes according to the number of stages. Two-stage detectors (also known as region-based), such as R-CNN, Fast R-CNN and Faster R-CNN, perform a strategy that (mainly) consists of: (1) propose some category-independent regions of interest, in order to reduce the number of

image patches feed to the CNN, and (2) classify the proposed regions. The R-CNN detector [64], uses the SS method [170] to propose some regions of interest, afterwards each proposed region is preprocessed and passed through the AlexNet architecture [89], finally the extracted features are passed through a set of SVMs to predict the class of the image patch. The Fast R-CNN detector [63], improved the speed and accuracy of R-CNN, by passing the whole image through the CNN architecture, and then selecting the proposed regions; Fast R-CNN used the VGG-Net that is much deeper than AlexNet (i.e., it has more layers), and SVM and softmax classifier. The Faster R-CNN detector [143], adopted similar design as the Fast R-CNN except, it improved its speed by implementing an internal CNN-based region proposal approach called RPN, which replaced the existing "slow" region proposal methods (e.g., SS, EdgeBox [196]). Faster R-CNN performs around 10 times faster than Fast R-CNN, while Fast R-CNN performs around 25 times faster than R-CNN [63, 143].

One-stage detectors (also known as single-shot), such as SSD, RetinaNet and YOLO, perform a detection strategy that consist of generating, in single step, the Bboxes and the classes, i.e., they do not have a separate step for region proposal as two-stage detectors. The SSD [103], uses the VGG-Net to extract the feature maps of the complete image, then applies a sequence of multi-scale convolutional layers and anchor boxes (as introduced in Faster R-CNN) and finally uses softmax classifier. The RetinaNet detector [99], proposes to increase the accuracy of one-stage detectors by reducing the foreground-background class imbalance during training. To achieve this, they introduced a new *focal loss* function that prevents the vast number of easy negatives from overwhelming the detector during training. The RetinaNet adopted the ResNET-FPN architecture [98] with sigmoid classifier. The YOLO detector [142], divides the image into regions and predicts Bboxes and probabilities for each region, moreover, the Bboxes are weighted by the predicted probabilities. The current version, YOLOv3, uses the DarkNet-53 architecture, for feature extractor, plus a multi-scale prediction method based on *Feature Pyramid Networks* (FPN) [98], that allows it to better detect small objects.

Generally, people detectors are evaluated by their precision and processing time. In the Pascal VOC Challenge [52], it was proposed to evaluate the precision by computing the Average Precision (AP), using a fix Intersection Over Union (IOU) threshold of 50% and the model confidence to compute an accumulated Precision and Recall curve (PR-curve). The IOU is used to evaluate the ability to “localize” and the confidence the ability to “classify”. Firstly all the detections are compared to the ground truth detection to determine their degree of overlapping, if the overlapping is greater (or equal) than the IOU threshold then the detection is considered as a True Positive (TP) (correct detection), if not, is considered as a False Positive (FP). Afterwards, the accumulated precision and recall is computed and plotted (PR-curve) starting from the most confident detection until the less confident one. Precision is the fraction of detected instances that are relevant (see Eq. 2.1), while recall is the fraction of relevant instances that are detected, if a detection is not detected then is consider as a False Negative (FN) (see Eq. 2.2). Finally, after obtaining the PR-curve, the AP is computed as the area under the curve (AUC). A common method, is to interpolate the PR-curve based on 11 recall points and then computed the AUC, from the interpolated curve, as an average of the their precision values. AP is a numerical metric, which simplifies the comparison of different detectors. Moreover, if there are multiple types of objects to be detected, then the AP for each type should be computed, and finally all those APs should be average to obtain the Mean Average



Method	mAP-50	time
[B] SSD321	45.4	61
[E] SSD513	50.4	125
[G] Faster R-CNN+FPN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

Figure 2.7 – Inference time (ms) versus mean average-precision (mAP) of one- and two-stage detectors. The figure was taken and adapted from [142].

Precision (mAP) metric.

$$Precision = \frac{TP}{TP + FP} \tag{2.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.2}$$

The processing time or inference time, is the time required by the detector to output the results (detections), and is normally measured in milliseconds (ms). The processing speed is very important, *not only* for real-time applications, but also for applications that required the use of systems with limited resources, such as smart cameras. In previous years, one-stage were considered as trading accuracy for real-time processing speed, thus having a lower AP compared to two-stage detectors [99]. However, as shown in Table 2.2 and Fig. 2.7, the recent developments of one-stage detectors have reached the same (or better) level of accuracy than state-of-the-art two-stage methods, while still being much faster. For example, the YOLOv3 detector is by far the fastest (see Fig. 2.7) object detector of all, and his accuracy (in most of the cases) is around 5% lower than the highest accuracy achieved by Faster R-CNN with FPN the state-of-the-art two-stage detector (see Table 2.2). The evaluations shown in Table 2.2 and Fig. 2.7 were performed on the COCO test-dev benchmark [100] using different variants of: SSD with different image scales (321 and 513) [103]; RetinaNet with different depths (50 and 101) and image scales (500 and 800) [99]; YOLOv3 with different image scales (320, 416 and 608) [142]; and a combination of Faster R-CNN with FPN [98]. Moreover, the accuracy shown is not only for the person class, but for the 80 object types considered in the COCO dataset (person, cat, banana, etc).

Some of the main disadvantages of deep-learning object detectors are: their high demand of processing resources (multiple GPU cores), the need of large amount of data

Table 2.2 – Average-precision (AP) of one- and two-stage detectors at different IOU thresholds and across different object scales (small (S), medium (M) and large (L)). Table taken and adapted from [142].

	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage method						
Faster R-CNN with FPN	36.2	**59.1**	39.0	18.2	39.0	48.2
One-stage methods						
SSD$_{513}$	31.2	50.4	33.3	10.2	34.5	49.8
RetinaNet	**39.1**	**59.1**	**42.3**	**21.8**	**42.7**	**50.2**
YOLOv3-$_{608}$	33.0	57.9	34.4	18.3	35.4	41.9

to train the models and their complex configuration (high number of parameters to tune). However, those disadvantages are relevant only if the model should be train from *scratch* using a new (and large) data set. Moreover, most of the state-of-the-art deep-learning models are shared already pre-trained, and they can be used *off-the-shelf* (i.e., they can be applied to new data directly without any modification). Still, the result of the pre-trained model in the new data might not be acceptable. In this case, a common solution is to apply *fine-tuning* techniques (a type of transfer learning [187]) to re-train only selected model layers using new data. Model fine-tuning requires much fewer resources and data than training from scratch.

Based on the current results of state-of-the-art people detectors, it can be concluded that this task is (almost) solved, however the detection accuracy depends on the resources available. Feature-descriptor based detectors tend to required lower processing resources compared to deep-learning based detectors, however their accuracy tend to be lower as well. Furthermore, the accuracy of deep-learning based detectors also depends on the amount of data used to train them.

Remark. *For additional details on the detectors, their evaluations and comparisons we refer the reader to the original publications.*

2.3.2/ PEOPLE TRACKING

The impressive progress in people detectors has led to great advancements in people tracking techniques [166]. People tracking is an important task in computer vision, especially for applications such as robotics and intelligent video surveillance. It consists of assigning an Unique Identifier (ID) to each people and maintaining it through time, while they move within single or multiple cameras [90]. Hence, multi-camera people tracking consists of two crucial functional modules: (1) *intra-camera tracking*, i.e., tracking people within a single camera view; and (2) *inter-camera tracking*, i.e., associating the people's tracks observed by different cameras, to maintain the identities of people when they move from one camera to another [90, 174]. Moreover, inter-camera tracking can be done between cameras which have *overlapping* fields of view (i.e., cameras that have a common view of the scene), or between *non-overlapping*.

Figure 2.8 depicts the relation between people detection and tracking modules. Abstractly, a people tracker takes as inputs a set of detections computed by a people detector algorithm (see Section 2.3.1). Afterwards, for each camera, the tracker should partition the detections into sets, where each set corresponds to one person—more precisely a

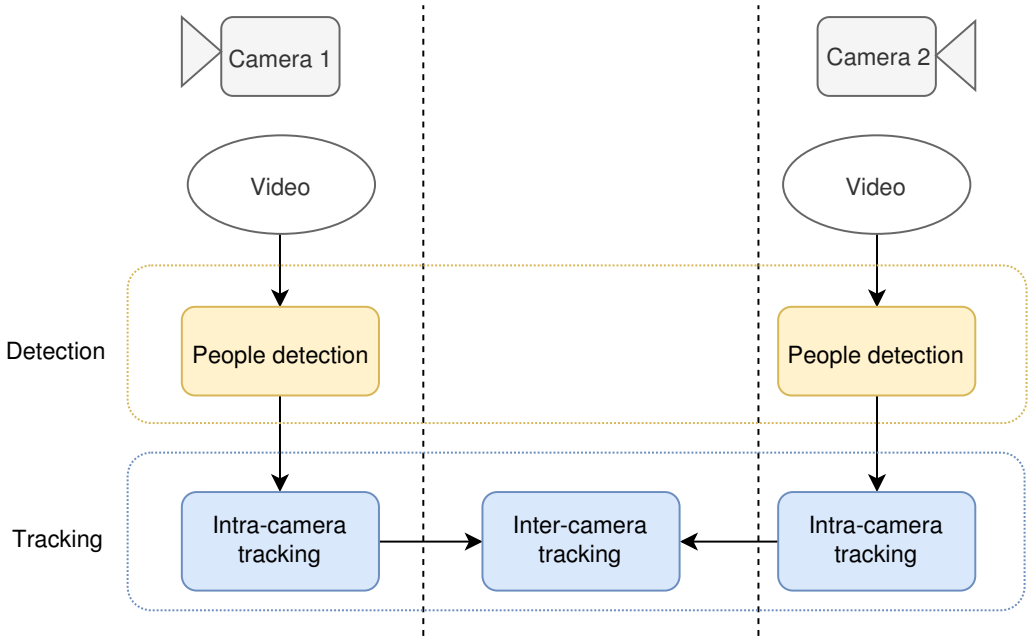


Figure 2.8 – Relation between people detection and tracking functional modules.

person's ID. The set of detections coming from a single cameras and order by time is known as *tracklet* [166], and is the output of the intra-camera module. Finally, the people tracker should associate the tracklets, coming from multiple cameras, that correspond to the same person. The set of a tracklets corresponding to a precise person's ID and ordered by time is known as a *trajectory*, and is the output of the inter-camera module.

People tracking is a challenging task, since different people may look alike, making it difficult to discriminate between them. Moreover, the appearance of a single person may also vary due to changes in illumination, camera viewpoints, pose (e.g., standing and sitting), partial or total occlusions, and background clutter [79]. In order to deal with these challenges, numerous researchers have proposed different approaches, as presented in the surveys of Hou et al. [79] Smeulders et al. [157] and Luo et al. [106]. In general, multi-camera tracking methods differ according to their assumptions and the type of information considered to tackle the tracking challenges. Hou et al. [79], categorized the tracking methods in: (1) generative trackers, (2) discriminative trackers, (3) Camera-Link Module (CLM)-based, (4) Graph Model (GM)-based, and (5) human re-identification (human re-id). The first two categories can be used only for intra-camera tracking, while the rest can be also be used for inter-camera tracking with or without overlapping.

Generative trackers estimate each people's location and correspondence, based on their locations and movements in previous frames. To reduce exhaustive search, the trackers use methods like kalman filter [177, 94], particle filter [185, 115] and kernel-based tracking [35, 80]. In contrast to generative trackers, *discriminative trackers* obtain all the people locations in each video frame (or a temporal window) and then jointly establish the people correspondence that optimize all the tracklets. The tracklets are optimized by using target association techniques such as joint probability data association filtering [124, 144], multiple-hypothesis tracking [197, 85] and flow network framework [24, 129]. The main drawback of generative and discriminative trackers is their strong dependence on target location and movement factors, which makes them usable only for intra-camera tracking.

The *CLM-based trackers* focus on establishing the link models (correlations) between the cameras in a network, to estimate feature correspondence relationships between adjacent cameras (i.e., space-time and appearance relationships). For example, some CLM trackers use the topology of the camera networks—more precisely the connectivity between entry zones and exit zones– to reduce mismatch across cameras [108, 26]; others compensate the illumination variations between cameras by estimating a Brightness Transfer Function (BTF) [84, 137]; and others combined the space-time information—obtained from the walking transition time between cameras—and BTF to obtained a more robust results [90, 34]. CLM-based trackers can be used for inter-camera tracking however, their main drawback is the requirement of a large amount of training data (manually or automatically labelled) to stablish correspondences, which limits the scalability to realistic applications.

The *GM-based trackers* use graph modelling techniques to deal with data associations across multiple cameras, using as inputs people's detections, tracklets and trajectories. The GM is composed of nodes, edges ad weights and is solved using the Maximum A Posteriori (MAP) optimization solution framework. Some examples of this type of tracker can be found at [83, 31, 32]. GM-based trackers can effectively track people in complex scenes, such as crowd and interference of human appearance. However, it is difficult to get optimal solutions from the data association, moreover, they require all observations in advance to create the GM.

Human re-id techniques, also known as *person re-id*, consists on matching two images/detections of the same person, under intensive appearance changes, based on the similarity of their visual features [195]. Compared to the other tracking methods where several cues can be applied (e.g., spatio-temporal information), person re-id methods typically use the appearance as the only cue (type of feature). Thus, person re-id trackers focus on extracting discriminative and robust visual features to characterize human appearance and shape. We are particularly interested in this type of techniques due to they can be used for both intra- and inter- camera tracking, they have good scalability and they can be used online (i.e., they just need the current and past information). Person re-id approaches differ by three main aspects:

1. **Feature descriptor:** there are a lot of feature descriptors which can be used in person re-id, such as color, texture, shape, regional features, patch-based features, etc [107]. In general, compared to other features, color feature is dominant under slight lighting changes since it is robust to changes in viewpoint [186, 79]. Thus, it is widely used in appearance matching for inter-camera tracking since the color of clothing provides information about the identity of the individual [107]. In the other hand, texture and shape features are stable under significant lighting changes, but they are subject to changes in viewpoint and occlusion.

   However, most visual features are either insufficiently discriminative for cross-view matching or insufficiently robust to viewpoint changes, thus a combination of features is required. Some examples of combined features are: the Ensemble of Localized Features (ELF) [65], which uses a combination of color histograms in the RGB, HS and YCbCr color spaces, as well as texture histograms of Gabor and Schmid filters; the dColorSIFT descriptor [192], which densely divided the image into patches, and for each patch they extract and concatenate CIELAB color histograms—to handle color variations—as well SIFT descriptors—to handle viewpoint and illumination changes; the Local Maximal Occurance (LOMO) [95], that uses HSV histograms

combined with scale-invariant LBP; the WHOS descriptor [101], which uses histograms in HS and RGB, combined with HOG texture features; and the Gaussian Of Gaussian (GOG) [114], that uses a combination of color histograms in HSV, RGB and Lab color spaces, as well as gradient features.

Furthermore, many studies have showed that color histograms are the most important features for person re-id [61, 65, 91, 186, 101, 114]. Specifically, the HS color channels are considered as the most discriminative cues given the illumination changes between different cameras.

2. **Distance metric:** the distance metric determines the correspondence/similarity between two feature descriptors. There are two types of metrics, the *standard unsupervised metrics* which directly compare two feature descriptors, and the *supervised metric learning* which maximize the human matching accuracy and improves person re-id performance. Cha [28] presented a survey on standard distance metrics, for histogram-based features, from which the most commonly used are bhattacharya distance, used in [65, 54]; the cosine distance, used in [95]; and the euclidean distance, used in [194, 114]. In the other hand, metric learning shifts the focus from capturing feature descriptors to learning a new feature space such that feature descriptors of the same person are close whereas those of different people are relatively far [160]. Srikrishna et al. [160] presented evaluations using different metric learning methods, from which the best performance were obtained using the Keep-It-Simple-and-Straightforward (KISSME) [88], the Cross-View Quadratic Discriminant Analysis (XQDA) [95] and the Discriminative Null Space Learning (NFST) [188].

   Even thought metric learning improves person re-id accuracy, it requires pairwise supervised labeling of training datasets, which is a severe limitation in real application scenarios where a priori labeled data is not available [101].

3. **Improvement:** there are many types of enhancements that might be employed to improve the correct matching between detections, the most common are:

   - *Background suppression*, which consist on reducing the inclusion of non-informative information in the feature descriptor. Farenza et al. [54], proposed to separate the foreground (the person) from background by segmentation, and then to extract features only from some localized regions in the foreground image. Similarly, Kuo et Nevatia [91], proposed to extract features directly from a set of localized regions. Zheng et al. [194] and Matsukawa et al. [114], proposed a simple solution by exerting a 2-D Gaussian kernel in order to reduce the influence of the pixels which are farther from the center. In a similar way, Lisanti et al. [101] propose an Epanechnikov kernel to diminish the influence of background information near the image boundary.
   - *Stripping*, which consist on dividing the detection image into a set of horizontal stripes and extracting the feature descriptors in each stripe. As shown in the literature, stripping makes the descriptors more robust to viewpoint variations, however, it may also lose spatial details within a stripe, thus affecting its discriminative power [95]. Some examples of stripping can be found in the following works [102, 95, 101, 194, 114].
   - *Normalization*, which consists in keeping the relative contribution of the each feature regardless of their absolute contribution. Normalization should be specially performed while working with color histograms to deal with images of

different scales. There are two typical histogram normalization methods that can be found in the literature, the l^1-norm and the l^2-norm. Some works using l^1-norm can be found at [127, 184], while some works using the l^2-norm can be found at [56, 149, 176, 114]. The choice of normalization method may be based on the type feature descriptor and distance metric used, however, Sanchez et al. [149] presented some advantages of using l^2-norm over l^1-norm. Moreover, according to Weiming et al. [176], the l^2-norm is widely used in the computer vision community.

A systematic evaluation of person re-id methods combining different feature descriptor with multiple distance metrics, can be found at [160].

The current results of state-of-the-art tracking methods are quite promising but not perfect [146]. The current tracking methods have a high dependence on the detection results, which themselves depend on the resources available (as stated in the previous section). Moreover, in real applications the detectors might have some problems such as false and miss detections, or sensor failures, which might lead to scenarios where the tracking methods cannot be applied. Furthermore, current tracking methods do not consider the semantic contextual information of the environment, which can help to improve the tracking.

For contextual information, we do not mean the network topology as consider in the CLM-based tracking methods; but we mean the complete information of the environment, its topology (i.e., connection between spaces), its functionality (e.g., the capacity of each space and if a space is restricted or not), the relation of the environment with the camera network (e.g., which doors are observed by each camera and where does those doors lead to) and human-skill knowledge of the environment. For example, logic knowledge that "*if a person has not leave the room, then the person is still in the room*" or that "*if two cameras observe, two different people, around the same door at the same time, then those people are the same*". **The consideration of semantic contextual information can facilitate more robust and accurate people tracking among different camera views and enhance the analysis of difficult situations.**

To exemplify the impact of contextual information, let us consider the example of Fig 5.11 were a person is walking in a building while being detected by two smart cameras that perform a high-accurate people detection algorithm. The example was analyzed using three (hypothetic) tracking methods: (1) a dummy *single-camera* method which does not consider any relation between cameras (i.e., only intra-camera), (2) a *multi-camera* methods which considers the relation between cameras (intra- and inter-camera), and is based on person re-id tracking method, i.e., considers visual similarity between people, and (3) a *multi-camera+context* method which considers the relation between cameras and some contextual information. Based on the tracking method, three questions need to be answered: how many people there are in $space_1$?, how many people there are in $space_2$?, and how many people there are in the complete building?.

The results are presented in Table 2.3. At t_1 the person was detected by a single camera, thus all methods responded correctly. At t_2 the person was detected by both cameras. In this case, the *single-camera* method made a mistake by considers that there are two people in the building due to each camera detects a person. In the other hand, the *multi-camera* method responded correctly by comparing the similarities between the detections of both cameras and determining that they belong to the same person. However, the comparison based on visual features might not always give the correct result (e.g., due

to illumination changes between cameras). Thus, the *multi-camera + context* method used the knowledge that "*if two cameras observe, two different people, around the same door at the same time, then those people are the same*" to determine the correspondence between both person detections. At t_3 the person was again detected by a single camera, thus all methods responded correctly. At t_4 the person was not detected by any camera, due to it was in a blind region (i.e., outside the cameras' field of views). Consequently, the *single-camera* and *multi-camera* methods made a mistake, stating that there is no person in any place. However, the *multi-camera + context* method used the knowledge that "*if a person has not leave the room, then the person is still in the room*" to determine that there was a person in $space_2$, even if it was not being observed. At t_5 and t_7 the person was again detected by a single camera, thus all methods responded correctly, while at t_6 the person was detected by both cameras an the situation of t_2 was repeated.

As observed in the previous example, the contextual information helps to resolve ambiguities and uncertainties that arise due to changes in visual features, and it allows to deal with the situation when the target is totally occluded, i.e., *it allows to know what the cameras do not see*.

Remark. *For additional details on the people trackers, their evaluations and comparisons we refer the reader to the original publications.*

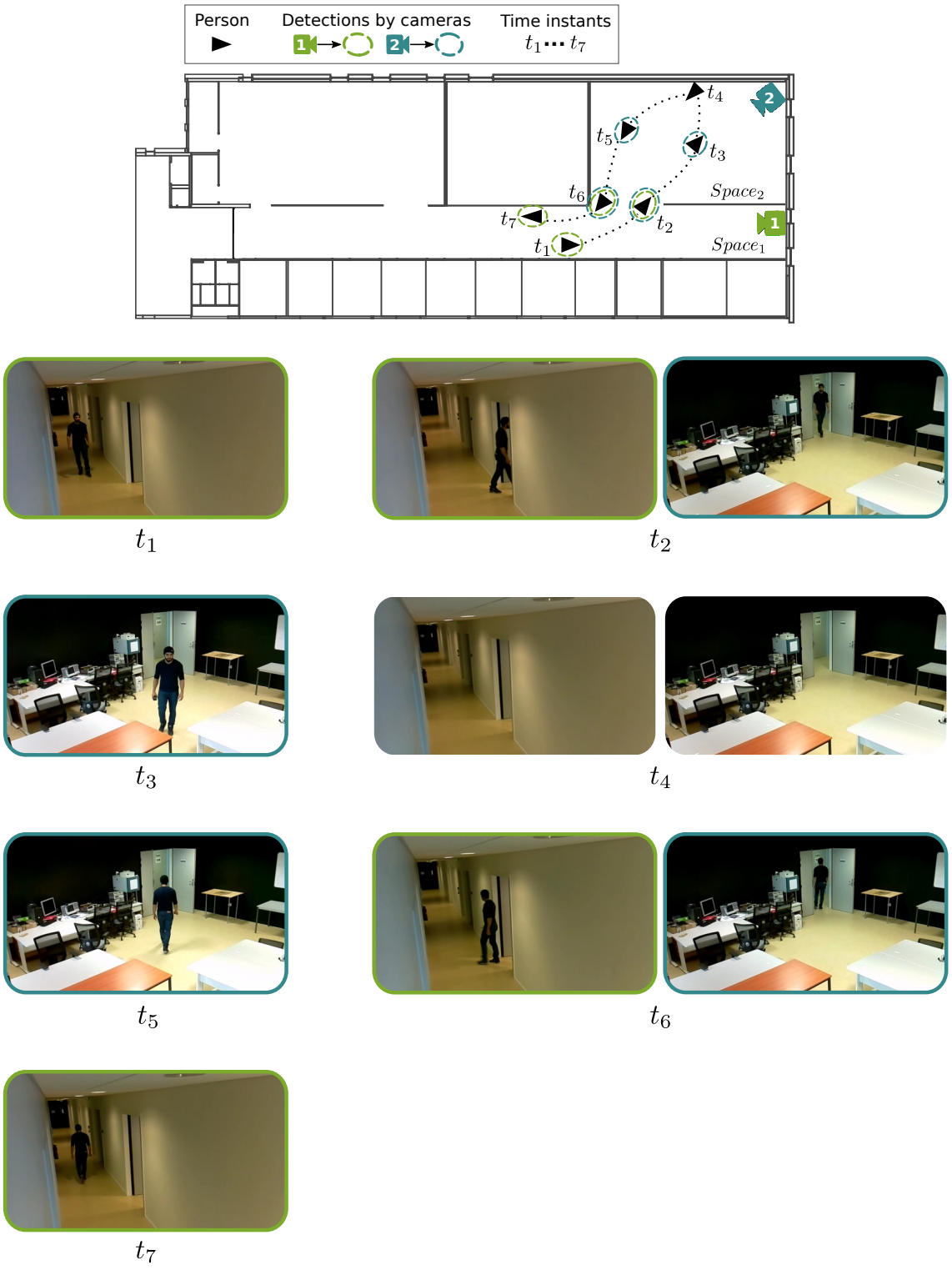


Figure 2.9 – People tracking example. One person is walking in a built environment while being detected by two cameras. The pertinent images taken by the cameras at the different time instants are shown in the lower part of the figure. Notice that at time instants t_2 and t_6 the person is being detected by both cameras, while at t_4 is not being detected by any camera.

Table 2.3 – Analysis of the example shown in Fig. 5.11 using single-camera, multi-camera and multi-camera+context methods. The analysis focus on answering the questions shown in gray, about the number of people at the different time instants ($t_1 \dots t_7$). "P_x" stands for "person detection X" and "$P_x@door$" stands for "person detection X was made around a door".

	Single-camera	**Multi-camera**	**Multi-camera+context**
t_1	$cam_1 \xrightarrow{detects} P_x$ $cam_2 \xrightarrow{detects} -$ People in $space_1$? 1 People in $space_2$? 0 People in the building? 1	$cam_1 \xrightarrow{detects} P_x$ $cam_2 \xrightarrow{detects} -$ People in $space_1$? 1 People in $space_2$? 0 People in the building? 1	$cam_1 \xrightarrow{detects} P_x$ $cam_2 \xrightarrow{detects} -$ People in $space_1$? 1 People in $space_2$? 0 People in the building? 1
t_2	$cam_1 \xrightarrow{detects} P_x$ $cam_2 \xrightarrow{detects} P_y$ People in $space_1$? 1 People in $space_2$? 1 People in the building? 2	$cam_1 \xrightarrow{detects} P_x$ $cam_2 \xrightarrow{detects} P_y$ $(P_x \xleftrightarrow{similar} P_y) \xrightarrow{then} P_x = P_y$ People in $space_1$? 1 People in $space_2$? 1 People in the building? 1	$cam_1 \xrightarrow{detects} P_x$ $cam_2 \xrightarrow{detects} P_y$ $(P_x@door \wedge P_y@door) \xrightarrow{then} P_x = P_y$ People in $space_1$? 1 People in $space_2$? 1 People in the building? 1
t_3	$cam_1 \xrightarrow{detects} -$ $cam_2 \xrightarrow{detects} P_y$ People in $space_1$? 0 People in $space_2$? 1 People in the building? 1	$cam_1 \xrightarrow{detects} -$ $cam_2 \xrightarrow{detects} P_y$ People in $space_1$? 0 People in $space_2$? 1 People in the building? 1	$cam_1 \xrightarrow{detects} -$ $cam_2 \xrightarrow{detects} P_y$ People in $space_1$? 0 People in $space_2$? 1 People in the building? 1
t_4	$cam_1 \xrightarrow{detects} -$ $cam_2 \xrightarrow{detects} -$ People in $space_1$? 0 People in $space_2$? 0 People in the building? 0	$cam_1 \xrightarrow{detects} -$ $cam_2 \xrightarrow{detects} -$ People in $space_1$? 0 People in $space_2$? 0 People in the building? 0	$cam_1 \xrightarrow{detects} -$ $cam_2 \xrightarrow{detects} -$ P_y has not left the $space_2$ People in $space_1$? 0 People in $space_2$? 1 People in the building? 1
t_5	$cam_1 \xrightarrow{detects} -$ $cam_2 \xrightarrow{detects} P_y$ People in $space_1$? 0 People in $space_2$? 1 People in the building? 1	$cam_1 \xrightarrow{detects} -$ $cam_2 \xrightarrow{detects} P_y$ People in $space_1$? 0 People in $space_2$? 1 People in the building? 1	$cam_1 \xrightarrow{detects} -$ $cam_2 \xrightarrow{detects} P_y$ People in $space_1$? 0 People in $space_2$? 1 People in the building? 1
t_6	$cam_1 \xrightarrow{detects} P_x$ $cam_2 \xrightarrow{detects} P_y$ People in $space_1$? 1 People in $space_2$? 1 People in the building? 2	$cam_1 \xrightarrow{detects} P_x$ $cam_2 \xrightarrow{detects} P_y$ $(P_x \xleftrightarrow{similar} P_y) \xrightarrow{then} P_x = P_y$ People in $space_1$? 1 People in $space_2$? 1 People in the building? 1	$cam_1 \xrightarrow{detects} P_x$ $cam_2 \xrightarrow{detects} P_y$ $(P_x@door \wedge P_y@door) \xrightarrow{then} P_x = P_y$ People in $space_1$? 1 People in $space_2$? 1 People in the building? 1
t_7	$cam_1 \xrightarrow{detects} P_x$ $cam_2 \xrightarrow{detects} -$ People in $space_1$? 1 People in $space_2$? 0 People in the building? 1	$cam_1 \xrightarrow{detects} P_x$ $cam_2 \xrightarrow{detects} -$ People in $space_1$? 1 People in $space_2$? 0 People in the building? 1	$cam_1 \xrightarrow{detects} P_x$ $cam_2 \xrightarrow{detects} -$ People in $space_1$? 1 People in $space_2$? 0 People in the building? 1

2.4/ CONCLUSION

In this chapter, we introduced the terms of Visual Sensor Networks (VSN), Intelligent Video Surveillance (IVS), Smart Cameras (SC) and computer vision. Furthermore, we presented the state-of-the arts of people detection and people tracking methods.

Nowadays, VSN have become a part of our daily life. They can be found in cities, commercial centers, supermarkets, offices, and even in houses. However, as the size of the network increases, it becomes more difficult (or even impossible) for the human operators to monitor all the video streams at the same time, due to the high cognitive load. A solution to this problem, is to implement and IVS system which pre-analyzes the video data to extract only the useful/pertinent information. The most prominent way of achieving the IVS vision is by using SC to capture high-level descriptions of a scene and analyze them in real-time by employing different computer vision algorithms.

The detection of people within an environment is a key task of most IVS systems. Thus, the performance of this task using computer vision algorithms has interest many researchers over time. Furthermore, as presented in the state-of-the-art methods, this problem is almost solved, specially by considering the results of deep-leaning based models. However, the accuracy of the detectors depends highly on the resources available —processing resources (hardware) and data for training.

Another important task of IVS systems is people tracking. This task is tightly related to the people detection, and it consists of tracking a person across a series of cameras. This is done by assigning an ID to each people and maintaining it through time while they move within cameras. The current results of state-of-the-art people tracking methods are quite promising. However, one main drawback of those methods is the disregard of semantic contextual information which can be very useful for an IVS system.

For example, in a built environment, context is an essential factor since it provides information about the current status of users, places, objects, sensors and events. All these information are important and necessary to understand what is really happening within the building. Therefore, we consider that an IVS system deployed in a smart building should be aware of its context to automatically adapt its functionality according to the contextual changes.

In the next chapters will focus on the different ways of obtaining the contextual information of an environment, and how to integrate it with the information coming from an IVS system.

3

CONTEXTUAL INFORMATION AND INTEROPERABILITY

> For me context is the key—from that comes the understanding of everything.
>
> *Kenneth Noland, 1988*

As presented in the previous chapter, visual information is very rich and most of that richness can be exploited with the advances in computer vision. However, current computer vision systems is that they are limited to extract data from what the cameras "observe", more precisely they lack of *contextual information* of the environment. For example, reconsider the Fig. 2.1 presented in the previous chapter. A state-of-the-art computer vision algorithm could easily detect a person in a scene, or the doors in a room, however *no computer vision algorithm* will be able to determine that: the scene happened in a corridor which is located in the third floor of a building, that the door—next to the person—connects the corridor with a storage room, or simply that the person is about to leave the corridor to enter the storage room. Those informations which add "extra meaning" to the image are examples of contextual information in a built environment.

The first section of this chapter will introduce the contextual elements required in an Intelligent Video Surveillance (IVS) system, while Section 3.2 presents some approaches to obtain them. Moreover, Section 3.3 will present some advantages of using an ontology as a machine understandable knowledge representation model, making emphasis in its availability of enabling interoperability between multiple (and maybe) heterogeneous sources of information.

3.1/ ELEMENTS OF CONTEXT

Let us start by answering two simple questions, what is context? and why is it important?

According to the Oxford dictionary, context is defined as: "*the circumstances that form the setting for an event, statement, or idea, and in terms of which it can be fully understood.*"[1]

In other words, context is everything—information, knowledge, etc—that enables a better understanding of an action, event, statement or an image. **Thus, a system that pretends**

[1] https://en.oxforddictionaries.com/definition/context

to be "smart"—such as an Intelligent Video Surveillance (IVS) system—should understand its context in order to have a better interpretation of a perceived action. That is why context is so important!

A system that considers contextual information for decision-making is known as a *context-aware* system. Thereby, an IVS context-aware system (referred simply as IVS system) should accurately perceive sensor data and integrate it with the contextual information, to automatically and "smartly" adapt its services according to the evolving information of the environment. By contextual information we refer to: information about the structure of the building (number and location of storeys, spaces, corridors, etc), its topology (storey-space relation, space-space relation, etc), the different elements contained in the spaces (doors, windows, walls, furnitures, etc), information about events that have occurred (information coming from sensors, their location, occurrence time, people involved, etc) and human-skill knowledge that can facilitate the analyze of a situation (e.g., knowledge that a person should enter a space by a door not by a window, or that a non-authorized person shouldn't be in a restricted space) [109].

As it can be observed, an IVS system deployed in a building should consider highly heterogeneous contextual information; that goes from evolving information coming from a network of sensors, to information concerning the built environment, passing through human-skill knowledge which is based on the fusion of sensor and environment information. Consequently, an IVS system requires and agent that enables the integration between multiple and heterogeneous sources of information (*interoperability*).

The rest of the chapter will focus on how to obtain the required built environment information (Section 3.2), and on how this information can be integrated with the rest of the contextual information (Section 3.3).

3.2/ MODELS TO REPRESENT A BUILT ENVIRONMENT

The need to represent building information in a compact and practical form has motivated several approaches in the history of Architecture, Engineering and Construction (AEC). Starting from the designs using pen and paper, passing through the geometrical models conceived using Computer-Aided Design (CAD) systems, and finally arriving to the models based on the Building Information Modelling (BIM) that goes beyond the geometrical representation of a building and allows to store, manage, exchange and share information between all the agents involved in a building's life-cycle.

The term BIM is becoming the standard in the building-design and -construction fields for over the past 25 years [38]. Eastman et al. [50] defines BIM as a technology that enables "*one or more accurate virtual models of a building to be constructed digitally. They support design through its phases, allowing better analysis and control than manual processes. When completed, these computer generated models contain precise geometry and data needed to support the construction, fabrications, and procurement activities through which the building is realized*". A BIM model is a set of numerical data, objects and processes appended during the complete life-cycle of a building—from the conception to construction and later maintenance. BIM's main advantage is allow the different actors involved in the development of a building—e.g., architectures, engineers, constructors, plumbers, and electricians—to exchange data in a uniform way.

Table 3.1 – Comparison of file formats for BIM. Notice that we did not have access to the proprietary formats RVT and DWG/DXF, therefore their features are unknown.

	Features					
Format	**Open source**	**Machine readable**	**Geometry data**	**Building topology**	**External interoperability**	**Machine understandable**
RVT	✗	–	–	–	–	–
DWG/DXF	✗	–	–	–	–	–
COBie	✓	✗	✗	✗	✗	✗
IFC	✓	✓	✓	✓	✓	✗

An IVS context-aware system could exploit the information contained in a BIM model. However, due to the nature of IVS systems, the BIM model should comprise the following features: (1) it should be open access, i.e., it shouldn't be limited to the use of specific proprietary softwares which harms interoperability; (2) it should be machine-readable (*not only* human-readable) to allow the automatic process of the data; (3) it should include geometric data from which two- and three- dimensional schemes could be generated; (4) it should include topological information, i.e., the relations between the building, the storeys, the spaces and the elements contained in the spaces like door, walls, etc; (5) it should allow the integration with external data sources such as sensors data or building usage information; (6) and it should have a formal semantic representation that will enable machines to understand—process and interpret—the information, i.e., new knowledge could be inferred from existing information thanks to the mathematical formalization of the semantics. There exist many formats used to represent data in a BIM workflow and that will characterize the BIM model itself. The most common BIM formats are: Revit (RVT) [6], Drawing/Drawing Interchange Format (DWG/DXF) [5], Construction Operations Building Information Exchange (COBie) [49] and the Industry Foundation Classes (IFC) [21]. All the previous formats model the built environment information in different manners, resulting in different quality of contextual information. Table 3.1 compares these formats based on the required features. As it can be observed, IFC is the one that posses most of the required features.

The IFC data model is an open specification that is aiming to be a global standard to BIM. The IFC was developed by buildingSMART (previously known as International Alliance for Interoperability)[2] in the 1990s, to facilitate interoperability in the building industry [81]. The IFC is defined using the EXPRESS language [3] and it gives the basis for describing all elements making the building, both semantically and graphically. The semantics level aims to clearly specify the meaning of each element making the BIM [81, 21]. However, as presented in [9, 193], there are several aspects that limit the IFC-EXPRESS format, the most important being the lack of formalization of the semantic information—mathematically rigid theory—and its limited reuse and interoperability due to the unpopularity of the EXPRESS language outside few engineer domains.

According to Table 3.1, none of the BIM formats available allows machines to process and interpret—understand—the information, instead they focus on storing it. However, to bridge this gap many papers propose to represent the BIM information in the form of an ontology [44, 131, 116]. An ontology is a semantic oriented model based on a mathematical formalism, where the knowledge is represented in a structured and well-understood

[2] www.buildingsmart.org

[3] EXPRESS is a standard data modeling language for product data https://en.wikipedia.org/wiki/EXPRESS_(data_modeling_language)

manner. Furthermore, according to Castano et al. [27], ontologies are recognized as *"an essential tool for allowing communication and knowledge sharing among distributed users and applications, by providing a semantically rich description and a common understanding of a domain of interest"*. In this manner, ontologies can also be used as interoperability agents, to combine the heterogeneous contextual information of an IVS system. Therefore, in the next section we will present what is an ontology, how to define it, and how to implement it.

3.3/ ONTOLOGY DOMAIN

In philosophy, the term *Ontology*—in upper-case—refers to the study of "being", "existence" and "reality" [180, 25]. Artificial intelligence (AI) has borrowed the term and gave a more technical meaning to it. An often used definition of *ontology*—in lower-case—is the one of Studer et al.:

> An ontology is a *formal*, *explicit* specification of a *shared conceptualization*. A 'conceptualization' refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon. 'Explicit' means that the type of concepts used, and the constraints on their use are explicitly defined. 'Formal' refers to the fact that the ontology should be machine readable, which excludes natural language. 'Shared' reflects the notion that an ontology captures consensual knowledge, that is, it is not private to some individual, but accepted by a group [163].

Based on that definition, an ontology can be represented via a set of *classes*, *relationships*, *constrains* and *individuals*. Classes (also called concepts) are abstract groups that represent a collection of individuals. Relationships (also called properties) are used to describe features and attributes of classes; they connect a class to other classes or to data values (e.g., integers and strings). Constrains are restriction that determine which relations are allowed and make sense. Individuals (also called instances or objects) are assertions of classes. The statements of membership of individuals in classes and relationships between individuals are called facts.

For exemplification, consider three basic statements capturing the knowledge about a building topology domain:

S.1 *A building contains storeys*.

S.2 *A storey contains spaces*.

S.3 *A building contains the spaces that are contained in its storeys*.

These statement, can be represented in a ontological form by defining the terms `Building`, `Storey` and `Space` as classes; and the terms `containsStorey` (which relates a `Building` with a `Storey`) and `containsSpace` (which relates a `Building` and a `Storey` with a `Space`) as relationships. Notice that classes are normally written in upper-case while relationships and individuals in lower-case. This *building topology*

Figure 3.1 – Graphical representation of a simple building topology knowledge, where the rectangles represent classes while the connections between them represents relationships.

example will be used as running example in the whole section. To have a better visualization of it, Fig. 3.1 presents a graphical representation where the different classes and their relationships can be easily observe.

Moreover, the statement S.3 can be used to deduce implicit knowledge. For example, consider we state the following facts: (i) a building `X` contains a storey `Y`, and (ii) the storey `Y` contains an space `Z`. Then using the knowledge stated in S.3, we can deduce that the building `X` also contains the space `Z` even if this fact was not explicitly stated, this is a simple example of *reasoning* which will be discussed afterwards.

The use of ontologies presents many advantages such as:

- Machine understandable, an ontology *collects* and *represents* knowledge of a given domain, using a semantic formalism (see Section 3.3.1) which enables machines to process and interpret (understand) the information, instead of only store it.
- Knowledge sharing, ontologies provide a shared and common understanding of a domain which can be communicated to multiple people and application systems. Therefore, ontologies are a way of standardizing and facilitating knowledge sharing and *reuse*. Moreover, the standardization enables more rapid development of applications, because we do not have to "re-invent the wheel".
- Data organization, an ontology provides a flexible and natural mean of organizing (*structuring*) data according to a hierarchy of classes and their relations, thus facilitating data browsing.
- Search improvement, the ontology's semantics aid for *disambiguation* during search. For example in linguistic, the term "space" is a polysemic word (i.e., it has many possible meanings), it might refer to the outer space, the physical space (framework of distance and directions), a film, a book, a game etc, hence the addition of a semantic annotation will facilitate the retrieval of the desired entity.
- Reasoning, the semantic formalism of the ontology enables the *inference* of implicit knowledge and facts from explicit ones. Moreover, the *ontology reasoning* can be used for query answering (i.e., finding patterns), to detect inconsistencies (i.e., violation of a constraints) and to identify redundancies.
- Data integration, due to the explicit and formal description of data and knowledge, an ontology can serve as semantic glue between heterogeneous information sources (i.e., sources which were not designed to work together), achieving *serendipitous interoperability* [92], i.e., discovered by chance in a beneficial way.

Interoperability is a crucial characteristic of multi-source systems. Some common heterogeneity factors that hinder the interoperability are the variety of data formats, different

data organization (schema), different protocols, the use of multiple languages, etc. An ontology overcomes these *semantic heterogeneity* issues by providing a uniform access to the multiple information sources, thus semantically unifying the data [1].

As presented in Section 3.1, an IVS context-aware system needs to consider multiple and heterogeneous sources of information such as information of the environment, data from the sensors and information about the different computer vision algorithms performed. Therefore, an IVS context-aware system needs an *interoperability agent* to deal with the multi-heterogeneous sources. Some important works that used the ontology as an interoperability agent in the built environment and computer vision domains are:

- Hong et al. presented many context-aware systems where the ontology plays a central role for enabling interoperability between devices which were not designed to work together [73].
- Dibley et al. developed an ontology framework that combines a sensor ontology with a building ontology and other ontologies [45].
- SanMiguel et al. used an ontology for combining image processing algorithms with knowledge about objects and events [150].
- Chaochaisit et al. presented a semantic connection between sensor specifications, localization methods and contextual information [29].
- Town presented an ontology that fuses multiple computer vision stages with context information for image retrieval and event detections [169].
- Suchan and Bhatt developed a framework that reason about human activities using commonsense knowledge founded in qualitative spatio-temporal relations and human-object interactions [164].

In all those works, the ontology played a crucial role in enabling the processing and sharing of information and knowledge between multiple sources.

Now that we know what is an ontology and what are its main advantages, the following subsections will focus on the mathematical formalism used to define them and the languages used to implement them, i.e., we will pass from the ontology conceptualization to its definition and finish with its implementation.

3.3.1/ ONTOLOGY FORMALISM

The representation of the *building topology* example presented in Fig 3.1 is easy to read and understand by humans, however that representation is *informal*, i.e., it is ambiguous (different people may represent the same knowledge differently, take for example the representation of statement S.3), and thus a machine wont be able to automatically process and understand it.

Therefore, ontologies need to be specified using a mathematical *formal* description language to be *machine understandable*. There exists several formal description languages that can be used to define an ontology, which differ by the level of *expressivity* (i.e., the variety or quantity of ideas that can be represented in that language) and *decidability* (i.e.,

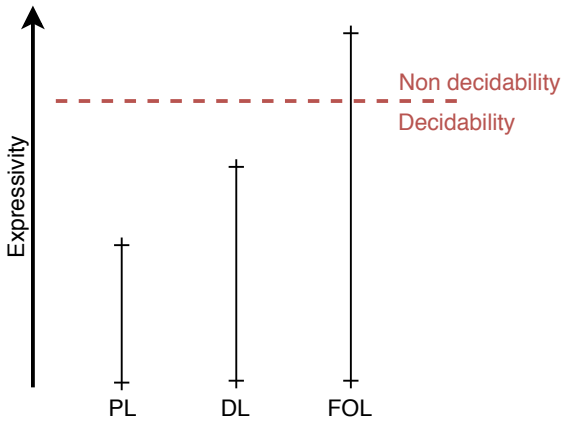


Figure 3.2 – Expressivity and decidability comparison of Propositional Logic (PL), Description Logic (DL) and First-Order Logic (FOL) formalisms.

the existence of effective methods for deriving a correct answer within an acceptable time span). The most well-known formal description languages are:

- Propositional Logic (PL), which is a branch of logic that only deals with binary propositions (*true* or *false* values) and their combination using logical operators (*and*, *or* and *not* operators) [87]. The sentence "if you have a son then you are a parent" is a simple example of a PL statement. Furthermore, PL is consider as zero-order logic, as it does not deal with non-logical objects such as quantifiers or variables, however, it is the foundation of all higher order logics. PL has a low expressivity level however it is decidable;
- First-Order Logic (FOL), extends PL by considering variables, existential and universal quantifiers, and non-logical relations (predicates) [15]. Thus, it is possible to extend the PL expression by the following FOL sentence "there exists X such that X has a son and X is a parent", where *there exists* is an existential quantifier while *X* is a variable. Furthermore, the relation *parent* between to variables *X* and *Y* is a predicate. FOL has a high expressivity level, however, its theoretical proof is very complex, consequently, there is no terminating decision algorithm that can determine if arbitrary formulas are logically valid or not, thus making it semi-decidable;
- Description Logics (DLs), are *decidable* fragment of FOL, which provides a set of constructors to describe classes, instances, relations and constraints. DLs are suited for modeling, due to its sufficient level of expressivity and low complexity which guaranties a complete and terminating reasoning.

Figure 3.2 summarize the expressivity and decidability differences between PL, DL and FOL. The relation between expressivity and decidability is as follows: **higher the expressivity of a language, higher its computational complexity and thus lower its decidability**. Therefore, due to its good balance between expressivity and decidability, DL is the standard formalism used to define ontologies [7].

Description logics (DLs) are a family of *formal* knowledge representation languages, used to represent knowledge in a structured and well-understood way [7]. Semantically, DL languages are decidable fragments of the classical FOL that allows a mathematical formalization of knowledge and the use of automated theorem proving, referred as *reasoning*.

Table 3.2 – Description Logic conventional notation. Where C and D are concepts, R and P are roles and a and b are instances.

Symbol	Description	Example	Read
$\top$	Most general concept	$\top$	Top
$\bot$	Empty concept	$\bot$	Bottom
$\sqcap$	Intersection	$C \sqcap D$	C and D
$\sqcup$	Union	$C \sqcup D$	C or D
$\neg$	Negation	$\neg C$	not C
$\sqsubseteq$	Inclusion	$C \sqsubseteq D$	All C are in D (subclass of)
$\equiv$	Equivalence	$C \equiv C$	C is equivalent to D
$\forall$	Universal restriction	$\forall R.C$	All R-successors are in C
$\exists$	Existential restriction	$\exists R.C$	An R-successor exist in C
$\circ$	Role chain	$R \circ P$	R-successor is P-predecessor
$^-$	Role inverse	$R \equiv P^-$	R is inverse of P
:	Concept assertion	$a : C$	a is a C (instance of)
:	Role assertion	$(a, b) : R$	a is R-related to b

By *reasoning* we mean deriving facts or information which were not express explicitly in the ontology , i.e., deducting new entailment of statements which are not express explicitly in the knowledge base). According to Baader et al. [7], common reasoning tasks are:

- checking the *satisfiability* and *consistency* of facts—determining weather a class or instance violates/contradicts the description (model);
- checking for *subsumption* of classes—determining if a class subsumes (is more general) than another;
- *retrival* of instances—finding all individuals that are instances of a class;
- *answering queries*—respond to database-style queries over the knowledge statements.

The family of DL languages differ by their level of expressiveness which is determined by the supported class and property constructors, and by the type of axioms they allow [7]. Table 3.2 presents the main symbols used to define DL constructors and axioms. Notice that in DL, classes are referred as *concepts*, relationships/properties are referred as *roles* and individuals as *objects*. Moreover, let N_C, N_R and N_O be (respectively) sets of concept names (also known as atomic concepts), role names and individual names; and $C, D \in N_C$, $R, P \in N_R$ and $a, b \in N_O$. Furthermore, if a is related to b by the role R, then we say that a is R-*related* to b, that b is the R-successor of a, and that a is the R-predecessor of b.

Currently, the most expressive and decidable DL language is $\mathcal{SROIQ}(\mathcal{D})$, where each letter represents a set of constructors as shown in Table 3.3. Furthermore, Table 3.4 present some axioms, assertions and property characteristics allowed in $\mathcal{SROIQ}(\mathcal{D})$. Notice that the Web Ontology Language (OWL) notation will be used afterwards, in the *implementation* section. The constructors can be nested arbitrary to create axioms, for example the *role domain* and *range* axioms, which respectively restrict the class of the predecessor

Table 3.3 – $\mathcal{SROIQ}(\mathcal{D})$ constructors, with their corresponding OWL functional syntax [121]. Where C and D are concepts, R and P are roles, a and b are instances, d is a data property, v is a data value and n is a non-negative integer.

	Constructor	DL syntax	OWL functional syntax
$\mathcal{S}$	Concept negation	$\neg C$	`ObjectComplementOf(` C `)`
	Concept intersection	$C \sqcap D$	`ObjectIntersectionOf(` $C\ D$ `)`
	Concept union	$C \sqcup D$	`ObjectUnionOf(` $C\ D$ `)`
	Universal restriction	$\forall R.C$	`ObjectAllValuesFrom(` $R\ C$ `)`
	Existential restriction	$\exists R.C$	`ObjectSomeValuesFrom(` $R\ C$ `)`
$\mathcal{R}$	Role inclusion	$R \sqsubseteq P$	`SubObjectPropertyOf(` $R\ P$ `)`
	Complex role inclusion	$R \circ P \sqsubseteq T$	`SubObjectPropertyOf(` `ObjectPropertyChain(` $R\ P$ `))`
$\mathcal{O}$	Nominals	$\{a\}, \{b\}$	`ObjectOneOf(` $a\ b$`)`
$\mathcal{I}$	Inverse role	$R \equiv P^-$	`InverseObjectProperties(` $R\ P$ `)`
$\mathcal{Q}$	Qualified cardinal restriction	$\geq n\,R.C$	`ObjectMinCardinality(` $n\ R\ C$ `)`
		$\leq n\,R.C$	`ObjectMaxCardinality(` $n\ R\ C$ `)`
$(\mathcal{D})$	Data properties	$(a, v)\colon d$	`DataPropertyAssertion(` $d\ a\ v$ `)`

and the successor of a property. Moreover, $\mathcal{SROIQ}(\mathcal{D})$ allows to declare special characteristics to properties. To exemplify each characteristic, we will use *family-relationships* which are familiar to everybody:

- **Transitive**, R is transitive if: $(a, b)\colon R$ and $(b, c)\colon R$ then $(a, c)\colon R$. For example, take the property `hasAncestor`, if a `hasAncestor` b and b `hasAncestor` c, then a also `hasAncestor` c.
- **Symmetric**, R is symmetric if: $(a, b)\colon R$ then $(b, a)\colon R$. For example, consider the property `hasSpouse`, if a `hasSpouse` b, then b also `hasSpouse` a.
- **Asymmetric**, R is asymmetric if: $(a, b)\colon R$ then $\neg(b, a)\colon R$. For example, take the property `hasChild`.
- **Reflexive**, R is reflexive if: $(a, a)\colon R$ for all a. For example, consider the property `hasRelative`.
- **Irreflexive**, R is irreflexive if: $\neg(a, a)\colon R$ for any a. For example, consider the property `parentOf`.

There exists other DL languages which are less expressive than $\mathcal{SROIQ}(\mathcal{D})$, i.e., they only consider a portion of $\mathcal{SROIQ}(\mathcal{D})$'s constructors. For example $\mathcal{SHOIQ}(\mathcal{D})$ *does not* consider *complex role inclusion*; or $\mathcal{SHIQ}(\mathcal{D})$ and $\mathcal{SHOQ}(\mathcal{D})$ do not consider *complex role inclusion* and (respectively) *nominals* and *inverse roles*. The price of high expressivity is paid in computational complexity of reasoning. $\mathcal{SROIQ}(\mathcal{D})$ has complexity of *N2ExpTime*, compare to $\mathcal{SHOIQ}(\mathcal{D})$'s *NExpTime* and $\mathcal{SHIQ}(\mathcal{D})$'s and $\mathcal{SHOQ}(\mathcal{D})$'s *ExpTime*, where the complexity classes are ordered according to following set inclusion [71]:

$$ExpTime \subseteq NExpTime \subseteq N2ExpTime \tag{3.1}$$

Introducing complexity theory is beyond the scope of this thesis, the reader is referred to [3] for a comprehensive overview of these classes. Moreover, the complexity of an

Table 3.4 – Some $\mathcal{SROIQ}(\mathcal{D})$ axioms, assertions and property characteristics, with their corresponding OWL functional syntax [121]. Where C and D are concepts, R and P are roles, a and b are instances, d is a data property and v is a data value.

Axioms	**DL syntax**	**OWL functional syntax**
Concept inclusion	$C \sqsubseteq D$	`SubClassOf(` $C\ D$ `)`
Concept equivalence	$C \equiv D$	`EquivalenceClasses(` $C\ D$ `)`
Concept assertion	$a\colon C$	`ClassAssertion(` $C\ a$ `)`
Role assertion	$(a,b)\colon R$	`ObjectPropertyAssertion(` $R\ a\ b$ `)`
Role domain	$\exists R.\top \sqsubseteq C$	`ObjectPropertyDomain(` $R\ C$ `)`
Role range	$\top \sqsubseteq \forall R.C$	`ObjectPropertyRange(` $R\ C$ `)`
Role transitivity	`Trans(`R`)`	`TransitiveObjectProperty(` R `)`
Role symmetry	`Sym(`R`)`	`SymmetricObjectProperty(` R `)`
Role antisymmetry	`Asym(`R`)`	`AsymmetricObjectProperty(` R `)`
Role reflexivity	`Ref(`R`)`	`ReflexiveObjectProperty(` R `)`
Role irreflexivity	`Irref(`R`)`	`IrreflexiveObjectProperty(` R `)`

ontology do not only depends on its expressivity but also in the quantity of axioms considered.

Even though $\mathcal{SROIQ}(\mathcal{D})$ has a high complexity, its decidability was proved in [76].

The *building topology* example can be defined using the $\mathcal{SROIQ}(\mathcal{D})$'s constructors and axioms as follows:

- Equations 3.2 and 3.3 define (respectively) the domain and range of property `containStorey`. Meaning that `containStorey` can only be used to relate a `Building` object to a `Storey` instance, which is a formal way representing the knowledge stated in S.1.

$$\exists \texttt{containsStorey}.\top \sqsubseteq \texttt{Building} \tag{3.2}$$

$$\top \sqsubseteq \forall \texttt{containsStorey}.\texttt{Storey} \tag{3.3}$$

- Equations 3.4 and 3.5 define (respectively) the domain and range axioms of property `containSpace`. Meaning that `containSpace` can only be used to relate a `Building` or a `Storey` object to a `Space` object, which is a formal way representing the knowledge stated in S.2 .

$$\exists \texttt{containsSpace}.\top \sqsubseteq (\texttt{Building} \sqcup \texttt{Storey}) \tag{3.4}$$

$$\top \sqsubseteq \forall \texttt{containsSpace}.\texttt{Space} \tag{3.5}$$

- Equation 3.6 defines the complex role inclusion between properties `containStorey` and `containSpace`. Which can be read as "if an object `X` is related by `containsStorey` to object `Y`, and `Y` is related by `containsSpace` to object `Z`, then `X` is also related to `Z` by `containsSpace`", which is a formal way of representing the knowledge stated in S.3.

$$\texttt{containsStorey} \circ \texttt{containsSpace} \sqsubseteq \texttt{containsSpace} \tag{3.6}$$

After formalizing the ontology, we need to implement it using a language that can facilitate its applicability and authoring (creation of programs and databases for computer applications). Recently, DLs have played a central role in the development of the semantic web [7, 74], specially in the development of ontology languages such as the Web Ontology Language (OWL), and its predecessors the Ontology Inference Layer (OIL) and the Darpa Agent Markup Language + OIL (DAML+OIL) [78].

Remark. *For more details about DLs, $\mathcal{SROIQ}(\mathcal{D})$ and the definition of the different reasoning task we refer the readers to [7, 76].*

3.3.2/ ONTOLOGY IMPLEMENTATION

The implementation or materialization of an ontology is known as a *Knowledge Base* (KB). A KB consists of a *terminological knowledge*, called the **TBox**, and an *assertional knowledge*, called the **ABox**. The TBox expresses the general structure on what (the abstraction of) the domain looks like, and it is composed of the set of classes, relationships and constrains. In the other hand, the Abox defines the concrete elements, their attributes and their relations (i.e., data), and is composed of the set of instances and instances-facts [7]. For simplicity, in this manuscript both terms—"ontology" and "knowledge base"—will be used interchangeably, even if the former is a conceptualization and the latter is its implementation.[4]

Currently, the most common language for implementing KBs is the Web Ontology Language (OWL) which was developed by the World Wide Web Consortium (W3C), the main international standards organization for the World Wide Web.[5] The second revision of OWL, denoted as OWL 2, is the current recommendation of W3C [182]. The semantics of OWL 2 are defined using the expressiveness of the $\mathcal{SROIQ}(\mathcal{D})$ DL language [76]. Tables 3.3 and 3.4 present the OWL syntax of DL's constructors and axioms.

For exemplification of a KB, OWL and interoperability, let us consider a simple KB about an IVS system, where the *building topology* example is combined/merged with some *visual sensor network* knowledge. The set of TBox and ABox statements are:

TBox.i *A building contains storeys*.

TBox.ii *A storey contains spaces*.

TBox.iii *A building contains the spaces that are contained in its storeys*.

TBox.iv *A space contains building elements such as a sensor*.

TBox.v *A smart camera is a type of sensor*.

TBox.vi *A smart camera observes the space that contains it*.

ABox.i *I3M is a building*.[6]

ABox.ii *I3M contains storey Level-3* .

ABox.iii *Level-3 contains space PhD-room* .

ABox.iv *PhD-room contains element Raspberry-Pi, which is a smart camera*.

[4] This is a common practice found in the literature.
[5] https://www.w3.org/
[6] The Institut Marey Maison de la Metalurgie (I3M) building is located in Dijon, France

Those KB statements can be represented—in a formal manner—as an OWL 2 ontology, as shown in Listing 3.1, where: Lines 2-15 declare the classes, relationships and individuals, e.g., `Building` is a class, `containsStorey` is a relationship and `Level-3` is an instance; Lines 19-20 state the subclass relationships stated in TBox.iv and TBox.v; Lines 26-33 constrains the relationship's domains and ranges based on the TBox statements; Lines 36-37 state the complex role inclusion stated in TBox.iii; and Lines 40-44 define the ABox assertion statements.

Moreover, the OWL 2 formalization allows—thanks to its formal semantics—the deduction of implicit knowledge. Specifically, the following five facts can be deduced from the previous TBox and ABox statements:

- `Level-3 is-a Storey`. Deduced thanks to statement TBox.i and fact ABox.ii.
- `PhD-room is-a Space`. Deduced by using statement TBox.ii and fact ABox.iii.
- `I3M containsSpace PhD-room`. Deduced thanks to statement TBox.iii and facts ABox.ii and ABox.iii.
- `Raspberry-Pi observes PhD-room`. Deduced by using statements TBox.vi and facts ABox.iv.
- `Raspberry-Pi is-a Sensor`. This fact is not shown in the Fig. 3.3 for simplicity, however, it can be deduced by using statement TBox.v and facts ABox.iv.

Figure 3.3 presents an *informal* representation of the IVS KB in a graphical form. As presented in the figure, the ontological definition of both domains—visual sensor network and built environment—enables semantic homogeneity which allows the semantic *interoperability* between heterogeneous domains. Moreover, notice that in the figure we have called Wised NETwork (WiseNET) to the merging between the domains, WiseNET is the name of our system and it will be explained in the next chapter.

SEMANTIC WEB

We cannot talk about ontology implementation without talking about the Semantic Web (SW). As it will be presented in this section, the ontology universe has greatly benefited from the SW "boom", and many Web standards and technologies developed for SW allow to implement, extend and interact with ontologies.

As Berners-Lee stated [12], the SW is

> an extension of the current Web, in which information is given well-defined meaning, better enabling computers and people to work in cooperation.

The *well-defined meaning* mentioned by Berners-Lee refers to *semantic information* that explains the *provenance* and *usability* of data or a resource. Therefore, an ontology is perfect for the job, thus it has taken a central role in the development of the SW [74].

The SW's vision has stimulated the development of many technologies in order to create a common framework for representing, publishing, sharing, exchanging and integrating data (and knowledge) from different sources. It is beyond the scope of this manuscript to provide a detailed description of all the different SW technologies, however a brief sketch

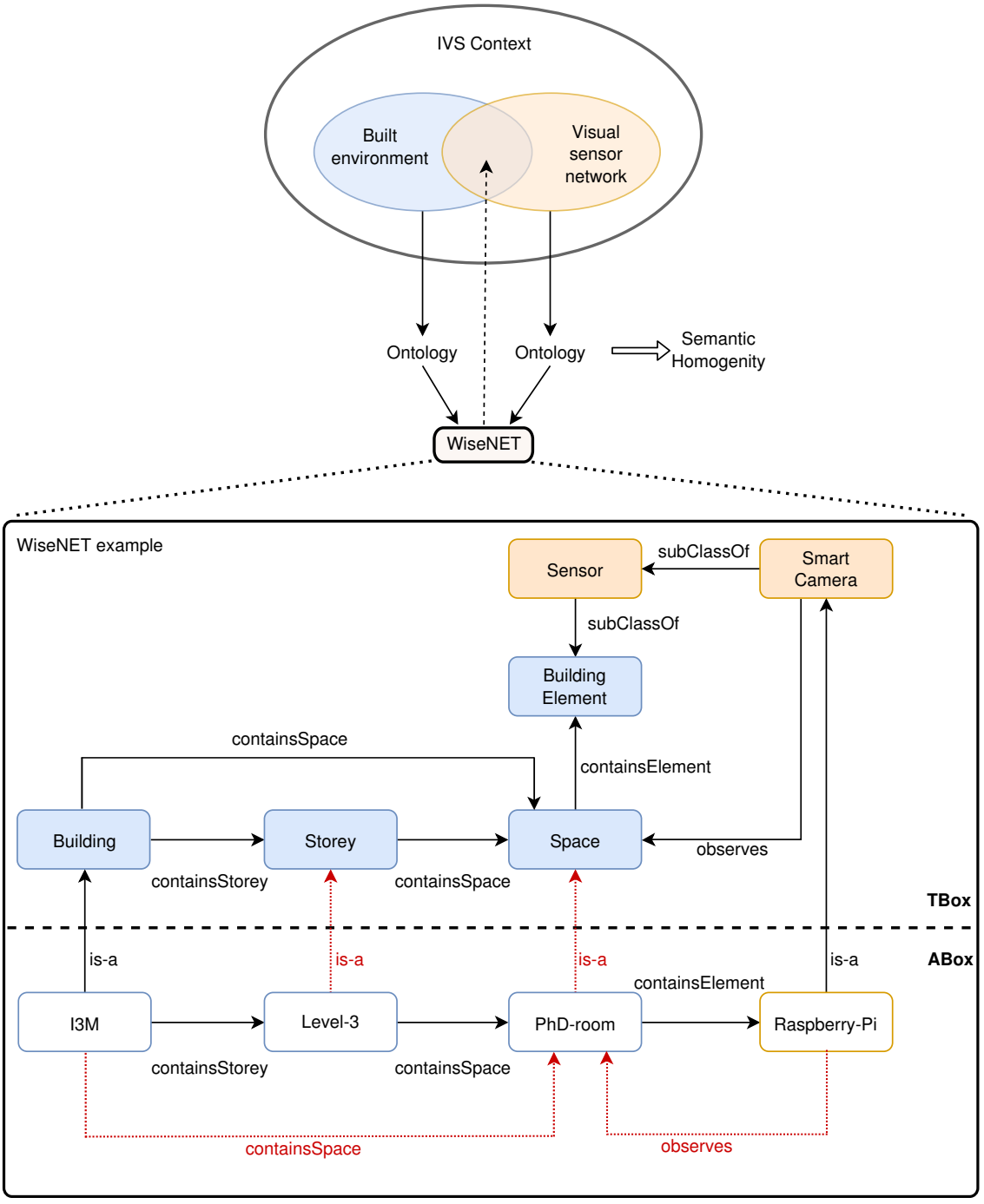


Figure 3.3 – An informal, graphical view of knowledge about Intelligent Visual Surveillance (IVS) context. TBox stands for *terminology knowledge* and ABox stands for *assertional knowledge*. Classes are shown in filled rectangles, relations are represented by the black arrows, instances are shown in empty rectangles and inference relations are represented by red arrows.

of the ones that empowered the ontologies will be given; interested readers are to refer to `https://www.w3.org/standards/semanticweb/` for complete information. Notice that all the technologies presented below—less SWRL—are W3C recommendations thus they are considered as Web standards.

- RDF [41], stands for Resource Description Framework, and it provides a graph-like data model, where each statement is represented as a *triple*, often written as

$$\langle s, p, o \rangle, \tag{3.7}$$

where s is the subject, p the predicate and o the object. The use of triples provides a natural way to capture ABox assertions [7], for example the triples $\langle a, R, b \rangle$ and $\langle a, \texttt{rdf:type}, C \rangle$ correspond respectively, to the role assertion $(a, b) \colon R$ and concept assertion $a \colon C$ in Table 3.4. Also, RDF assigns special meaning to certain predicates, for example the previous predicate `rdf:type` represents the "`is-a`" relation. RDF is the standard model for data interchange on the Web. Both RDFS and OWL are modeling languages for describing RDF data.

- RDFS [19], stands for RDF Schema, and it extends RDF by assigning special predicates to capture "schema" level statements. For example, the RDFS special predicates `rdfs:subClassOf` and `rdfs:subPropertyOf` allow the creation of hierarchies between classes and properties respectively, such as $\langle C, \texttt{rdfs:subClassOf}, D \rangle$ and $\langle R, \texttt{rdfs:subPropertyOf}, P \rangle$ which correspond, respectively, to the concept inclusion $C \sqsubseteq D$ and role inclusion $R \sqsubseteq P$ in Tables 3.4 and 3.3.

- OWL [182], stands for Web Ontology Language, and is the standard and most common language to formally represent knowledge. As stated previously, most of OWL's logic-semantics is based on DL, however its data structure and some of the schema axioms are based on two pre-existing W3C recommendations: RDF and RDFS.

- OWL Functional syntax [121], which is a high level syntax close to the ontology structure of OWL. Tables 3.3 and 3.4 presents some translations from DL to OWL functional syntax, moreover, a complete example of the use of this syntax is presented in Listing 3.1.

- SPARQL [183], stands for SPARQL Protocol and RDF Query Language, and it provides standard means to interact with the ontology's data by performing database-style queries. SPARQL provides specific graph traversal syntax for data that can be thought of as a graph. Therefore, SPARQL can be seen as a basic *graph pattern matching*, where the graph patterns are RDF triples that contains variables at any arbitrary place (subject, predicate, object). The SPARQL features are: extraction of data, exploration of data, transformation of data, construction of new RDF graph, updates of RDF graphs and logical deduction.

  The most common SPARQL query commands are:

  - **SELECT** query, used to query the RDF database and show all results that fulfill the required conditions (pattern). The results are return in a table format.
  - **ASK** query, used to check whether there exist at least one result. The result is boolean.
  - **DESCRIBE**, used to give all the information concerning the result. The result is an RDF graph.
  - **CONSTRUCT**, used to construct a new RDF graph from a template.
  - **INSERT DATA**, used to add some triples.
  - **DELETE DATA**, used to remove some triples.

Each of these query forms takes a **WHERE** block to restrict the query.

For exemplification, Listing 3.2 presents and SPARQL query over the IVS example ontology. The query returns the instances of `Sensor` that are contained in the `PhD-room`. This query joins together all of the triples where the predicate and object are `rdf:type` and `Sensor` (Line 3), with all the triples where subject and predicate are `PhD-room` and `containsElement` (Line 4). The result of this join, which itself is the result of the query, is stored in the variable *?sensors*. In this simple example the result is the following one element table,

?sensors
Raspberry-Pi

.

- SWRL [77], stands for Semantic Web Rule Language, and it allows to overcome some expressivity limitations of DL languages by inserting knowledge in the form of semantic rules, i.e., SWRL allows to define expressions which can not be defined using OWL DL alone. For example, DL formalisms does not allow the composition of complex classes and properties by combining both classes and properties simultaneously, however this limitation can be overcome by using rule-based knowledge. Moreover, the rules should have the form of an implication between an antecedent (body) and consequent (head):

$$\mathtt{b}_1 \wedge \ldots \wedge \mathtt{b}_n \Longrightarrow \mathtt{h}_1 \wedge \ldots \wedge \mathtt{h}_n, \tag{3.8}$$

where $\mathtt{b}_1, \ldots, \mathtt{b}_n$ are the antecedents of the rule and $\mathtt{h}_1, \ldots, \mathtt{h}_n$ are the consequent. Both antecedents and consequents should be a conjunctions of atoms. The intended meaning can be read as: whenever the antecedent conditions hold, then the consequent conditions must also hold. Variables are indicated using the standard convention of prefixing them with a question mark (e.g., `?x`).

Using this syntax, a SWRL rule asserting that "*the elements contained in a space are also contained by the building containing that space*", can be written as,

$$\begin{aligned} &\texttt{Building(?x)} \wedge \texttt{Space(?y)} \\ &\wedge \texttt{containsSpace(?x,?y)} \\ &\wedge \texttt{containsElement(?y,?z)} \\ &\Longrightarrow \texttt{containsElement(?x,?z)}, \end{aligned} \tag{3.9}$$

and can be read as "*if there is a building 'x' that contains a space 'y' and 'y' contains an element 'z' then 'x' contains 'z'* ". Currently, SWRL is not W3C recommendation, however is still used in many application.

Remark. *The list of Semantic Web technologies presented in this section is by no means exhaustive, it focuses only on the technologies that are relevant to our work.*

Listing 3.1 – Intelligent Visual Surveillance (IVS) ontology example using OWL functional syntax [121]. Lines that start by '#' are comments.

```
### Declaring classes, relations and instances,
Declaration(Class(:Building))
Declaration(Class(:BuildingElement))
Declaration(Class(:Sensor))
Declaration(Class(:SmartCamera))
Declaration(Class(:Space))
Declaration(Class(:Storey))
Declaration(ObjectProperty(:containsElement))
Declaration(ObjectProperty(:containsSpace))
Declaration(ObjectProperty(:containsStorey))
Declaration(ObjectProperty(:observes))
Declaration(NamedIndividual(:I3M))
Declaration(NamedIndividual(:Level-3))
Declaration(NamedIndividual(:PhD-room))
Declaration(NamedIndividual(:Raspberry-Pi))

### Stating class hierarchies,
# e.g.: 'Sensor' is a sub-class of 'BuildingElement'.
SubClassOf(:Sensor :BuildingElement)
SubClassOf(:SmartCamera :Sensor)

### Constraints,
# e.g.: 'containsStorey' can only be used
#       by a 'Building' individual (its domain)
#       to connect to a 'Storey' individual (its range)
ObjectPropertyDomain(:containsElement :Space)
ObjectPropertyRange(:containsElement :BuildingElement)
ObjectPropertyDomain(:containsSpace ObjectUnionOf(:Building :Storey))
ObjectPropertyRange(:containsSpace :Space)
ObjectPropertyDomain(:containsStorey :Building)
ObjectPropertyRange(:containsStorey :Storey)
ObjectPropertyDomain(:observes :SmartCamera)
ObjectPropertyRange(:observes :Space)

### Role Chain,
SubObjectPropertyOf(ObjectPropertyChain(
        :containsStorey :containsSpace) :containsSpace)

### Instances,
ClassAssertion(:Building :I3M)
ObjectPropertyAssertion(:containsStorey :I3M :Level-3)
ObjectPropertyAssertion(:containsSpace :Level-3 :PhD-room)
ObjectPropertyAssertion(:containsElement :PhD-room :Raspberry-Pi)
ClassAssertion(:SmartCamera :Raspberry-Pi)
```

Listing 3.2 – SPARQL query for getting the sensors located in the PhD-room.

```
SELECT ?sensors
WHERE {
        ?sensors rdf:type Sensor.
        PhD-room containsElement ?sensors.
}
```

3.4/ CONCLUSION

In this chapter, we introduced the notions of context and context-aware system in the built environment. Furthermore, we presented a semantic model that can be used to represent the context, an ontology.

A "smart" system should no longer be limited to connecting sensor information, but it must also consider the contextual information in order to have a better interpretation of a perceived action. Thus, it could adapt its services according to the context. In the built environment, contextual information refers to: information about the structure of the building, its topology, the different elements contained in the spaces, information about the sensors deployed in the environment, information about events that have occurred, and human-skill knowledge about the environment. Thus, a smart system deployed in a built environment requires an agent that (1) will be able to represent the multiple and heterogeneous sources of information and that (2) will enable interoperability between them. An ontology agent fulfills both requirements.

One important feature of ontology is to enable *semantic fusion*, which consists in integrating and organizing data and knowledge coming from multiple heterogeneous sources and to unify them into a consistent representation, i.e., it enables interoperability between heterogeneous sources of information. Therefore, many works have been done using ontologies along with different semantic web standards, to represent contextual data. Furthermore, important works were made to represent the environment information using an ontology-based model. This way of modeling environment information presents many advantages compared to others, as shown in Table 3.5. Specifically, the ontology-based model is the only one to be *machine understandable* due to its formal semantic representation, i.e., new knowledge could be inferred from existing information thanks to the mathematical formalization of the semantics. This is an important feature in smart applications were machines should be able to automatically process and understand the information.

The ontology development is guided by the formalisms and tools for knowledge representation that have emerged over the past decades. Thus, we introduced how to pass from the ontology conceptualization to the formalization, until arriving to the ontology implementation. In the next chapter, we will present the process followed for the development of an ontology, that will combine the information extracted by a smart camera network with building information, events that may occur and human-skills that help the analysis of the information. The process uses all of the semantic web technologies explained in this chapter.

Table 3.5 – Comparison of file formats for BIM. Adaptation of Table 3.1, with the *ontology* format added.

	Features					
Format	**Open source**	**Machine readable**	**Geometry data**	**Building topology**	**External interoperability**	**Machine understandable**
RVT	✗	–	–	–	–	–
DWG/DXF	✗	–	–	–	–	–
COBie	✓	✗	✗	✗	✗	✗
IFC	✓	✓	✓	✓	✓	✗
ontology	✓	✓	✓	✓	✓	✓

4

WISENET SYSTEM

As presented in the previous chapters, the main function of an Intelligent Video Surveillance (IVS) system is to combine different information in the context of a built environment, to enable automatic and real-time deduction of events and the triggering of actions. Hence, a IVS system should not only understand the static data of an environment, but it should also perceive and understand the environment's dynamic and evolving data, i.e., it should be aware of its context. The dynamic environment data can be obtained from different sensors deployed in the environment. Due to the IVS application, we focus on visual sensor networks, specifically on Smart Cameras Networks (SCNs) that can perform a plethora of Computer Vision (CV) algorithms (see Chapter 2 for more details). As well, the static environment data—e.g., it's topology and the elements contained in it—-can be obtained from a Building Information Modelling (BIM) model defined using the Industry Foundation Classes (IFC) specification, which is becoming the standard in the building-design and -construction fields (see Chapter 3 for more details).

The creation of an IVS context-aware system is a complex task. Even if the environment's static and dynamic data can be obtained from existing models/algorithms, an IVS system requires extra information. For example, information about the different events that may occur in the environment, their location, their occurrence time, the agents involved in them, the relation to other events and their consequences. Moreover, an IVS context-aware system should be able to combine all those heterogeneous data sources to have a correct vision of what is really happening in the built environment.

To achieve our vision of an IVS context-aware, we developed the WiseNET system, which will gather and combine the heterogeneous data sources, and will enable automatic deduction of events and ease the monitoring of a built environment.

This chapter is divided into two parts. Firstly, Section 4.1 introduces the WiseNET system and the main elements that compose it. Secondly, Section 4.2 presents the methodology followed to build the WiseNET ontology and a way to extend it by including semantic rules. The WiseNET ontology is the core element of the WiseNET system, and it is in charge of enabling the interoperability between the previously mention heterogeneous data sources and allowing the deduction of new facts. Thus, at the end of this chapter, the kernel of our WiseNET system will be completely defined, and it will be ready to be populated with the static and dynamic data, as it will be presented in Chapter 5.

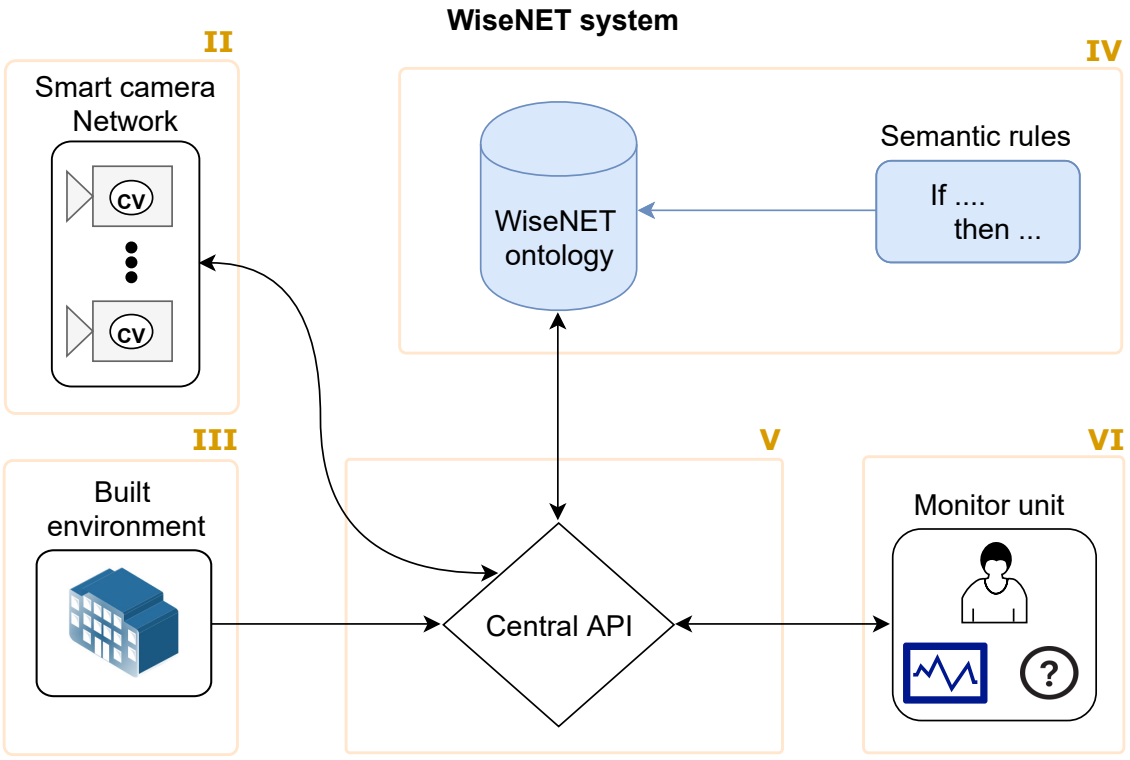


Figure 4.1 – Overview of the WiseNET system. The number of each module corresponds to the thesis chapter that explains it. We highlighted in light blue the elements that will be explained in this chapter.

4.1/ WISENET SYSTEM OVERVIEW

The *Wised-NETwork* (WiseNET) is a semantics-based system that aggregates heterogeneous data such as data coming from a SCN (after performing CV algorithms within the smart camera), and multiple contextual information (e.g., building topology and building usage). The main goals of the WiseNET system are: (1) to enhance classical computer vision by considering the contextual information of the environment and by performing real-time reasoning, (2) to provide a set of innovative services to building managers, to ease their work. As a result, WiseNET overcomes some limitations of computer vision— such as occlusions— as well as limitations of classical Visual Sensor Network (VSN) deployed in a built environment—such as human-monitoring high cognitive load and privacy protection. Moreover, the WiseNET system enables building managers to perform queries, in real-time, to have information related to the environment and its usage.

Figure 4.1 illustrates an overview of the WiseNET system. As shown, the system is composed of six main elements: smart camera network, built environment, WiseNET ontology, semantic rules, central API, and monitor unit

Smart camera network The SCN is a set of smart cameras distributed in an environment. The main functions of the SCN, in the WiseNET system, is to constantly extract data from a scene—by performing CV algorithms—to convert it into knowledge and then to send it to the central unit. The data extracted by the SCN represents the evolving in-

formation of the environment, thus we will refer to it as *dynamic data*. Moreover, there is the calibration of the SCN, which is performed once during the system set-up, thus, this data will be referred as *static calibration data*.

Notice that our contribution is not in designing a new SCN or a new CV algorithm, but is in the extraction of knowledge from the results of CV algorithms and their combination with other knowledge. Thus, the WiseNET system is totally independent on the hardware used (i.e., type of smart camera) or the CV algorithms deployed on them, meaning that we could use different smart cameras and different CV algorithms without changing the system itself. However, the hardware and software choice will have an impact on the final result of the system, i.e., a smart camera with high resources will cope better with real-time constraints; in the same manner, a robust CV algorithm will increase the overall accuracy of the system.

As a guide, Chapter 2 presented different smart cameras as well as different CV algorithms in the domain of IVS. Furthermore, this chapter will introduce the vocabulary used to represent the SCN data in the form of an ontology. Moreover, the processing of the *dynamic data* and the *static calibration data* will be presented in Chapter 5, while the impact of using different CV algorithms will be presented in Chapter 6.

Remark. *Based on the IVS application, we focused in this manuscript on a specific type of sensors: smart cameras; however, the WiseNET system is not dependent on this type of sensors, and could be easily extended to include other sensors such as temperature, humidity, depth sensor, etc.*

Built environment A IVS system should understand its environment and consider its context, in order to have a better interpretation of a perceived action. The WiseNET system focus on indoor built environments contexts. The environment data required by the WiseNET system are the structure of the building (i.e., number and location of storeys, spaces, doors, etc), its topology (i.e., the building-storey, storey-space and space-space relation) and the different elements contained in the spaces (e.g., doors, windows and walls). An example of the building topology relations will be: "the building 'A' contains a storey 'B', which itself contains spaces 'c' and 'd', that are connected to each other". The environment data is—normally—static, i.e., it does not change often during time. Therefore, this type of data will be inserted once during the system set-up, and it will be referred as *static environment data*.

Chapter 3 presented different BIM models used to represent the environment data. Likewise for the SCN, the WiseNET system is independent of the built environment source, however, it should contain the minimum required data. Furthermore, this chapter will introduce the vocabulary used to represent the built environment data in an ontology form. Moreover, the processing of the *static environment data* will be presented in Chapter 5.

WiseNET ontology and semantic rules The WiseNET ontology is the kernel of the WiseNET system, and is in charge of enabling the interoperability between the data coming from the SCN, the environment data, the information about the events that have occurred and set of semantic rules, in order to deduce new knowledge and detect events/anomalies. The semantic rules are a set *common-sense knowledge* concerning the built environment context, expressed in a rule-like form (i.e, if-then rules) . For example, the knowledge that "*a restricted space should be empty*", can be express in the

rule-like form as "*if there is a space X, and X is restricted, then X should be empty*". Another example will be "*two spaces that are connected contain a common door*", which can be express in a rule-form as "*if there are two spaces X and Y, and X and Y contain the same door D, then X and Y are connected*.

Central API The ontologies are not meant to be used in situations were data is constantly changing over time, which is the case of our system, thus the role of the central API is crucial to enable this type of ontology usage. The central API was developed as a set of web services, and is in charge of the updating/management of the WiseNET ontology, i.e., capturing the knowledge coming from the SCN and inserting it into the ontology, inserting the environment data into the ontology, retrieving the inferred knowledge from the ontology, transferring data to the monitor unit and sending new configurations to the smart cameras. In other words, the central API is the interface between the WiseNET ontology and the incoming and outgoing data.

Furthermore, the central API performs some processing tasks before inserting the data (static and dynamic), as it will be presented in Chapter 5.

Monitor unit The monitor unit's main function is the visualization of the static and dynamic built information; this unit automatically retrieves information and presents it in a graphical manner, for example a people heat map that shows the number of people in the each space, or a space-time graph that shows the location of each person in a period of time. Also, the monitor unit implements many queries to answer questions related to the building management such as: questions related to security e.g., how many people are present in a room? and what is the location of a person?; question related to the space usage like, what is the space more used in a period of time? and a what time a precise space contains more people?; or questions related to the building maintenance e.g, how many times a door have been used? and if a space has been used or not in order to clean it?.

More details on the monitor unit will be presented in Chapter 6.

4.2/ WISENET ONTOLOGY

The WiseNET ontology provides a formal model that aggregates, analysis and repurposing the information coming from a SCN deployed in a built environment. The ontology is defined using the Web Ontology Language (OWL) version 2 [182], and it incorporates a vast corpus of concepts in the domain of an IVS context-aware system. The main functions of the WiseNET ontology are to enable interoperability between the heterogeneous data and to deduce implicit facts from the explicit ones. Thus, allowing to answer queries about the building usage and to perform real-time event/anomaly detection.

This section will firstly present the development process followed to build the WiseNET ontology, and secondly it will present a way to extend the ontology's knowledge by including semantic rules in the form of Semantic Web Rule Language (SWRL) rules.

Notice that all the formulas presented in this section were written using the Description

Logic (DL) notation. Details about this notation can be found in Section 3.3. Moreover, all the terms used in or related to the ontologies have been written using the `typewriter` font.

4.2.1/ ONTOLOGY DEVELOPMENT: FROM REQUIREMENTS TO IMPLEMENTATION

An ontology development methodology refers to a sequence of activities (guidelines) that should be performed to build an ontology. In the last few decades, many methodologies have been proposed. Iqbal et al. [82] and Stadlhofer et al. [161] have performed detailed surveys and a critical analysis on some of these methodologies. To develop the WiseNET ontology, we followed the well-known *101-methodology* presented by Noy and McGuinness [125]. This is an easily understood methodology, which has an iterative nature, covers some critical ontology design issues and it supports reusability (i.e., it makes use of existing ontologies, reducing the ontology development time and effort). The 101-methodology consists of seven steps: (1) determine the domain and scope of the ontology, (2) consider reusing existing ontologies, (3) enumerate terms, (4) define classes and their hierarchies, (5) define properties, (6) define constrains and (7) create instances. Figure 4.2 shows and overview of the different procedures of each step. In the following subsection we will present how each step was performed, which lead to the creation of the WiseNET ontology.

STEP 1. DETERMINE THE DOMAIN AND SCOPE

The determination of an ontology domain/scope consists of finding the different knowledge domains that should be covered by it and its expected use. A way to formalize those requirements is by using a list of Competency Questions (CQs), as introduced by Grunier and Fox [66]. The CQs are a set of question used to characterize an ontology by presenting some tasks that the ontology should be able to represent and answer. Furthermore, CQs *not only* provide an initial direction for the development of an ontology, but also can be used to evaluate if the resulting ontology contains enough information to answer these types of question (as presented in Chapter 6).

Table 4.1 presents the CQs used to determine the requirements of the WiseNET ontology. The questions were obtained from conversations with smart building and indoor surveillance specialists, and from our expectations of a IVS context-aware system. Please note that, for the moment, we are just interested in the CQs (first column of Table 4.1), their categorization concerns the next step.

Judging from the list of CQs, the WiseNET ontology should include information about the built environment, its topology, the different sensor deployed in the building with a focus on visual sensors, and information concerning the space usage.

STEP 2 AND 3. ENUMERATE TERMS AND CONSIDER REUSE

According to the 101-methodology [125], Step 2 is to *consider reusing existing ontologies* and Step 3 is to *enumerate important terms in the ontology*. However, we believe that before searching for external ontologies to reuse, it is important to know in advance the terms that should be included in the ontology itself. In this way, the choice of an external

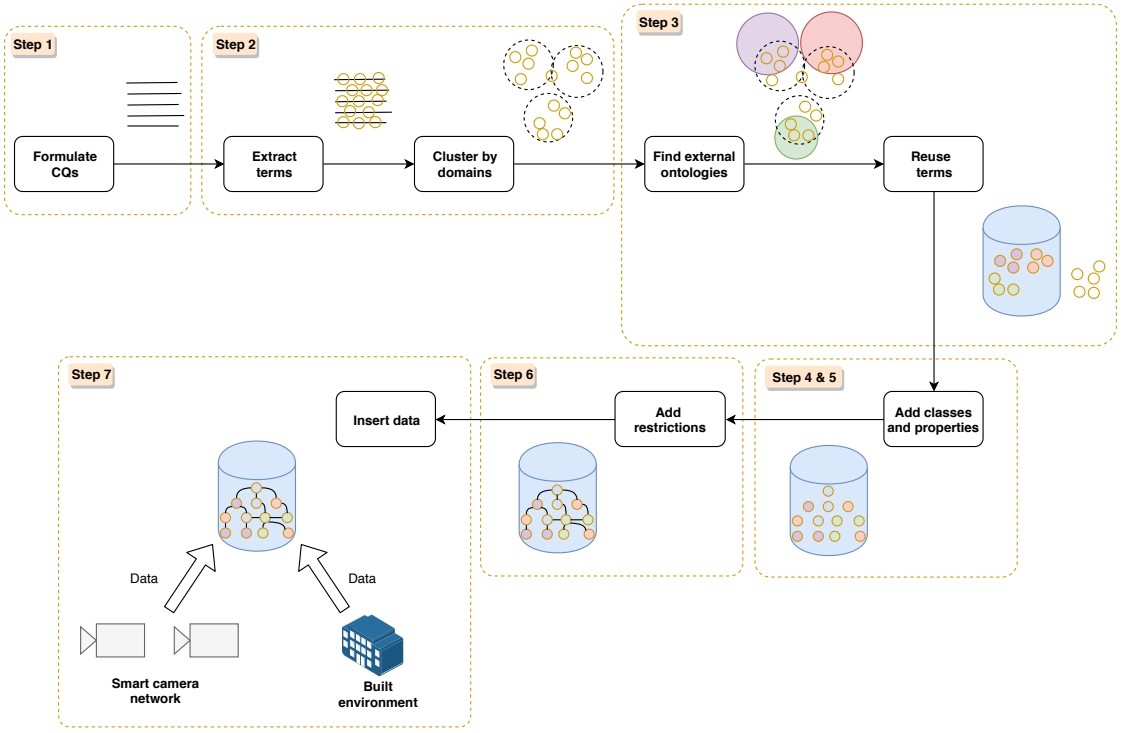


Figure 4.2 – Overview of the steps followed to developed the WiseNET ontology. The orange circles represent the terms of interest extracted from the CQs (competency questions). These terms get colored according to the ontology that will define them. The dotted circles represent the domain categories, while the purple, red and green circles represent the external ontologies that define some of the terms of interest. Finally, the blue shape represents the resulting ontology.

ontology can be based on the inclusion of most necessary terms. Therefore, we propose to *swap steps 2 and 3*, as presented below.

Step 2 - Enumerate terms Most of the terms that should be included in the WiseNET ontology can be extracted from the CQs by considering their main composing elements (e.g., subject, verb and objects), as highlighted in each CQs in Table 4.1. For example, from CQs 1–5, we extracted the terms building, storey, space and door that define classes of objects; the terms contains and are contained that define relationships between objects; and the terms capacity, and functionality that define attributes of a space. To have a better visualization of the terms of interest, the first appearance of each term has been highlighted. Moreover, the CQs (with their corresponding terms) were cluster/categorized according to their domain of interest. They were categorized into three domain axes (see Table 4.1):

- **built environment:** questions regarding the building, its structure, it topology and the different elements that can be found in the building;
- **sensor:** questions concerning the sensors deployed in the environment, their locations, the process they implement and their observations;
- **building usage:** questions associating the building users (person) with the envi-

Table 4.1 – Competency questions used for developing the WiseNET ontology. The questions are categorized by domains, according their essence and the terms that composed them. The first appearance of important terms have been highlighted.

Competency question (CQs)	Domains		
	Built environment	Sensor	Building usage
CQ1. How many stories/spaces does a building contains?	X		
CQ2. How many doors are contained in each storey?	X		
CQ3. How many spaces are contained in each storey?	X		
CQ4. What is the functionality of each space?	X		
CQ5. What is the capacity of each space?	X		
CQ6. What to do if a space's capacity is exceed?	X		
CQ7. Which doors are contained in a space?	X		
CQ8. What are all the spaces connected to a space?	X		
CQ9. Which door connects two spaces?	X		
CQ10. Which types of alarms are in the system?	X		
CQ11. When an alarm should be triggered?	X		
CQ12. Which type of sensors are in the building?	X	X	
CQ13. What are the locations of sensors?	X	X	
CQ14. Which/where are the nearest sensors to a sensor?		X	
CQ15. Is a camera in the same space than another one?	X	X	
CQ16. Which building elements does a camera observes?	X	X	
CQ17. What algorithms are implemented in a smart camera?		X	
CQ18. Which cameras overlap their field of view?		X	
CQ19. What is the IP address of a camera?		X	
CQ20. Are there two cameras observing the same person?		X	X
CQ21. What visual descriptors describe a person?		X	
CQ22. Is there a person which is occluded?		X	X
CQ23. Where is a person located?	X		X
CQ24. Where was a person in the last few minutes?	X		X
CQ25. For how long a person has been in a space?	X		X
CQ26. Where were all the people at a specific time?	X		X
CQ27. How many people are/were in a space?	X		X
CQ28. Is a space empty/occupied?	X		X
CQ29. Is there an intruder?			X
CQ30. Is somebody in a restricted space?	X		X
CQ31. At what time does a person entered/left a space?	X		X
CQ32. From which door a person entered/left a space?	X		X
CQ33. Where does a person stayed the longest time?	X		X
CQ34. What is the most visited space in the building?	X		X
CQ35. Which is the most used door in the building?	X		X
CQ36. At what time there are more people in a space?	X		X
CQ37. How many people entered a space in a period of time?	X		X
CQ38. What are the events that may occur?			X
CQ39. Which events are related?			X
CQ40. What is the location of an event?	X		X
CQ41. Who was involved in an event?			X

ronment by considering the different events that may occur. Some of these questions also consider the notion of time.

Notice that the domain axes are not disjoint, i.e., a question may concern multiple domains (e.g., CQs 12, 16, 24, 31 and 33).

As a result, we have sets of terms grouped by domains of interest. For example built environment-related terms include: building, storey, space, door, space's capacity and

space's connection; sensor-related terms include: type of sensor, sensor's location, camera, observation and the process implemented by them; building usage-related terms include person, person's location, type of event, event's location, event's relation and event's duration.

Step 3 - Consider reuse When developing a new ontology, it is recommended to reuse existing ontologies as much as possible. Thus, one can focus on defining the new and specific knowledge for a particular application. *Ontology reuse* is defined as "*the process in which existing ontological knowledge is used as input to generate new ontologies*" [154]. The reuse of existing ontologies not only saves time but also gives the advantage of using mature and proved ontological resources that have been validated by domain experts and (some) by the World Wide Web Consortium (W3C).

The search for ontologies to reuse starts by considering the results obtained from the CQs, such as the domains that should be covered by the ontology (i.e., built environment, sensor and building usage) and the different terms that each domain should include. With that information, the next step is to search for adequate and matured ontologies that cover most of the terms of each domain. To perform the search, the Linked Open Vocabularies (LOV) dataset was used.[1] LOV is a library containing high-quality and reusable ontologies/vocabularies. LOV allows to focus the search by vocabularies (domains) or by specific terms. During the search, we followed a top-down approach, meaning we first searched for ontologies based on the domains of interest and then checked if the resulting ontologies included most of required terms. We prioritized ontologies that are maintained, recommended by the W3C, and widely used. The search results for each category were as follows:

- For the built environment domain, three ontologies were considered: `rooms`, `ifcowl` and `bot`. The `rooms` ontology [40], is a small ontology that provides a simple set of terms for describing buildings, storeys and rooms. This ontology is missing many important terms, for example terms for describing a building's topology. The `ifcowl` ontology [135], is a semantic representation of the IFC schema (standard for representing building and construction data) [135, 132] (see Section 3.2). This ontology is rich in information, it provides (directly and indirectly) all the required concepts needed to describe the building structure, its topology and the different elements contained in a space. However, in practice, the complexity and the structure of the `ifcowl` makes it hard to use and to extend. For example, the relation `IfcRelAggregates` for stating that a building contains a space, is defined in `ifcowl` as intermediate instance, which raises the complexity unnecessarily [112, 134]. In the first version of the WiseNET ontology we managed to used the `ifcowl` by omitting the intermediate relations and creating simpler ones [112]. However, by the same time of our work (early 2017) the Linked Building Data Community developed the Building Topology Ontology (`bot`) [140]. `bot` is a simple ontology covering the core concepts for describing a building, its topology and its elements. This ontology provides all the required built environment terms, while using a simple schema which makes it easy to extend. Consequently, we decided to use `bot` instead of `ifcowl`.
- Regarding the sensor domain, three ontologies were considered: `saref` and

[1] https://lov.linkeddata.es/dataset/lov

Table 4.2 – Prefixes and namespaces used in WiseNET ontology and in this document.

Prefix	Namespaces	Description
bot	https://w3id.org/bot#	The building topology ontology
event	http://purl.org/NET/c4dm/event.owl#	The event ontology
foaf	http://xmlns.com/foaf/0.1/	The friend of a friend vocabulary
geo	http://www.w3.org/2003/01/geo/wgs84_pos#	The geo positioning vocabulary
owl	http://www.w3.org/2002/07/owl#	The OWL 2 schema vocabulary
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	The RDF concepts vocabulary
rdfs	http://www.w3.org/2000/01/rdf-schema#	The RDF schema vocabulary
sosa	http://www.w3.org/ns/sosa/	The sensor, observation, sample, and actuator ontology
ssn	http://www.w3.org/ns/ssn/	Semantic sensor network ontology
time	http://www.w3.org/2006/time#	OWL-Time ontology
wisenet	http://ontology.wisenet.checksem.fr#	The WiseNET ontology
wni	http://ontology.wisenet.checksem.fr/inst#	WiseNET instances
xml	http://www.w3.org/XML/1998/namespace	XML specification
xsd	http://www.w3.org/2001/XMLSchema#	XML schema definition

ssn/sosa. The Smart Appliances REFerence (saref) ontology [43], provides terms to specify devices in smart environments. saref focus on the device's functions, services, states and commands. This ontology could be easily extended to include the required sensor terms. The Sensor, Observation, Sample, and Actuator (sosa) and Smart Sensor Network (ssn) ontologies [4], provides concepts to describe sensors, their observations, their procedures, their results, etc. This ontology provides most of the required terms, and can be easily extended to include the rest of them. Both ontologies—saref and ssn/sosa—could be reused in the WiseNET ontology however, the ssn/sosa is a W3C recommendation and its widely used, therefore we decided to use it as the basis of our sensor domain.

- For the building usage domain we used the well-known event ontology [139], which deals with the notion of events and their different properties such as location, time, agents (people involve), factors and products. This is an easily extendable ontology that provides most of the vocabulary required for describing activities and events that may happen in a built environment. Moreover, the event ontology imports the owl-time ontology which is a W3C recommendation for describing the temporal properties of resources [72]. Also, the event ontology makes reference to the terms Person and Agent from the Friend Of A Friend (foaf) ontology [20], terms which are also important to us.

Figure 4.3 shows the external classes and properties reused by the WiseNET ontology. In total, the WiseNET ontology reuses 45 terms from 6 external ontologies, from which 15 are classes, 28 are object properties and 2 are data properties. The suggested prefixes for the external ontologies and their namespaces Uniform Resource Identifiers (URIs) are presented in Table 4.2. As a reminder, the namespace URI is a unique global identifier of the ontology resource, while the prefix name is used as an abbreviation of the namespace. For example, instead of writing https://w3id.org/bot#Building, we can write bot:Building because bot is the prefix for the namespace https://w3id.org/bot#. Moreover, because we are inside the WiseNET ontology, the default prefix corresponds to wisenet and can be written by the *empty prefix* ":". In the rest of this chapter, the prefixes will be used to refer the external ontologies and the empty prefix to refer the terms defined in the WiseNET ontology.

There are two ways of reusing external ontologies inside a base-ontology (in our case

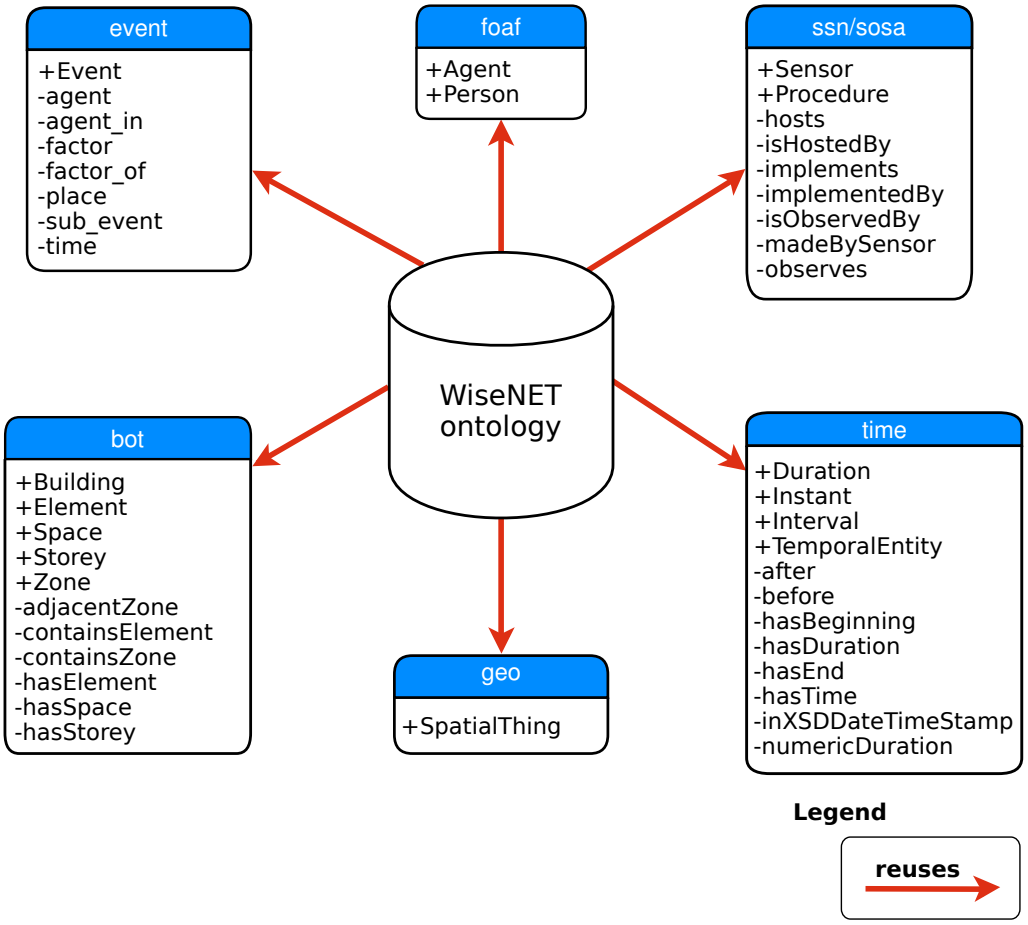


Figure 4.3 – External classes and properties reused by the WiseNET ontology. Classes are marked with (+) and properties (object and data properties) are mark with (-).

WiseNET): *complete import* or *selective import*. The first solution—complete import—consists of including the whole set of statements (classes, properties, axioms, individuals, etc) defined in the imported ontology. This is achieved by using the `owl:imports` property in OWL. The advantage of this solution is that all the terms and axioms of the imported ontology (and of the ontologies that themselves they import) are taken into consideration while performing inference, thus resulting in a rich model. However, this richness is often unnecessary, and it may cause many drawbacks in the base-ontology, such as: increasing its size; decreasing its comprehension, thus making more difficult its maintenance and its flexibility (i.e., to be reused); and rising its reasoning complexity and thus reasoning time, by including more complex DL constructors and by increasing the ontology size [16]. The second solution—selective import—consists in only importing the terms of interest, not the complete ontology that defined them. This is done by referring to the terms using their namespaces URI, which points to their definition in the external ontology. The advantages of this solution is that the base-ontology stays understandable and there is no unnecessary explosion in size and complexity. Moreover, this solution is practical if only some few terms of the external ontologies are required, which is the case with most of our external ontologies. For example, from the `ssn/sosa` ontologies we only need 2 classes out of 19 and 7 properties out of 38; from the `foaf` vocabulary, we only need 2 classes out of 23 classes and 71 properties. Table 4.3 presents a comparison of both solutions. The comparison was performed by considering the WiseNET classes and

Table 4.3 – Comparison between methods for reusing ontologies, by *importing the complete ontology* or just *importing specific terms*. The WiseNET terms presented in Steps 4 and 5, were also included in both experiments.

	Complete import	**Selective import**
#Classes	85	27
#Object properties	153	40
#Data properties	74	17
#Axioms	3049	609
DL expressivity	$\mathcal{SROIN}(\mathcal{D})$	$\mathcal{SRI}(\mathcal{D})$
Reasoning time (ms)	81.6	18.1

properties (that will be presented in the next steps) and adding them two sets of extra knowledge: (1) the complete axioms of the 6 external ontologies (*complete import*), or (2) only the necessary external terms (*selective import*). The experiment was done using the Protégé ontology editor [162] and the Pellet reasoner [155]. The reasoning time was obtained by executing the reasoner 10 times and performing an average. The results are as expected, the selective import considers a much lower set of axioms and has a lower DL-expressivity, which translate in a lower reasoning time—around 4.5 times faster than the complete import. **Based on those results and in our application where the goal is to obtain, as close as possible, real-time reasoning, we decided to use the second solution—selective import**.

The reused terms will not provide enough information to answer the competency questions defined in Step 1. More specific terms related to our precise application— intelligent visual system in a built environment—are still missing as shown in the Step 3 in Figure 4.2 by the terms outside the blue ontology. For example, terms that define the different types of events that can occur, that define the capacity of a space, the different elements that can be contained in the environment, the connectivity between cameras, etc. Hence, in the following two steps, we will define the missing terms and organize them in a hierarchical way.

STEP 4 AND 5. DEFINE CLASSES AND PROPERTIES

According to the 101-methodology [125], the Step 4 consists in defining the missing classes and organizing them into a taxonomical hierarchy. While, the Step 5 consists in defining the missing properties that will allow us to characterize the classes according to our needs. Both steps are closely related, therefore we decide to develop them in the same section.

Classes are collections of objects with similar properties. Thus, the terms that describe objects as an independent existence, rather than characterize the objects, were defined as *classes*. In the other hand, the terms that characterize an object by describing its attributes were defined as *properties*. For example, an instance of the class `Space` can be characterize by properties stating if it *is occupied*, if it *is restricted*, its *maximal capacity*, the *number of people* in the space, etc; another example, an instance of the class `Detection` can be characterize by properties stating the *place* where it occur, the *time*, if it was done *in region of interest*, etc.

There are 42 new terms defined in WiseNET ontology, as follows:

Classes `:Alarm`, `:BoundingBox`, `:Detection`, `:Door`, `:FieldOfView`, `:ImageProcessing`, `:InstantEvent`, `:IntervalEvent`, `:PersonDetection`, `:PersonInSpace`, `:RegionOfInterest`, `:SmartCamera`.

Object Properties `:appearsIn`, `:containsPerson`, `:hasAlarm`, `:hasBoundingBox`, `:hasFieldOfView`, `:hasNearbySensor`, `:inRegionOfInterest`, `:isFieldOfViewOf`, `:isSubEventOf`, `:isRelatedTo`, `:overlaps`, `:personLocation`, `:represents`, `:shows`.

Data Properties `:dimension`, `:ipAddress`, `:isEntryViolation`, `:isEventOpen`, `:isIntruder`, `isNoise`, `:isOccluded`, `:isOccupied`, `:isRestricted`, `:maxCapacity`, `:numberOfPeople`, `:startRecording`, `:triggeredByIntruder`, `:triggeredByMaxCapacity`, `:visualDescriptors`, `:xywh`.

Based on the standards, class names are written in *majuscule* while property names in *minuscule*.

After defining the missing terms, the classes and properties were organized in a hierarchical manner by using the top-down approach presented by Noy's et al. [125], that consists in starting from a general class/properties and then specialize it in sub-classes/sub-properties. One way of doing this for the classes is by asking *if by being an instance of class B, will necessarily (i.e., by definition) mean being an instance of class A*. In other words, if B is a sub-class of A ($B \sqsubseteq A$) then every instance of B will also be an instance of A, thus the class B represents a concept that is a specialized version A. Based on this, we started by specializing (i.e., extending) the reused terms as follows:

$$\texttt{:IntervalEvent} \sqsubseteq \texttt{event:Event}, \tag{4.1}$$

$$\texttt{:InstantEvent} \sqsubseteq \texttt{event:Event}, \tag{4.2}$$

$$\texttt{:ImageProcessing} \sqsubseteq \texttt{sosa:Procedure}, \tag{4.3}$$

$$\texttt{:Door} \sqsubseteq \texttt{bot:Element}, \tag{4.4}$$

$$\texttt{:SmartCamera} \sqsubseteq \texttt{sosa:Sensor}. \tag{4.5}$$

Then, we specialized the newly defined classes as follows:

$$\texttt{:RegionOfInterest} \sqsubseteq \texttt{:BoundingBox} \tag{4.6}$$

$$\texttt{:PersonDetections} \sqsubseteq \texttt{:ImageProcessing} \tag{4.7}$$

$$\texttt{:Detection} \sqsubseteq \texttt{:InstantEvent}, \tag{4.8}$$

$$\texttt{:PersonInSpace} \sqsubseteq \texttt{:IntervalEvent}. \tag{4.9}$$

Furthermore, we performed some *containment semantic relation* (i.e., a type of mapping specifications) between the external terms. Specifically, we stated that the element in one ontology represents a more specific aspect of the world than the element in the other ontology, as follows:

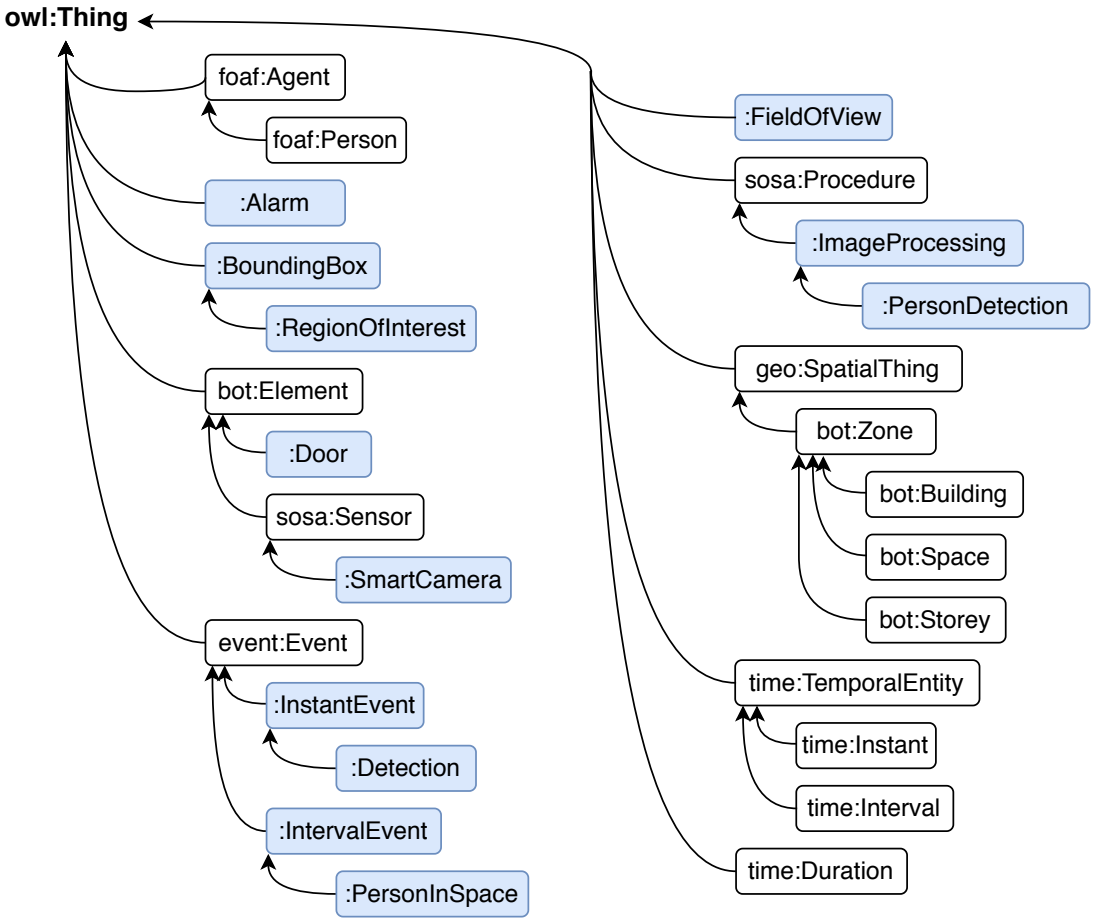


Figure 4.4 – WiseNET class hierarchy. The arrow represents the "sub-class of" relationship. The `owl:Thing` class is the root of all the classes.

$$\texttt{bot:Zone} \sqsubseteq \texttt{geo:SpatialThing}, \tag{4.10}$$

$$\texttt{sosa:Sensor} \sqsubseteq \texttt{bot:Element}, \tag{4.11}$$

$$\texttt{sosa:hosts} \sqsubseteq \texttt{bot:containsElement}, \tag{4.12}$$

$$\texttt{event:time} \sqsubseteq \texttt{time:hasTime}. \tag{4.13}$$

The complete terminology of the WiseNET ontology—classes, object properties and data properties—ordered in a hierarchical manner is presented in Figures 4.4 and 4.5. Notice that for brevity we did not state above all the hierarchical formulas defined in the external ontologies (e.g., `time:Instant` $\sqsubseteq$ `time:TemporalEntity` and `bot:hasSpace` $\sqsubseteq$ `bot:containsZone`), however they were considered as it can be observed in the figures. The terms defined in WiseNET, allow us to complete the information from the different domains, to describe attributes of instances according to our needs and to relate (*i.e.,* bridge) the different domains. To create those "bridges" we need to constraint the class expressions and the properties, as shown in the next step.

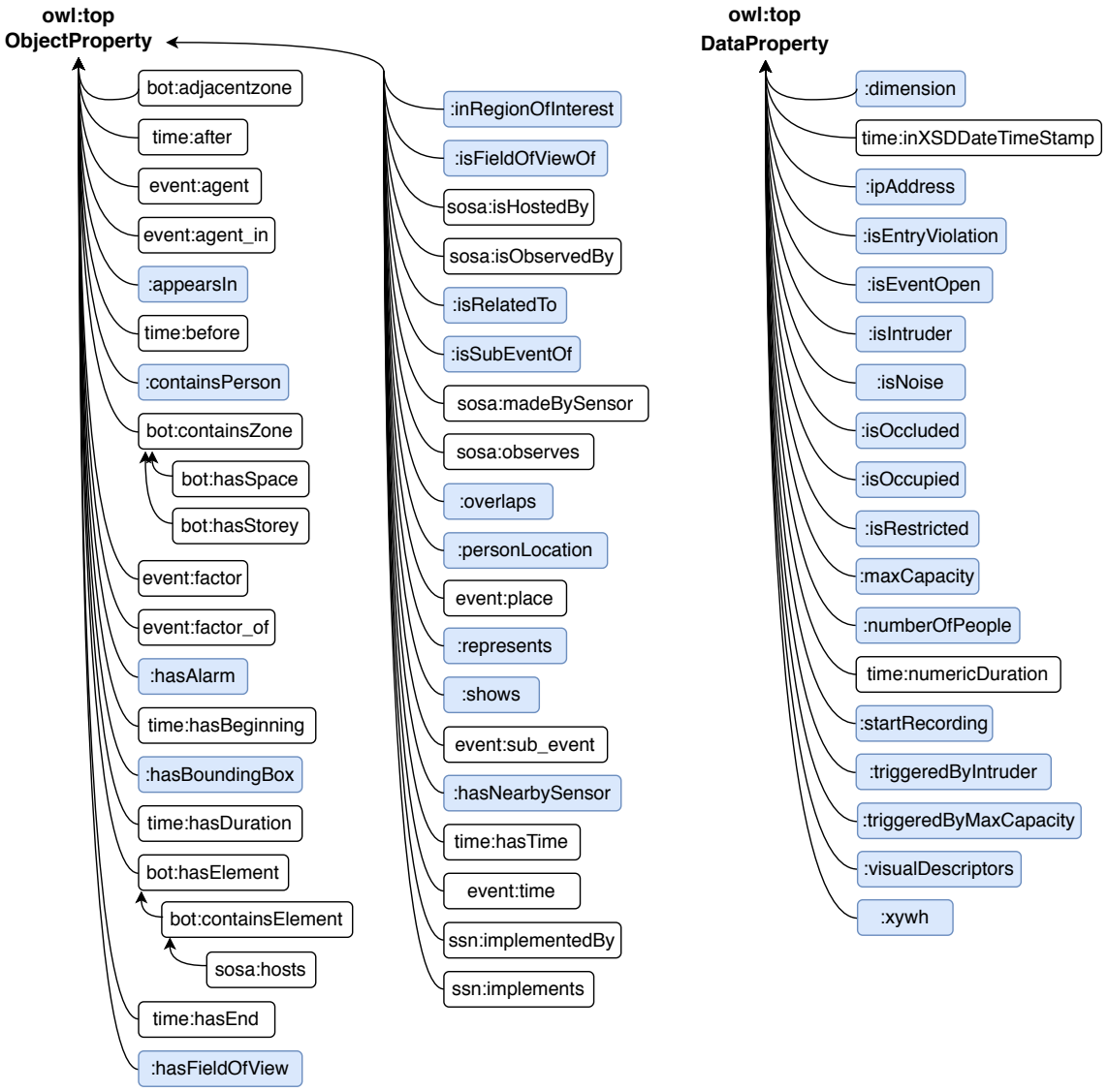


Figure 4.5 – WiseNET object and data properties. The arrow represents the "sub-property of" relationship. The properties `owl:topObjectProperty` and `owl:topDataProperty` are the root of all the object properties and data properties, respectively.

Remark. *A detailed description of each term composing the WiseNET ontology can be found in the Appendix A.*

STEP 6. DEFINE CONSTRAINS

After having the complete set of terms needed in the WiseNET ontology, some constraints/restrictions on the use of properties need to be defined. The WiseNET ontology is defined in OWL 2, thus the type of constraints depend on the DL constructors used. They can be constraints about the value type (i.e., data types like boolean, string, integer, etc.), the classes allowed to use the properties (i.e., property's domain and range), the number of values a property can have (i.e., cardinality), the characteristic of a property

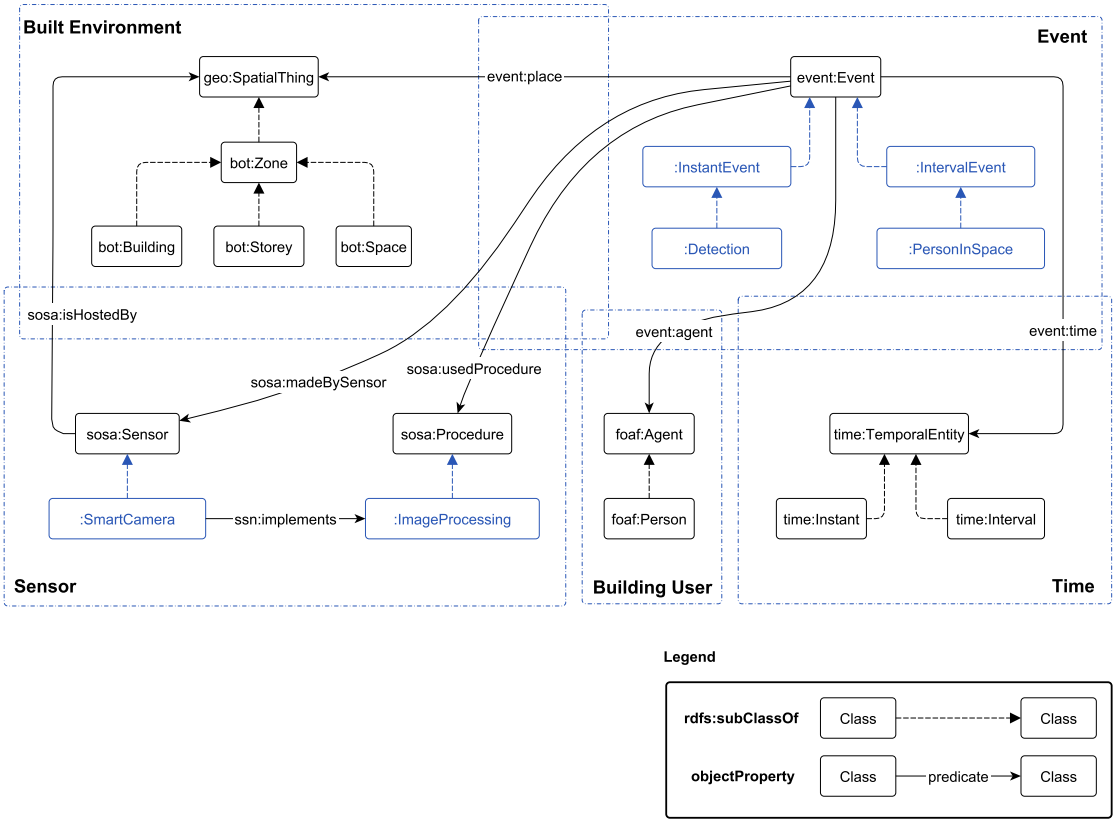


Figure 4.6 – General view of the WiseNET ontology, showing the different domains involved.

(i.e., transitive, inverse, symmetric, etc.), the inclusion of a property which is the combination of multiple properties (i.e., chain rules), etc (see Section 3.3, specifically Tables 3.2, 3.3 and 3.4 for more details). The constraints define the intended (and allowed) use of the classes and properties and will define the semantic relations between the classes of different domains.

Figure 4.6 shows a general view of the WiseNET ontology showing the main domains that compose it—built environment, sensor, event, building user and time. For each domain, an overview of the main classes, properties, and the property's domain/range restrictions will be presented below. All the figures presented in this section were created based on the Graffoo notation [53]. In the figures, the elements (i.e., nodes and edges) in blue denotes our extensions of that domain.

Built environment An overview of the core classes and properties related to built environment domain is shown in Figure 4.7. The core concept is `bot:Zone` which is a sub-class of `geo:SpatialThing` and is divided into three sub-classes `bot:Building`, `bot:Storey` and `bot:Space`, these sub-classes will share the same properties as `bot:Zone`. An instance of `bot:Zone` is characterized by the following attributes: if it is a restricted zone or not (specified by the relation `:isRestricted`); if it is occupied or not (`:isOccupied`); by the number of people that it contains (`:numberOfPeople`); the maximum number of people it can contains `:maxCapacity`; the instances of people

that it contains (:containsPerson), notice that this property links the built environment and the building user domains; if it has an alarm (:hasAlarm) that can be triggered if there is somebody in a restricted zone (:triggeredByIntruder) or if the maximum number of people allowed has been exceeded (:triggeredByMaxCapacity); the other zones it contains (bot:containsZone); and the building elements it contains (bot:hasElement). Notice that the property sosa:host is specifically used to relate a bot:Space with a sosa:Sensor, thus linking both domains.

Moreover, some of the properties incorporate extra constraints. The property :containsPerson has as *inverse* the property :personLocation, meaning that, if <zoneX> (instance of bot:Zone) *contains person* <personA> (instance of foaf:Person), then it can be inferred that <personA> has *person location* <zoneX>, and vice-versa as presented in Eq. 4.14. In the same manner, sosa:hosts has inverse sosa:isHostedBy (Eq. 4.15). In addition, the property bot:containsZone is *transitive* as stated in Eq. 4.16, for example, if <buildingX> *contains zone* <storeyY> and <storeyY> *contains zone* <spaceZ>, then it can be inferred that <buildingX> also *contains zone* <spaceZ>. As well, the property bot:adjacentZone is *symmetric* as stated in Eq. 4.17, meaning that, if <spaceX> is an *adjacent zone* of <spaceY>, then it can be inferred that <spaceY> is also an *adjacent zone* of <spaceX>. Furthermore, complex role inclusions (i.e., property chains of the forms $s \circ p \sqsubseteq p$, $p \circ s \sqsubseteq p$ and $r \circ s \sqsubseteq p$), were also incorporated. The Eq. 4.18 states that: if <zoneX> *contains zone* <zoneY> and <zoneY> *contains person* <personA>, then <zoneX> also *contains person* <personA>. In the same manner, the Eq. 4.19, taken from the bot ontology [140], states that: if <zoneX> *contains zone* <zoneY> and <zoneY> *has element* <elementD>, then <zoneX> also *has element* <elementD>.

$$\begin{aligned} \texttt{:personLocation} &\equiv \texttt{:containsPerson}^{-} \\ \texttt{:containsPerson} &\equiv \texttt{:personLocation}^{-} \end{aligned} \tag{4.14}$$

$$\begin{aligned} \texttt{sosa:hosts} &\equiv \texttt{sosa:isHostedBy}^{-} \\ \texttt{sosa:isHostedBy} &\equiv \texttt{sosa:hosts}^{-} \end{aligned} \tag{4.15}$$

$$\texttt{Trans(bot:containsZone)} \tag{4.16}$$

$$\texttt{Sym(bot:adjacentZone)} \tag{4.17}$$

$$\texttt{bot:containsZone} \circ \texttt{:containsPerson} \sqsubseteq \texttt{:containsPerson} \tag{4.18}$$

$$\texttt{bot:containsZone} \circ \texttt{bot:hasElement} \sqsubseteq \texttt{bot:hasElement} \tag{4.19}$$

Sensor Figure 4.8 presents the core classes and properties related to the sensor domain. The core concepts are sosa:Sensor and its sub-class :SmartCamera. An instance of bot:Sensor is characterized by the space that host it (specified by the relation sosa:isHostedBy), the sensor it has close by (:hasNearbySensor) and by its IP address if any (:ipAddress). An instance of :SmartCamera is characterized by its recording flag (:startRecording); its field of view (FOV) (:hasFieldOfView); by what it observes (sosa:observes) that can be a person (foaf:Person) or a particular region of interest (ROI) in the image (:RegionOfInterest); and by the

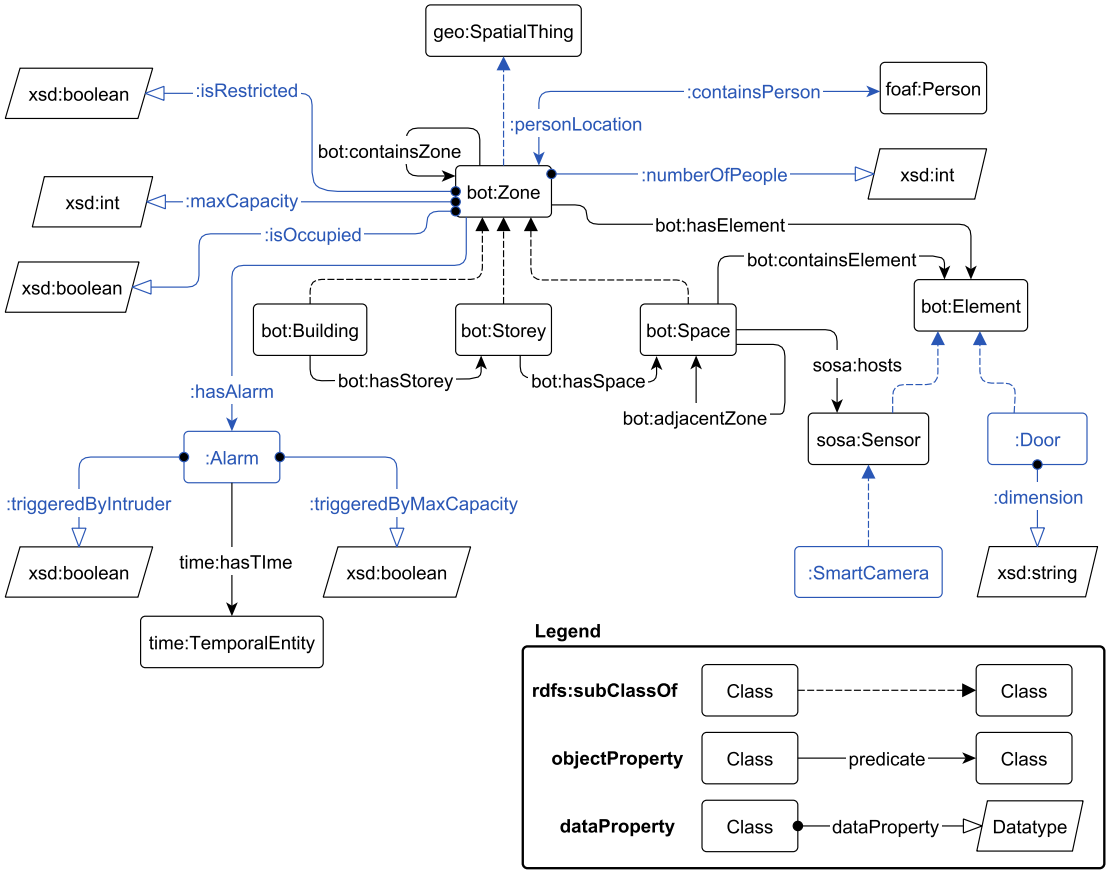


Figure 4.7 – Overview of the WiseNET classes and properties in the perspective of the built environment domain.

image processing algorithm (`:ImageProcessing`) it implements (`ssn:implements`). We focus in person detection algorithms (`:PersonDetection`), specifically in four different algorithms—`HOG_SVM`, `SSD`, `YOLOv3` and `groundTruth`[2]—inserted as instances (see Section 2.3.1 for more details in the algorithms). Moreover, the class `:RegionOfInterest` is a sub-class of `:BoundingBox`, which represents (`:represents`) a physical building element like a door. An instances of `:BoundingBox` is characterized by its coordinates (`:xywh`), defined by it's top-left point coordinate (x,y), width (w) and height (h). Furthermore, a FOV may overlap with other FOVs (`overlaps`), and it may show a person or a ROI (`:shows`).

Moreover, the property `:shows` has *inverse* `:appearsIn` (Eq. 4.20); in the same way, `:hasFieldOfView` has *inverse* `:isFieldOfViewOf` (Eq. 4.21), `sosa:observes` has *inverse* `sosa:isObservedBy` (Eq. 4.22) and `ssn:implements` has *inverse* `ssn:isImplementedBy` (Eq. 4.23). In addition, the properties `:overlaps` and `:hasNearbySensor` are *symmetric* (Eqs 4.24 and 4.25 respectively). Lastly, complex role inclusions were stated for the properties `sosa:observes` and `:shows` as presented in Eqs. 4.26 and 4.27.

[2]the person detection ground-truth is not really an algorithm but we will consider it as a "perfect" detector

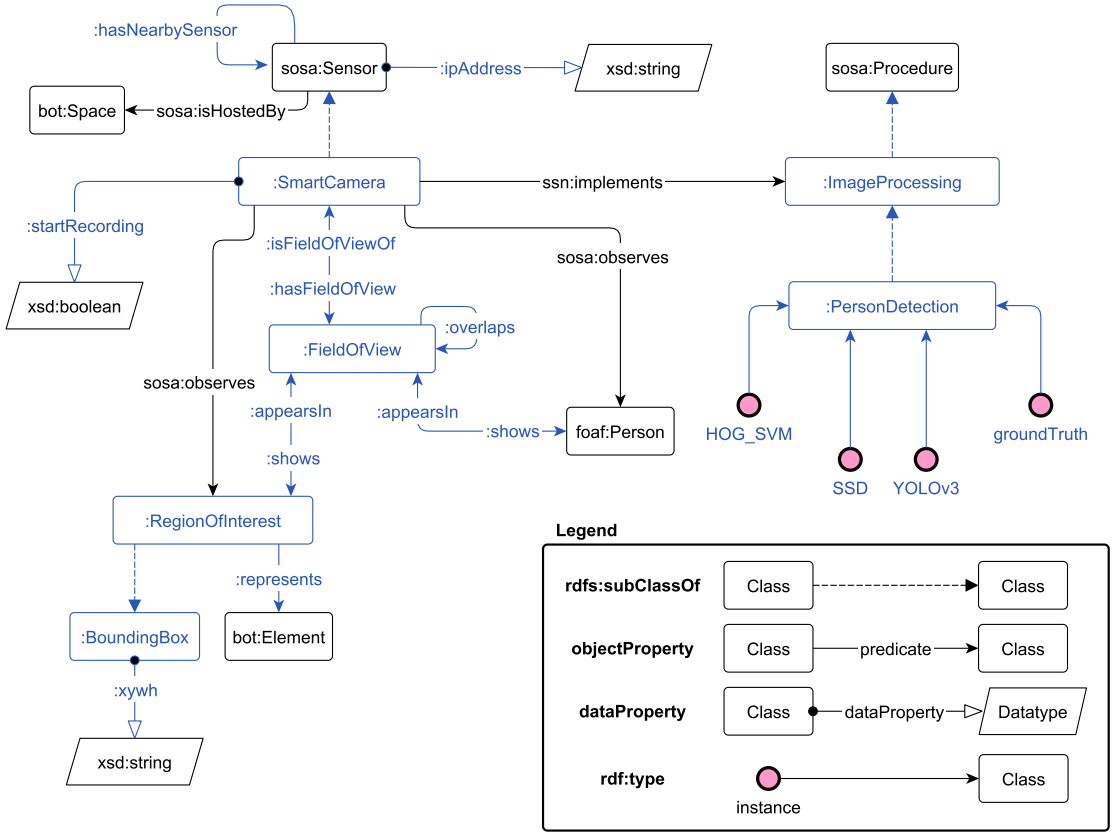


Figure 4.8 – Overview of the WiseNET classes and properties in the perspective of the sensor domain.

$$:\text{shows} \equiv :\text{appearsIn}^{-} \qquad :\text{appearsIn} \equiv :\text{shows}^{-} \tag{4.20}$$

$$:\text{hasFieldOfView} \equiv :\text{isFieldOfViewOf}^{-} \qquad :\text{isFieldOfViewOf} \equiv :\text{hasFieldOfView}^{-} \tag{4.21}$$

$$\text{sosa:observes} \equiv \text{sosa:isObservedBy}^{-} \qquad \text{sosa:isObservedBy} \equiv \text{sosa:observe}^{-} \tag{4.22}$$

$$\text{ssn:implements} \equiv \text{ssn:implementedBy}^{-} \qquad \text{ssn:implementedBy} \equiv \text{ssn:implements}^{-} \tag{4.23}$$

$$\text{Sym}(:\text{overlaps}) \tag{4.24}$$

$$\text{Sym}(:\text{hasNearBy}) \tag{4.25}$$

$$:\text{hasFieldOfView} \circ :\text{shows} \sqsubseteq \text{sosa:observes} \tag{4.26}$$

$$:\text{shows} \circ :\text{represents} \sqsubseteq :\text{shows} \tag{4.27}$$

Event An overview of the core classes and properties related to event domain is shown in Figure 4.9. The core concepts are `event:Event` and its subclasses `:InstantEvent` and `:IntervalEvent`. To start, we constraint the class `event:Event` by stating that is the union of `:InstantEvent` and `:IntervalEvent`, as presented in Eq. 4.28. An instance of `event:Event` is characterized by the person involve in it (specified by the relation `event:agent`), this relation links the event and the building user domains; the place were it occur (`event:place`), this relation links the event and the built environment domain; the sensor that created the event (`sosa:madeBySensor`), this relation links the event and the sensor domains; the procedure used to create the event `event:factor`, this relation also links the event domain with the sensor domain; and the time when it occurred (`event:time`). The type of time instance will define the type of event. The class `:InstantEvent` is a kind of event which the property `event:time` can relate *only to* `time:Instant` class, while the class `:IntervalEvent` can relate *only to* `time:Interval` class, as stated in Eqs. 4.29 and 4.30. An instance of `:InstantEvent` can compose an `:IntervalEvent` (stated by `:isSubEventOf`), and inversely an `:IntervalEvent` can be composed by an `:InstantEvent` (`event:sub_event`). Thus, the class `:Detection` is a kind of `:Event` that occurs in a specific point in time/space and which normally is the output of sensing device. An instance of `:Detection` can be an entry violation or not (`:isEntryViolation`), can be performed around a ROI (`:inRegionOfInterest`) for example around a door, and can have a bounding box (`:hasBoundingBox`). The class `:PersonInSpace` is a container of `:Detection` instances relating a specific person with a specific space during a period of time. An instance of `:PersonInSpace` might be open or closed (`:isEventOpen`), if it is not open it means that the person involved left the space hence no more detections can be attached to it. Also, a `:PersonInSpace` instance can be created by noise detection, thus the boolean property `isNoise` can be used to distinguish them. Moreover, multiple instances of `:PersonInSpace` can be related to each other (`:isRelatedTo`) to constitute the space/time trajectory of a person, as it will be shown in Chapter 6.

Furthermore, the property `event:agent` has *inverse* `event:agent_in` (Eq. 4.31); in the same way, `event:factor` has *inverse* `event:factor_of` (Eq. 4.32) and `event:sub_event` has *inverse* `:isSubEventOf`. In addition, the property `:isRelatedTo` is *transitive* and *symmetric*, meaning that if `<intervalEventA>` (instance of `:IntervalEvent`) *is related to* `<intervalEventB>` and `<intervalEventB>` *is related to* `<intervalEventC>`, then it can be inferred that: `<intervalEventA>` is also *related to* `<intervalEventC>`, by *transitivity*; and that `<intervalEventB>` is also *related to* `<intervalEventA>`, as well as `<intervalEventC>` is also *related to* `<eventB>`, by *symmetry*.

$$\texttt{event:Event} \equiv \texttt{:InstantEvent} \sqcup \texttt{:IntervalEvent} \tag{4.28}$$

$$\texttt{:InstantEvent} \sqsubseteq \texttt{event:Event} \sqcap \forall\, \texttt{event:time.time:Instant} \tag{4.29}$$

$$\texttt{:IntervalEvent} \sqsubseteq \texttt{event:Event} \sqcap \forall\, \texttt{event:time.time:Interval} \tag{4.30}$$

$$\begin{aligned} \texttt{event:agent} &\equiv \texttt{event:agent\_in}^{-} \\ \texttt{event:agent\_in} &\equiv \texttt{event:agent}^{-} \end{aligned} \tag{4.31}$$

$$\begin{aligned} \texttt{event:factor} &\equiv \texttt{event:factor\_of}^{-} \\ \texttt{event:factor\_of} &\equiv \texttt{event:factor}^{-} \end{aligned} \tag{4.32}$$

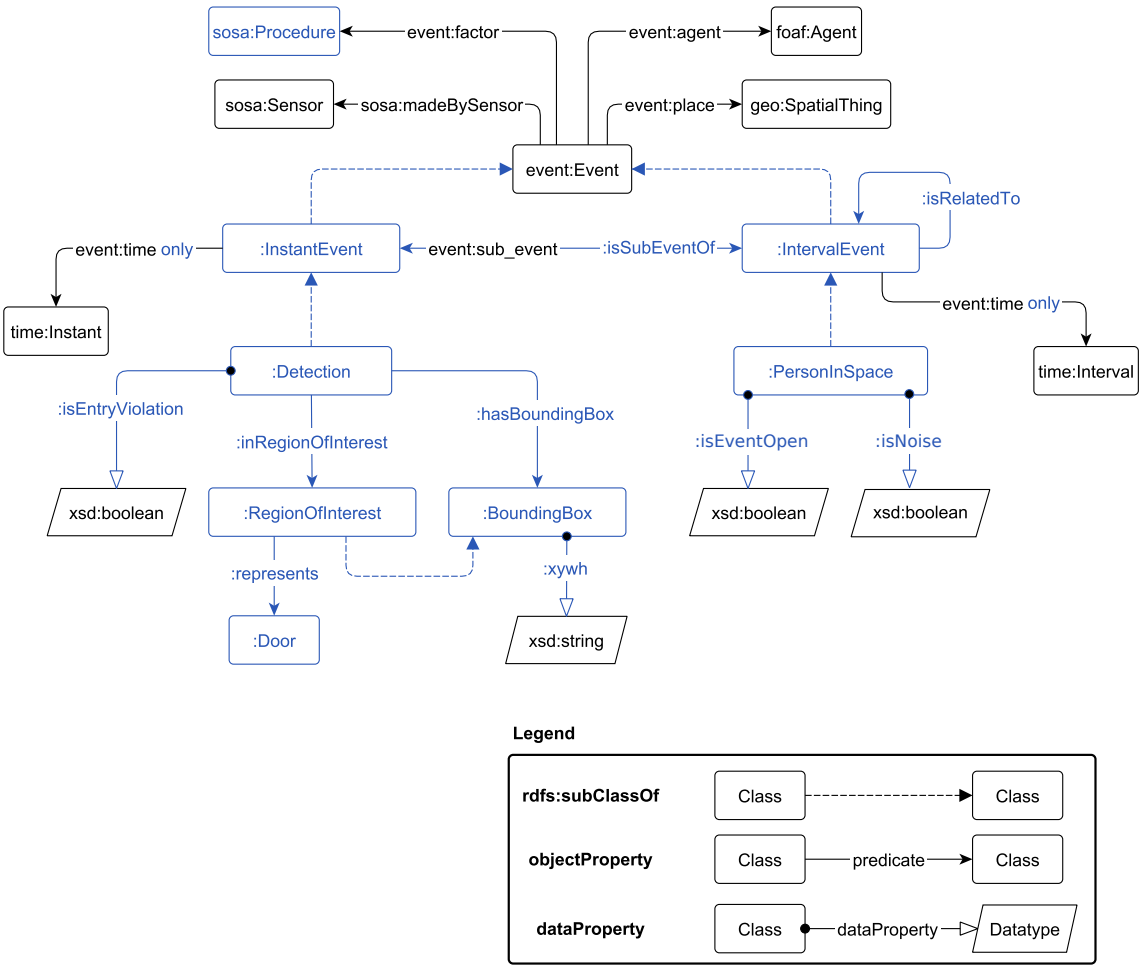


Figure 4.9 – Overview of the WiseNET classes and properties in the perspective of the event domain.

$$\texttt{event:sub\_event} \equiv \texttt{:isSubEventOf}^{-} \qquad \texttt{:isSubEventOf} \equiv \texttt{event:sub\_event}^{-} \tag{4.33}$$

$$\texttt{Sym(:isRelatedTo)} \tag{4.34}$$

$$\texttt{Trans(:isRelatedTo)} \tag{4.35}$$

Building user An overview of the core classes and properties related to building user domain is shown in Figure 4.10. The core concept is `foaf:Person`. An instance of `foaf:Person` is characterized by its location (specified by the relation `:personLocation`), by an array of visual features that describes it (`:visualDescriptors`), if it is occluded or not (`:isOccluded`) and if it is a intruder or not (`:isIntruder`). As previously stated, the relations `:personLocation` and its *inverse* `:containsPerson` (Eq. 4.14), link the building user domain with the built environment domain.

Time An overview of the core classes and properties related to *built environment* domain is shown in Figure 4.11. As it can be observed in the schema, this domain

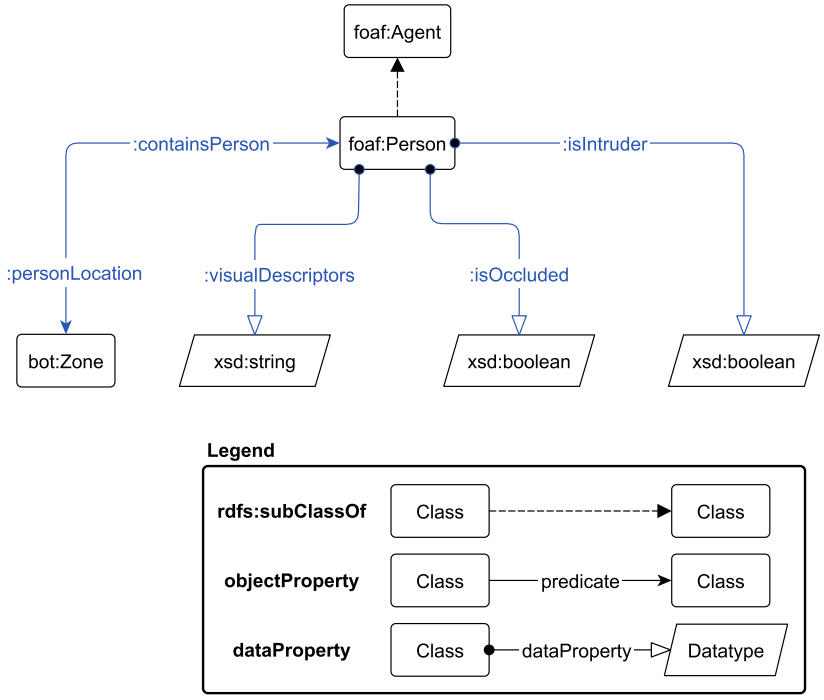


Figure 4.10 – Overview of the WiseNET classes and properties in the perspective of the building user domain.

was not extended in any way, i.e., all the restrictions were taken from the `time` ontology [72]. The core concepts are the `time:Instant` and `time:Interval`, which are sub-classes of `time:TemporalEntity`. An instance of `time:Instant` has a timestamp that states the date and time of day when it occur (`time:inXSDDateTimeStamp`). An instance of `time:Interval` has a starting time-instant (`time:hasBeginning`), an end time-instant (`time:hasEnd`), and a duration (`time:hasDuration`) which itself has a value (`time:numericDuration`). Moreover, the class `time:TemporalEntity` is the union of its sub-classes, as presented in Eq. 4.36. Furthermore, an instance of `time:TemporalEntity` can occur before (`time:before`) or after (`time:after`) another instance. As expected, the properties `time:before` and `time:after` are *inverse*, and *transitive*, as stated in Eqs. 4.37, 4.38 and 4.39 respectively.

$$\texttt{time:TemporalEntity} \equiv \texttt{time:Instant} \sqcup \texttt{time:Interval} \tag{4.36}$$

$$\begin{aligned} \texttt{time:before} &\equiv \texttt{time:after}^{-} \\ \texttt{time:after} &\equiv \texttt{time:before}^{-} \end{aligned} \tag{4.37}$$

$$\texttt{Trans(time:after)} \tag{4.38}$$

$$\texttt{Trans(time:before)} \tag{4.39}$$

As previously stated, the type of constraints used define the ontology's DL language—i.e., it's expressivity level. Based on the constraints presented above, the WiseNET ontology is defined in the $\mathcal{SRI}(\mathcal{D})$ DL language. A definition of the $\mathcal{SRI}(\mathcal{D})$ constructors and the corresponding formulas that used those constructors are presented in the Table 4.4. Even though stronger constraints could be added to the WiseNET ontology (e.g., qualified number restrictions constraint ($\mathcal{Q}$ constructor) stating that all the spaces have to have at least

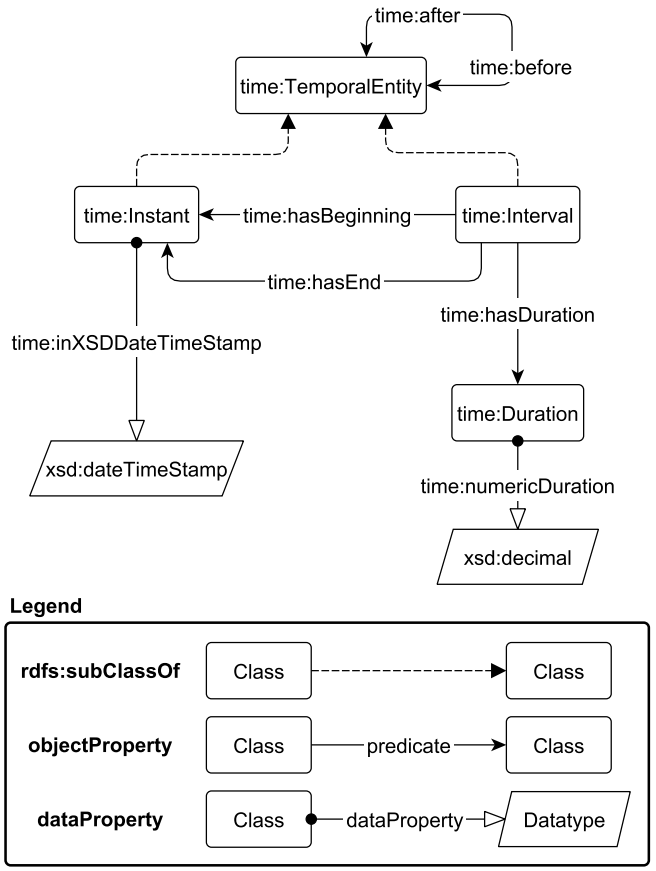


Figure 4.11 – Overview of the WiseNET classes and properties in the perspective of the time domain.

one door, or that an smart camera can have only one field of view), this will incurred in a higher computational cost during inference, which is undesirable in our application. Moreover, $\mathcal{SRI}(\mathcal{D})$ allowed us to represents all the constraints that we considered essential while providing a suitable computational cost. Furthermore, Horrocks et al. [75], presented a tableau decision procedure for $\mathcal{SRI}(\mathcal{D})$ that solves the ontology consistency problem and allows the use of reasoning services, thus demonstrating the decidability of $\mathcal{SRI}(\mathcal{D})$.

In the last three steps—4, 5 and 6—we have defined a set of axioms (formulas) describing the structure of the WiseNET domain. This set of structural axioms represents the terminological knowledge called TBox. The TBox's constraints and characteristics allow us to link concepts between the different domains, thus attaining interoperability in a semantic level. The next step concerns the assertion of axioms describing a concrete situations (i.e., instantiation of classes and properties). The set of assertional knowledge is called ABox.

Step 7. Create instances

The last step of the 101-methodology [125] consist in inserting instances of classes and creating relations between the different instances by using the properties. For example,

Table 4.4 – Definition of $\mathcal{SRI}(\mathcal{D})$ constructors. The equations in bold corresponds to the examples given in OWL functional syntax [121]. The $\mathcal{ALC}$ constructors contains the basic concept constructor such as intersection, negation and universal/existential restrictions, for more details refer to Section 3.3.1.

Constructor	Definition	Equations	Example in OWL functional syntax
$\mathcal{S}$	$\mathcal{ALC}$ + Transitivity	4.16, 4.28, 4.29, 4.30, **4.35**, 4.36, 4.38, 4.39	`TransitiveObjectProperty( :isRelatedTo)`
$\mathcal{R}$	Complex role inclusions	4.18, 4.19, 4.26, **4.27**	`SubObjectPropertyOf( ObjectPropertyChain( :shows :represents):shows)`
$\mathcal{I}$	Inverse role	**4.14**, 4.15, 4.20 4.21, 4.22, 4.23 4.31, 4.32, 4.33 4.37	`InverseObjectProperties( :containsPerson :personLocation)`
$(\mathcal{D})$	Datatypes		`DataProperty(:xywh)`

using the WiseNET ontology we can represent the situation presented in Figure 4.12 where there is a small building that contains 1 storey and 2 spaces; one of the spaces contains 2 doors and 1 camera, while the other one contains 1 door and 1 camera; moreover, both cameras are looking at the same door and there is one of them which also observes a person. The situation can be modeled in the WiseNET ontology by inserting some knowledge as shown below. Notice that to ease readability, we will use in the rest of this chapter the *triple notation* for asserting axioms in the ontology, where the DL notation "$a : C$" corresponds to the triple notation "`<a> rdf:type C`" that means that "a" is an instances of the class "C". In a similar manner, DL notation "$(a, b) : r$" corresponds to the triple notation "`<a> r <b>`" that means that the instance "a" relates to the instance "b" by the property "r".

Firstly, the classes instances needs to be created. The inserted class instances are:

```
<building1> rdf:type bot:Building,
<storey1> rdf:type bot:Storey,
<space1> rdf:type bot:Space,
<space2> rdf:type bot:Space,
<door1> rdf:type :Door,
<door2> rdf:type :Door,
<cam1> rdf:type :SmartCamera,
<cam2> rdf:type :SmartCamera,
<fov1> rdf:type :FieldOfView,
<fov2> rdf:type :FieldOfView,
<person1> rdf:type foaf:Person.
```

Secondly, the created instances are related by properties. The inserted properties are:

```
<building1> bot:hasStorey <storey1>,
<storey1> bot:hasSpace <space1>,
<storey1> bot:hasSpace <space2>,
```

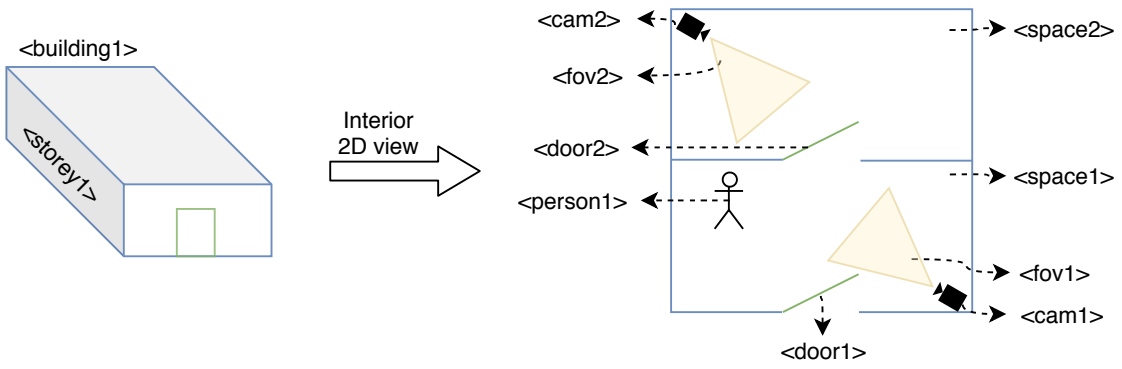


Figure 4.12 – Example of a situation that can be represented using the WiseNET ontology.

```
<space1> bot:containsElement <door1>,
<space1> bot:containsElement <door2>,
<space2> bot:containsElement <door2>,
<space1> sosa:hosts <cam1>,
<space2> sosa:host <cam2>,
<cam1> :hasFieldOfView <fov1>,
<cam2> :hasFieldOfView <fov2>,
<door2> :appearsIn <fov1>,
<door2> :appearsIn <fov2>,
<person1> :appearsIn <fov1>.
```

The inserted classes and properties allow to completely and correctly model the situation presented in Figure 3. Moreover, after inserting the knowledge, the reasoner can be executed to deduce new information. This new information was not explicitly inserted and can be inferred thanks to the constraints defined in Step 6, for example:

```
<building1> bot:containsZone <storey1>,
<building1> bot:containsZone <space1>,
<building1> bot:containsZone <space2>,
<building1> bot:hasElement <door1>,
<building1> bot:hasElement <door2>,
<building1> bot:hasElement <cam1>,
<building1> bot:hasElement <cam2>,
<cam1> sosa:isHostedBy <space1>,
<cam1> sosa:observes <door2>,
<cam1> sosa:observes <person1>,
<fov1> :shows <door2>,
<fov1> :shows <person1>,
<fov1> :isFieldOfViewOf <cam1>,
<door2> :isObservedBy <cam1>,
<door2> :isObservedBy <cam2>,
<person1> :isObservedBy <cam1>.
```

As a conclusion, the WiseNET ontology was able to completely and correctly model the example situation and, moreover, it enabled the deduction of new useful information.

However, this example situation revealed two main limitations of our current ontology.

1. The manual insertion of knowledge. This is feasible for toys examples where there are two spaces, two doors and two cameras (as the example presented). However, in a real video surveillance application where we have a camera network sending data constantly, manual insertion is unfeasible. In Chapter 5 we will present a solution to extract and insert the building and camera knowledge (i.e., knowledge about the structure and topology of the building, and the detections performed by the camera network) in an automatic way.

2. Missing inference information. Re-consider the example situation, with the inserted knowledge, we expected the ontology to deduce much more information, for example: that `<space1>` is connected to `<space2>` because they contained the same door; that `<cam1>` is nearby `<cam2>` because they are hosted by connected spaces; that `<fov1>` overlaps `<fov2>` because they show the same door; or simply that `<person1>` is located in `<space1>` because it is being observed by a camera hosted by that space. These type of rules cannot be represented using DL formalism, however they can be represented using *semantic rules*. Thus, the next section will present how to extend our WiseNET ontology by including this type of rules. Notice that the inclusion of semantic rules are not consider by the 101-methodology.

Moreover, the ontology needs to be checked according to all the CQs formulated in the Step 1. This evaluation will be presented in Chapter 6.

4.2.2/ SEMANTIC RULES

As previously stated the WiseNET ontology is defined is OWL 2 using the $\mathcal{SRI}(\mathcal{D})$ DL expressivity. However, all DL formalisms have expressive limitations, for example the composition of complex classes or properties resulting from the combination of both classes *and* properties simultaneously. Those limitations can be overcome by adding rule-based knowledge, especially by using Semantic Web Rule Language (SWRL) rules [77]. SWRL is built on OWL DL and shares its formal semantics, meaning that conclusions reached by SWRL rules have the same formal guarantees as the conclusions reached using standard OWL constructs. Moreover, SWRL provides more expressivity than OWL DL alone [77].

However, reasoning may become undecidable for the combination of OWL 2 + SWRL, i.e., inference with SWRL rules is not guaranteed to terminate. Therefore, the expressivity of SWRL needs to be reduced to assure decidability. Although, many procedures exists to guarantee decidability of SWRL, the DL-safe rules are the most commonly adopted [122]. This procedure consists in restricting the possible variables assignments to known individuals in an ontology (instances). All the rules presented in this manuscripts are DL-safe SWRL rules.

Listings 4.1 to 4.11 present the SWRL rules defined in the WiseNET ontology. An explanation of the SWRL rules syntax can be found in Section 3.3.2. The rules state the following:

- **Rule 1:** *If two spaces contain the same door, then those spaces are adjacent/connected* (Listing 4.1).

Listing 4.1 – SWRL rule for inferring if two spaces are adjacent.

```
bot:Space(?sp1) ∧ bot:Space(?sp2) ∧ :Door(?d)
∧ bot:containsElement(?sp1,?d) ∧ :containsElement(?sp2,?d)
⟹ bot:adjacentZone(?sp1,?sp2)
```

- **Rule 2:** *If two sensors are hosted by the same space, then those sensors are nearby* (Listing 4.2).

Listing 4.2 – SWRL rule for inferring if two sensors are nearby.

```
bot:Space(?sp) ∧ sosa:Sensor(?ss1) ∧  sosa:Sensor(?ss2)
∧ sosa:hosts(?sp,?ss1) ∧  sosa:hosts(?sp,?ss2)
⟹ :hasNearbySensor(?ss1,?ss2)
```

- **Rule 3:** *If two sensors are hosted by adjacent spaces, then those sensors are nearby* (Listing 4.3).

Listing 4.3 – SWRL rule for inferring if two sensors are nearby (2).

```
sosa:Sensor(?ss1) ∧  sosa:Sensor(?ss2) ∧ bot:Space(?sp1)
∧ bot:Space(?sp2) ∧ bot:adjacentZone(?sp1,?sp2)
∧ sosa:hosts(?sp1,?ss1) ∧ sosa:hosts(?sp2,?ss2)
⟹ :hasNearbySensor(?ss1,?ss2)
```

- **Rule 4:** *If two fields of view show the same door, then those fields of view overlap* (Listing 4.4).

Listing 4.4 – SWRL rule for inferring if two fields of views overlap.

```
:FieldOfView(?fov1) ∧ :FieldOfView(?fov2) ∧ :Door(?d)
∧ :shows(?fov1,?d) ∧ :shows(?fov2,?d)
⟹ :overlaps(?fov1,?fov2)
```

- **Rule 5:** *If a person is observed by a smart camera, then that person is located in the space hosting the smart camera* (Listing 4.5).

Listing 4.5 – SWRL rule for inferring the location of a person.

```
foaf:Person(?p) ∧ bot:Space(?sp) ∧ :SmartCamera(?sc)
∧ sosa:hosts(?sp,?sc) ∧ sosa:observes(?sc,?p)
⟹ :personLocation(?p,?sp)
```

- **Rule 6:** *If a zone contains a person, then that zone is occupied* (Listing 4.6).

Listing 4.6 – SWRL rule for inferring if a zone is occupied.

```
bot:Zone(?z) ∧ foaf:Person(?p) ∧ :containsPerson(?z,?p)
⟹ :isOccupied(?z,true)
```

- **Rule 7:** *If two person-in-space events involve the same person, then those events are related* (Listing 4.7).

Listing 4.7 – SWRL rule for inferring if two person-in-space events are related.

```
:PersonInSpace(?pis1) ∧ :PersonInSpace(?pis2) ∧ foaf:Person(?p)
∧ event:agent(?pis1,?p) ∧ event:agent(?pis2,?p)
⟹ :isRelatedTo(?pis1,?pis2)
```

- **Rule 8:** *If a person is located in a restricted space, then that person is an intruder* (Listing 4.8).

Listing 4.8 – SWRL rule for inferring if a person is an intruder.

```
foaf:Peson(?p) ∧ bot:Space(?sp)
∧ :personLocation(?p,?sp) ∧ :isRestricted(?sp,true)
⟹ :isIntruder(?p,true)
```

- **Rule 9:** *If a detection is performed, by a smart camera in a restricted space, then that detection is an entry violation event and the smart camera should start recording* (Listing 4.9). Notice that this rule could be divided into two—one for each consequence—however, both consequences are tightly related thus we decided to keep them together.

Listing 4.9 – SWRL rule for inferring if a detection is an entry violation and if a smart camera should start recording.

```
:Detection(?dt) ∧ bot:Space(?sp) ∧ :SmartCamera(?sc)
∧ sosa:madeBySensor(?dt,?sc) ∧ event:place(?dt,?sp)
∧ :isRestricted(?sp,true)
⟹ :isEntryViolation(?dt,true) ∧ :startRecording(?sc,true)
```

- **Rule 10:** *If there is a person located in a restricted space, and the space has an alarm, then the alarm should be triggered with an intruder flag* (Listing 4.10).

Listing 4.10 – SWRL rule for triggering the alarm if there is an intruder.

```
foaf:Peson(?p) ∧ bot:Space(?sp) ∧ Alarm(?a)
∧ :personLocation(?p,?sp) ∧ :isRestricted(?sp,true)
∧ :hasAlarm(?sp,?a)
⟹ :triggeredByIntruder(?a,true)
```

- **Rule 11:** *If the number of people in a space is greater or equal than its maximal capacity, and the space has an alarm, then the alarm should be triggered with a maximal-capacity flag* (Listing 4.11). Notice that `swrl:greaterThanOrEqual` is a SWRL built-in function that allows the comparison of two values, more details on the SWRL built-in functions can be found in [77].

Listing 4.11 – SWRL rule for triggering the alarm if the maximum capacity is exceeded.

```
bot:Space(?sp) ∧ :Alarm(?a) ∧ :hasAlarm(?sp,?a)
∧ :maxCapacity(?sp,?val1) ∧ :numberOfPeople(?sp,?val2)
∧ swrlb:greaterThanOrEqual(?val1,?val2)
⟹ :triggeredByMaxCapacity(?a,true)
```

To observe the impact of the SWRL rules, let us re-considered the example-situation presented in the Step 7 of the previous section (see Fig. 4.12 in Section 4.2.1). In order to trigger most of the rules, let us insert—in addition to the knowledge inserted in the Step 7—some axioms stating that `<space1>` is a restricted space, that it has a maximal capacity of 1, that it has an alarm and that currently there is one person in it. The complete set of inserted knowledge is presented in Table 4.5. After inserting the knowledge in the ontology, the reasoner is executed to infer new information. Table 4.6 presents some of the inferred information, along with the rules and constraints used to deduce them. As it can be observed, the consideration of the SWRL rules and the ontology constraints allows

Table 4.5 – Knowledge inserted in the WiseNET ontology to model the situation presented in Fig. 4.12. The axioms in gray were introduced in this section while the rest were introduced in the Step 7 of Section 4.2.1.

Class assertions	**Property assertions**
<building1> rdf:type bot:Building	<building1> bot:hasStorey <storey1>
<storey1> rdf:type bot:Storey	<storey1> bot:hasSpace <space1>
<space1> rdf:type bot:Space	<storey1> bot:hasSpace <space2>
<space2> rdf:type bot:Space	<space1> bot:containsElement <door1>
<door1> rdf:type :Door	<space1> bot:containsElement <door2>
<door2> rdf:type :Door	<space2> bot:containsElement <door2>
<cam1> rdf:type SmartCamera	<space1> sosa:hosts <cam1>
<cam2> rdf:type SmartCamera	<space2> sosa:host <cam2>
<fov1> rdf:type :FieldOfView	<cam1> :hasFieldOfView <fov1>
<fov2> rdf:type :FieldOfView	<cam2> :hasFieldOfView <fov2>
<person1> rdf:type foaf:Person	<door2> :appearsIn <fov1>
<alarm1> rdf:type :Alarm	<door2> :appearsIn <fov2>
	<person1> :appearsIn <fov1>
	<space1> :isRestricted "true"
	<space1> :maxCapacity "1"
	<space1> :hasAlarm <alarm1>
	<space1> :numberOfPeople "1"

the deduction of rich information from few explicit information. The inference process was performed using the Pellet reasoner [155] in the Protégé ontology editor [162]. The inference task took around 19.3 ms, time obtained by executing the reasoner 10 times and then performing an average. This procedure was performed in a machine with the following configuration: Intel Core i7-4790 CPU @3.6GHz $\times$ 4, 16GB of RAM and a "Java Heap" size set to 200MB.

Table 4.6 – Some inferred information after executing the reasoner. The explanation of the axioms refers to the rules and constrain formulas used to deduce the information.

Inferred information	Explaination
`<building1> :containsPerson <person1>`	Rule 5 + Eq. 4.18
`<building1> :isOccupied "true"`	Rule 5 + Eq. 4.18 + Rule 6
`<storey1> :containsPerson <person1>`	Rule 5 + Eq. 4.18
`<storey1> :isOccupied "true"`	Rule 5 + Eq. 4.18 + Rule 6
`<space1> :isOccupied "true"`	Rule 5 + Rule 6
`<space1> bot:adjacentZone <space2>`	Rule 1
`<space2> bot:adjacentZone <space1>`	Rule 1
`<cam1> :hasNearbySensor <cam2>`	Rule 1 + Rule 3
`<cam2> :hasNearbySensor <cam1>`	Rule 1 + Rule 3
`<fov1> :overlaps <fov2>`	Eq. 4.20 + Rule 4
`<fov2> :overlaps <fov1>`	Eq. 4.20 + Rule 4
`<person1> :personLocation <space1>`	Eq. 4.20 + Eq. 4.26 + Rule 5
`<person1> :personLocation <storey1>`	Rule 5 + Eq. 4.18 + Eq. 4.14
`<person1> :personLocation <building1>`	Rule 5 + Eq. 4.18 + Eq. 4.14
`<person1> :isIntruder "true"`	Rule 5 + Rule 8
`<alarm1> :triggeredByIntruder "true"`	Rule 10
`<alarm1> :triggeredByMaxCapacity "true"`	Rule 11

4.3/ CONCLUSION

In this chapter, we introduced a framework called WiseNET(Wised NETwork), which is a semantic-based system that fuses heterogeneous sources of data in the Intelligent Video Surveillance (IVS) and smart building domains. The creation of a context-aware system in the built environment is a complex task; it requires information from different domains such as environment data, sensing devices, spatio-temporal facts and details about the different events that may occur. For example, the required event information could be a set of concepts and relations concerning the different events that may occur in a built environment, their location, the time they occurred, the agents involved, the relation to other events and their consequences. In the case of the sensor information, the required data could be the description of the different sensing devices, the processes implemented on them and their results. Regarding the environment, the required data could be the building topology and the different elements contained in the spaces.

The main goals of WiseNET are to enhance what sensors "observe" by considering contextual information, and to provide a set of services to the building managers to ease their work by fusing pertinent information. In this work we focused on visual sensors, however, the WiseNET system was designed in a generic manner, meaning that any type of sensor could be considered.

The WiseNET ontology is the kernel of the WiseNET system and is responsible for (1) describing the different kinds of information presented in the system and (2) enabling interoperability between them. The interoperability between heterogeneous sources of information is performed in a *semantic level*, i.e., a set of terms are defined which bridge the heterogeneous sources. Moreover, the definition of each term in the ontology should not contradict the model's constraints.

The ontology development procedure was performed using different semantic web technologies, and it consisted of: defining a set of questions that the ontology should answer (competency questions); reusing different domain ontologies (`bot`, `event`, `ssn`, `time` and `foaf`); creating a set of classes and properties to connect the different domain ontologies and to complete the application knowledge; defining a set of constrains and extending the expressiveness by using logic rules. However, we propose to modify the beginning of the procedure by extracting the relevant terms from the competency questions, enumerate them and cluster them into categorize, and then to look for external ontologies that have already defined them. The reuse of external ontologies not only saves time but also gives the advantage of using mature and proved ontological resources that have been validated by their applications and by the W3C. Moreover, SWRL allows to extend the knowledge of an ontology, by enabling the inclusion of human-skill knowledge in the form of rules. In our case, the rules help the analyzes of the multiple situations that can occur in the environment, such as knowing if a space is occupied or if the maximal capacity of a space has been reached.

The resulting WiseNET ontology allows the description of information concerning smart building manage and IVS systems. Furthermore, as shown by the examples, it accurately infers information relating a sensor network, with the environment and its users. Thus, allowing to understand what is happening in the building in an intuitive way, which is human and machine understandable.

Once the WiseNET ontology is formally defined, the next step is to populate the ontology with information about the built environment and sensor setup (static information), and with information about the detections performed by the sensors (dynamic population). The population step will be presented in the next chapter, along with the different processes developed for performing the population in an automated way.

Remark. *Most of the information presented in this chapter was validated in the following papers [109, 110, 112].*

5

STATIC AND DYNAMIC ONTOLOGY POPULATION

The WiseNET ontology, defined in Chapter 4, incorporates a vast corpus of concepts in the domain of an Intelligent Video Surveillance (IVS) system, which enables the aggregation of knowledge coming from a Smart Camera Network (SCN) deployed in a built environment. After having defined the WiseNET ontology, the next step is to automatically *populate* the ontology with pertinent information that will enable the analysis of an IVS system. The ontology population process can be divided into *static* and *dynamic population*. The static population, that will be presented in Section 5.2, consist in inserting the knowledge that (normally) stays unchangeable in an IVS system, such as information of the environment and the calibration information of the SCN. The dynamic population, that will be presented in Section 5.3, consist in extracting pertinent knowledge from the continuous data sent by the SCN (e.g., detections), to process it and to insert it into the ontology. Furthermore, the central API, presented in Section 5.1, is in charge of managing the ontology, thus enabling the static and dynamic population of data, as it can be observed in Fig. 4.1. In summary, this chapter focus on the insertion of data coming from the SCN and the built environment into the WiseNET ontology, by using the central API.

All the vocabulary, prefixes, rules and equations used and referred to in this chapter, were defined in Chapter 4. Moreover, specific details about the vocabulary can be also find in Appendix A.

5.1/ CENTRAL API

An Application Programming Interface (API) is an intermediary software that enables the communication/interaction and data sharing between various components [148]. An API furnishes a set of well-defined methods—referred as web services—accessible via web protocols, such as the Hypertext Transfer Protocol (HTTP) [57]. HTTP is a protocol utilize for machine-to-machine communication, more specifically for transferring machine-readable file formats such as JSON (JavaScript Object Notation). HTTP defines different commands that allows accessing the API's web services, for example the GET and POST commands used to retrieve and send data to a web service, respectively.

In the case of the WiseNET system, the central API is the interface between the WiseNET ontology and the incoming and outgoing data (see Fig. 4.1). The central API is composed of web services in charge of the updating/management of the WiseNET ontology. Specif-

ically, the central API is in charge of:

- inserting the environment data into the ontology,
- capturing the data coming from the SCN and inserting it into the ontology,
- retrieving the inferred knowledge from the ontology, and
- transferring data to the monitor unit.

Furthermore, the central API performs some processing tasks before inserting the data, as it will be presented in this chapter, for example it filters the incoming data, to don't over saturate the WiseNET ontology with redundant information.

All the population processes and queries presented in this chapter are performed by the central API. Thus, making it a key element of the static and dynamic populations.

Moreover, ontologies are not meant to be used in situations were data is constantly changing over time, which is the case of the dynamic population in our system, thus the role of the central API is crucial to enable this type of ontology usage.

5.2/ STATIC POPULATION

We considered as *static population*, the insertion of information that is (normally) unchangeable after its introduction in the system. This concerns the information of the environment, presented in Section 5.2.1, and the general information of a SCN, presented in Section 5.2.2.

As a running example for this chapter, we will use the *Institut Marey et Maison de la Métallurgie* (I3M) building located in Dijon (France). The I3M building has three storeys, from which we will focus on the third storey where a SCN has been deployed. Notice that, **all the process presented in this chapter are general, and they do not depend on the I3M building nor of the SCN deployed in it**. We decided to use it as an example because its information is easily available to us (it is our laboratory) and we believe that is easier to understand something if an example is given.

Figure 5.1 depicts the position of the camera nodes in the I3M building, as well as the position of the spaces and doors relevant to our system. Mostly, we are interested in the doors observed by the camera nodes and the spaces containing those doors.

5.2.1/ ENVIRONMENT KNOWLEDGE EXTRACTION AND POPULATION

To analyze the activities of a building, the information of the environment is crucial to understand what is happening and *where* is happening.

In the case of the WiseNET system, it requires information about the structure of the building (number and location of storeys and spaces), its spatial topology (storey-storey relation, storey-space relation and space-space relation) and the different elements contained in the spaces. Figure 5.2 summarize, in the form of a graph, the required environment elements and their relations that need to be populated into the WiseNET system. As shown in the figure, we are only interested on four classes of objects `Building`,

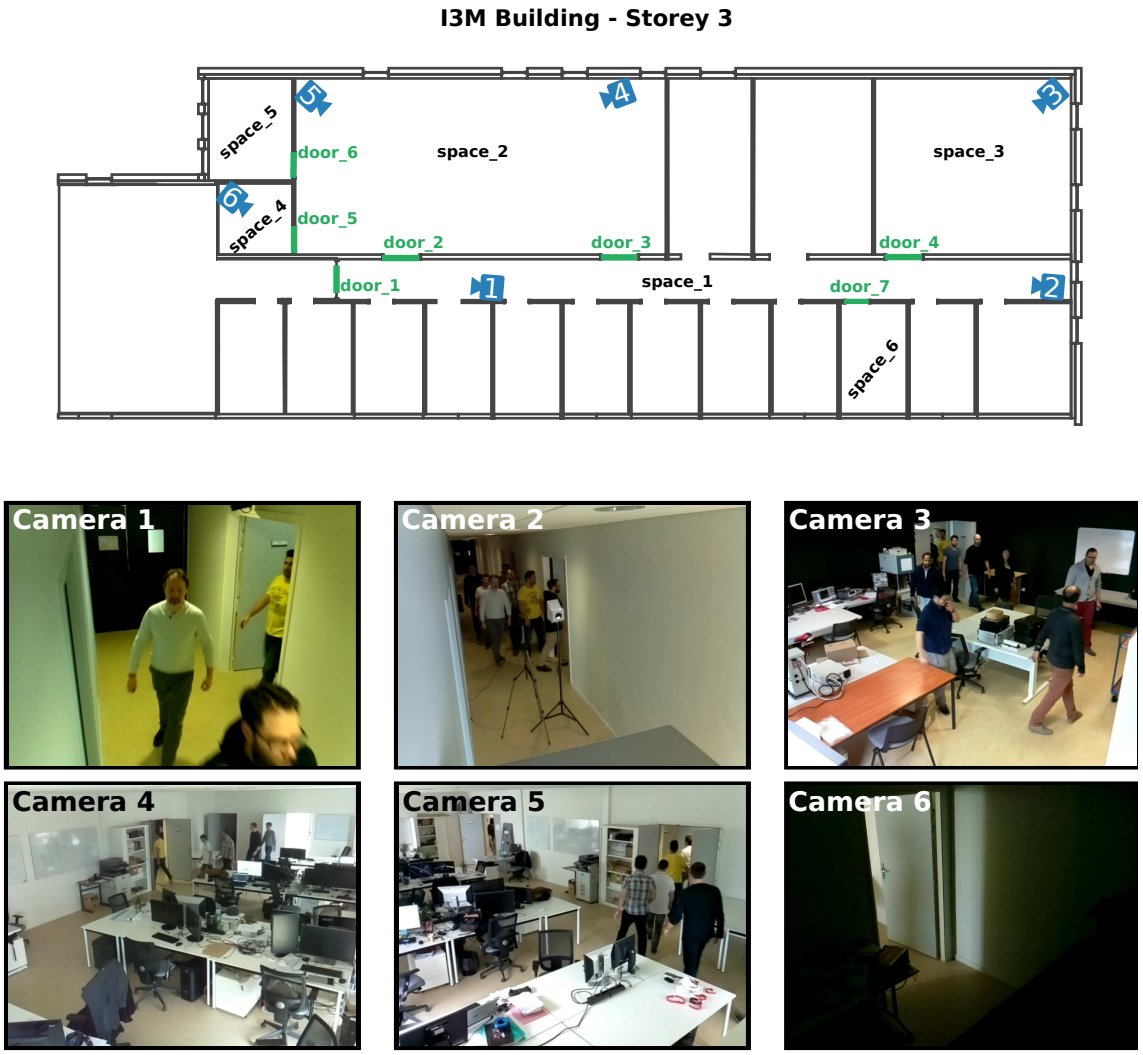


Figure 5.1 – WiseNET network deployed in the I3M building. (Top) Illustration of the position of the six camera nodes (in blue), and some spaces and the doors (in green) of interest. (Bottom) Example images from the camera nodes. Images from cameras 1, 4, 5 and 6 were taken at the same time; similarly for images 2 and 3.

`Storey`, `Space` and `Door` and the relations between them. We are specifically interested in *door* elements, instead of other elements (such as walls, stairs or windows), due to their importance in a building environment, e.g., they connect two spaces and people have to pass through them to enter/exit a space, however the other elements could also be considered if needed.

The process shown in Fig. 5.3 was developed for automatically obtaining and populating the required environment data into the WiseNET ontology. The process consists mainly of five components: (1) an Industry Foundation Classes (IFC) file, (2) a requirement checker of the IFC file, (3) a conversion of the IFC into the `ifcowl` ontology, (4) the extraction of the pertinent instances from `ifcowl`, and finally (5) the population of the extracted instances and their relationships into the `bot` part of the WiseNET ontology (see Section 4.2.1).

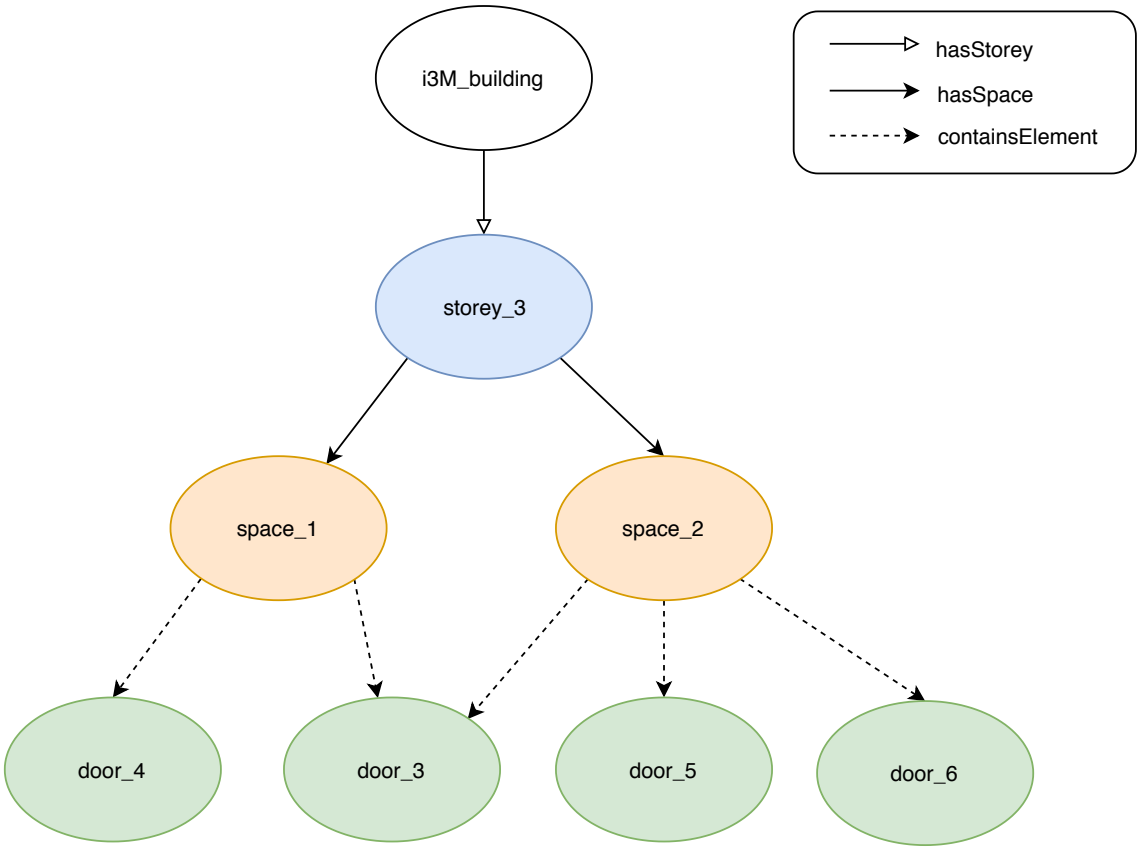


Figure 5.2 – Example of the environment elements and their relations that need to be populated into the WiseNET system. The graph only represents a small selection of spaces and elements present on the third storey of the I3M building (see Fig 5.1). The colors of the graph's nodes denote the different types (classes) of elements such as *Building* (in white), *Storey* (in blue), *Space* (in orange) and *Door* (in green).

(1) IFC file The process starts with an IFC file of a building, which will be the source of information. In the context of Building Information Modelling (BIM), a digital representation of the building comes in the form of one or several IFC files [44] (see Section 3.2 for more details). An IFC file contains a large amount of information concerning a built environment, including the information that the WiseNET system requires. For example, it includes information about all the elements composing the building (storeys, spaces, stairs, elevators, doors, walls, electrical elements, gas/heating/water distribution systems, etc.), their geometrical information (dimension of each space/stair/door/wall, dimension of distribution elements such as pipes, etc.), their position and their relation to other elements (a space is restricted by walls, a wall has a door/window in it, etc.).

Following our example, the IFC file describing all the elements composing the I3M building was obtained from the company in charge of the construction of this building.

(2) Requirements checker The *requirements checker* process is in charge of verifying the presence of necessary entities in the IFC file, according the environment information required by the WiseNET system. The necessary IFC-entities that need to be instantiated are:

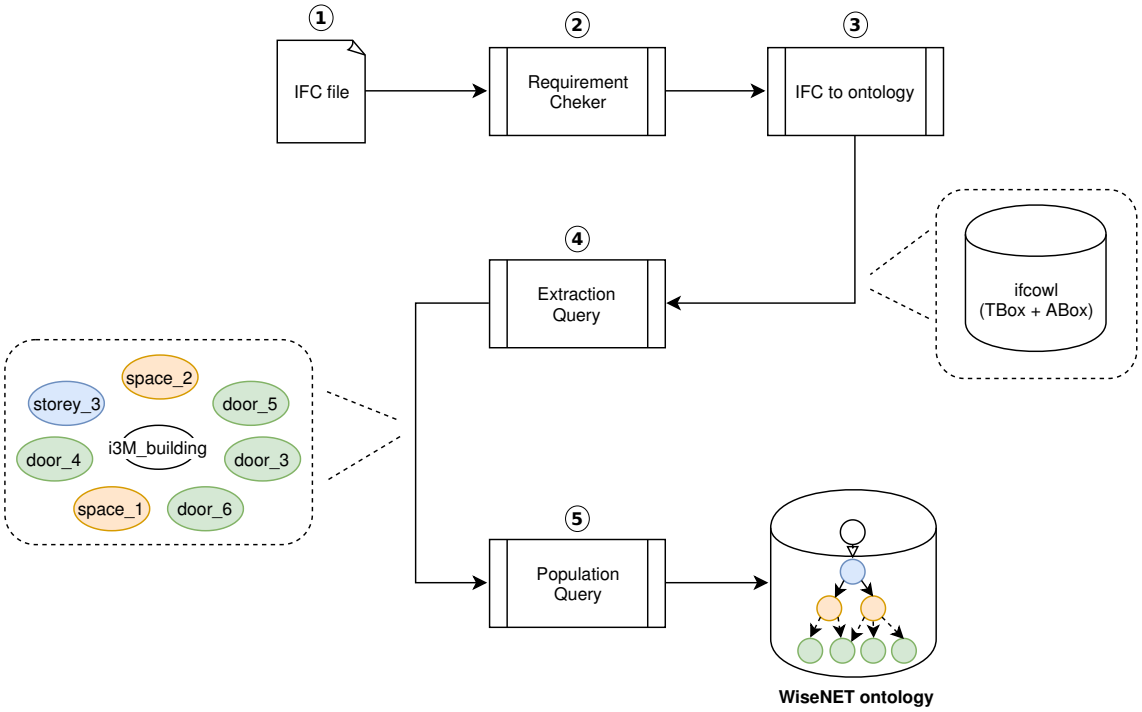


Figure 5.3 – IFC to WiseNET: environment extraction and population process. The dotted elements show outputs of some components.

- `IfcBuilding`, use to represent building instances,
- `IfcBuildingStorey`, use to represent storey instances,
- `IfcSpace`, use to represent space instances,
- `IfcDoor`, use to represent door instances,
- `IfcRelAggregates`, is an intermediate class used to represent a composition (and decomposition) relationship between an instance and a set of instances. For example, if a building instance contains a set of storeys, this "set of storeys" is represented by an instance of `IfcRelAggregates`, which itself is decomposed by each storey instance. The same occurs for the relationship between a storey and a set of spaces.
- `IfcRelSpaceBoundary`, is and intermediate class used to represent the relation between a space and different building elements, such as door, walls, windows, etc.

(3) IFC to ontology The manipulation of data contained in an IFC file is a fastidious process, mainly performed manually and therefore source of numerous errors [44]. In order to facilitate the handling of IFC files, some researchers propose to convert (represent) the IFC into an ontology [55, 135]. An IFC-based ontology model enhances the interoperability of the IFC data and facilitates its access and manipulation, for example by using Semantic Web technologies such as SPARQL queries.

Therefore, after checking the compliance of the IFC file, the file is converted to an ontology by using the *IFC-to-ontology* converter developed by Pauwels and Oraskari [133].

The result of the conversion is the `ifcowl` ontology which consists of a semantic representation of the IFC schema (TBox) with the instances of a building (ABox), in our example the instances concerning the I3M building.

In the `ifcowl` ontology, the TBox elements (classes and relationships) are defined using the prefix `ifcowl:`, while the ABox (instances) are defined using the prefix `inst:` which is a prefix assigned by the *IFC-to-ontology* converter and is used only to define IFC instances.

(4) Extraction query The `ifcowl` ontology (which is a loyal conversion of the IFC file) contains a large amount of information concerning the environment. However, only a small portion is required in the WiseNET system. For example, the `ifcowl` of the I3M building contains 1,350,346 instances including Cartesian points, polygons, measurements, characteristics of each building element, etc., from which we are only interested in 75 instances. Therefore, to improve the WiseNET ontology performance (by reducing its data complexity), only the required data will be extracted from the *ifcowl* ontology.

The required data could be obtained as an `ifcowl` sub-graph, as shown in Fig 5.4. However, in practice, the complexity and the structure of the `ifcowl` makes it hard to use and to extend. For example, consider the relationship for stating that a building contains a space, in `ifcowl` this relationship is defined by using an intermediate `IfcRelAggregates` instance, which raises the complexity unnecessarily [134, 112]. Moreover, by comparing the required data graph shown in Fig 5.2 with the `ifcowl` graph shown in Fig 5.4, it can be easily observed that the `ifcowl` is not only overcomplicated but also it uses more instances and relationships to define the same information—12 instances and 12 relationships in compare to 8 respectively. Therefore, instead of extracting a sub-graph from the `ifcowl`, we will just extract the pertinent instances (the ellipses in Fig 5.4) and afterwards we will re-create their relationships in a simpler manner.

To extract the pertinent instances, the `ifcowl` ABox is queried using the SPARQL code shown in Listing 5.1, where:

- Line 4 obtains the building instance by using its class.
- Line 7 acquires the array of building storeys that decompose the building; and line 8 obtains the storeys inside that array.
- The same is done for the spaces that decompose the storeys on Lines 11-12.
- Lines 15-16 obtain the elements that are contained in a space.
- Lines 19 filter out the undesired elements (such as windows, walls, etc) just leaving the doors.

The result of the query is the *extracted table* presented in Table 5.1, where the columns corresponds to the variables used with the SELECT operator (Listing 5.1, Line 1) and each row correspond to instance entries. Notice that the *?building* and *?storey* columns have the same value, this is due to there is only one building in the file—the `i3m_building`—and we are only interested in the entries related to a specific building storey—the `storey_3` where the cameras are installed.

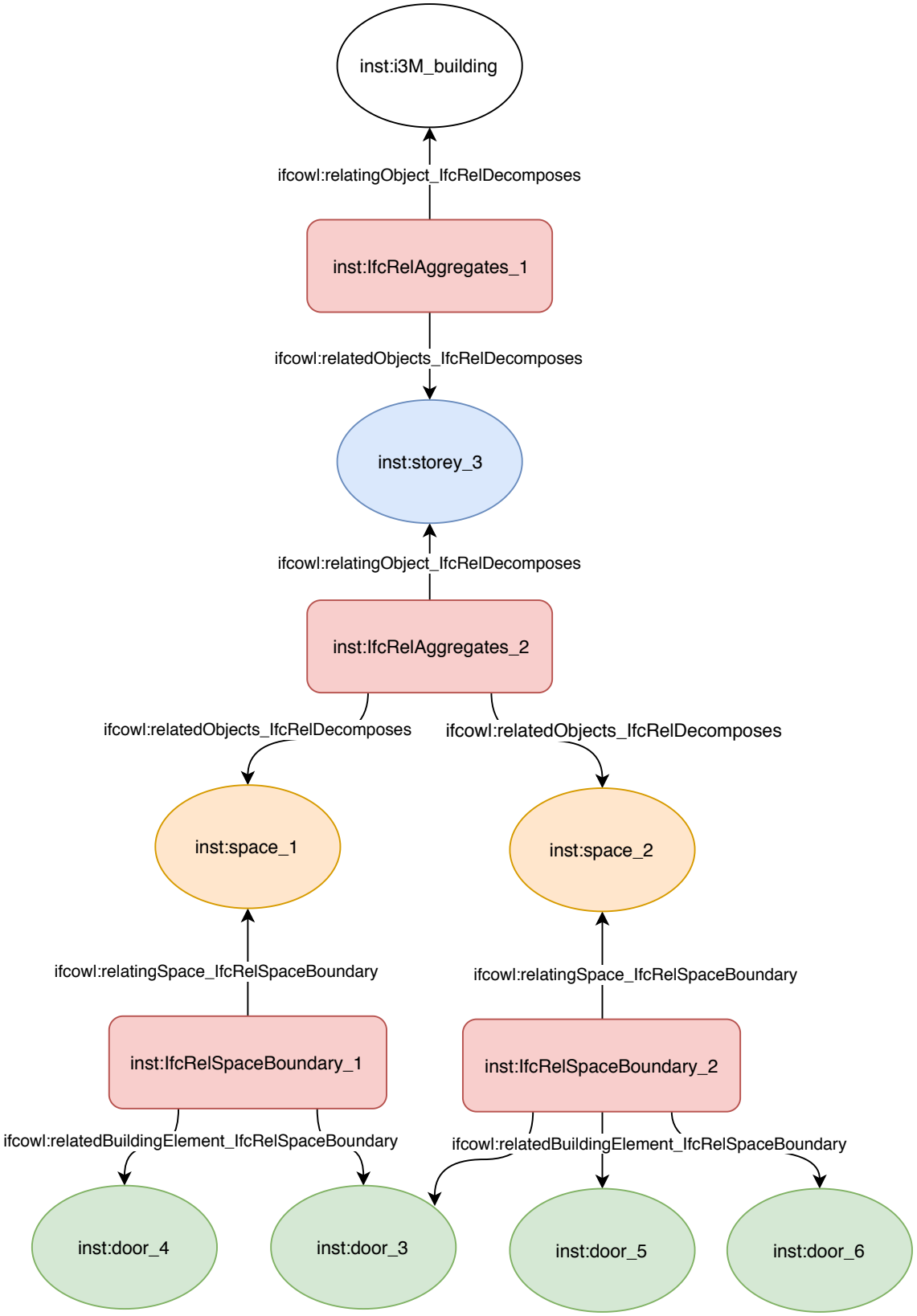


Figure 5.4 – Graph showing the representation of the required data in the `ifcowl` ontology. The red rectangles are intermediate instances which overcomplicate the `ifcowl` structure.

Listing 5.1 – SPARQL query for extracting pertinent instances from the `ifcowl` ontology. Lines that start with # are comments.

```
1  SELECT ?building ?storey ?space ?door
2  WHERE {
3     # Get building
4     ?building rdf:type ifcowl:IfcBuilding.
5
6     # Get building storeys
7     ?storey_array ifcowl:relatingObject_IfcRelDecomposes ?building;
8                   ifcowl:relatedObjects_IfcRelDecomposes ?storey.
9
10    # Get spaces: room, corridors, hall, etc
11    ?space_array ifcowl:relatingObject_IfcRelDecomposes ?storey;
12                 ifcowl:relatedObjects_IfcRelDecomposes ?space.
13
14    # Get elements: doors, windows, walls, floor, furnitures, etc
15    ?element_array ifcowl:relatingSpace_IfcRelSpaceBoundary ?space;
16                   ifcowl:relatedBuildingElement_IfcRelSpaceBoundary ?door.
17
18    # Filter elements to just keep doors, walls and windows
19    ?door rdf:type ifcowl:IfcDoor.
20 }
```

(5) Population query After extracting the pertinent instances from `ifcowl`, we need to insert them into the WiseNET ontology, while re-creating the relationship between instances. To re-create the relationships we will use the vocabulary defined in the Building Topology Ontology (`bot`) [140]. The `bot` ontology covers the core concepts for describing a building, its topology and its elements. Moreover, `bot` uses a simple schema that allows the creation of direct relationships between the pertinent instances. The `bot` classes used are `bot:Building`, `bot:Storey` and `bot:Space`; while the `bot` relationships used are `bot:hasStorey`, `bot:hasSpace` and `bot:containsElement`. The `bot` schema corresponds to the graph presented in Fig. 5.2.

To accomplish the population, each row of the extracted table (Table 5.1) is passed through the *population query*, and then executed in the WiseNET ontology. Listing 5.2 shows the population query, where:

- Line 1 inserts into the WiseNET ontology the triples defined below.
- Lines 3-6 define the objects as instances/individuals (`owl:NamedIndividual`).
- Lines 9-12 states the class of each instances. Notice that the class `Door` was not defined in `bot`, thus it was defined in the `wisenet` ontology.
- Lines 15-17 relate the extracted instances of the `ifcowl` using `bot` relationships.

The process needs to be repeated for all the rows of the extracted table, which is achieved by using an external loop.

To summarize, the WiseNET ontology is populated with the environment knowledge obtained from an IFC file. The process starts by converting the IFC file into an `ifcowl` ontology, and then extracting the relevant environment information. Finally, the extract information is inserted into the WiseNET ontology by using the `bot` vocabulary. This extraction and population process uses different semantic web technologies such as SPARQL

Table 5.1 – Extracted environment instances from the `ifcowl` ontology. The results were obtained after executing the extraction query in Listing 5.1.

?building	?storey	?space	?door
inst:i3m_building	inst:storey_3	inst:space_1	inst:door_1
inst:i3m_building	inst:storey_3	inst:space_1	inst:door_2
inst:i3m_building	inst:storey_3	inst:space_1	inst:door_3
inst:i3m_building	inst:storey_3	inst:space_1	inst:door_4
inst:i3m_building	inst:storey_3	inst:space_1	inst:door_7
inst:i3m_building	inst:storey_3	inst:space_2	inst:door_2
inst:i3m_building	inst:storey_3	inst:space_2	inst:door_3
inst:i3m_building	inst:storey_3	inst:space_2	inst:door_5
inst:i3m_building	inst:storey_3	inst:space_2	inst:door_6
inst:i3m_building	inst:storey_3	inst:space_3	inst:door_4
inst:i3m_building	inst:storey_3	inst:space_4	inst:door_5
inst:i3m_building	inst:storey_3	inst:space_5	inst:door_6
inst:i3m_building	inst:storey_3	inst:space_6	inst:door_7

queries. Furthermore, if required, extra information could be extracted from the `ifcowl` ontology using the same methodology. For example, the dimensions of a door, its material and the dimensions of the spaces.

THOUGHTS ON IFC EXTENSION

The IFC data could be enhanced by considering information about the space functionalities and space security. Particularly, we propose to extend the IFC data by adding extra information regarding:

- **Space functionality**: spaces have many usages, such as: corridor, office, co-working room, lobby, kitchen and infirmary room. By considering the space functionality it is possible to design specific rules according to each function. For example, knowledge that people should not loiter in a particular corridor could be inserted into the ontology in form of a rule. Another example could be the knowledge that in the infirmary room it is normal to have people laying down.
- **Space capacity**: the capacity of a space is very important for security reasons, for example, to have and efficient emergency protocol, the quantity of people should not exceed a certain number depending on the environment. Moreover, when spaces, and environments in general, are design, their intended capacities need to be taken into account, thus facilitating the addition of this information into the IFC.
- **Space alarm property**: spaces could have different types of alarm, such as: fire alarm, siren and light alarm. We consider that a smart building should know the information about which spaces have alarms, the type and what triggers them.
- **Security system property**: spaces or more precisely doors might have different types of security systems, such as: key-lock system, card reader, keypad and biometric systems. This information could be used to know the level of restriction of a space.

Listing 5.2 – SPARQL query for inserting the instances and their relationships into the WiseNET ontology. In italics are the instances of the first row of Table 5.1. Lines that start with # are comments.

```
INSERT DATA {
   # Define instances
   inst:i3m_building rdf:type owl:NamedIndividual.
   inst:storey_3 rdf:type owl:NamedIndividual.
   inst:space_1 rdf:type owl:NamedIndividual.
   inst:door_1 rdf:type owl:NamedIndividual.

   # Insert types
   inst:i3m_building rdf:type bot:Building.
   inst:storey_3 rdf:type bot:Storey.
   inst:space_1 rdf:type bot:Space.
   inst:door_1 rdf:type wisenet:Door.

   # Create relations
   inst:i3m_building bot:hasStorey inst:storey_3.
   inst:storey_3 bot:hasSpace inst:space_1.
   inst:space_1 bot:containsElement inst:door_1.
}
```

The extra information may allow the deduction of security restrictions and the design of rules according to the space usage.

5.2.2/ SMART CAMERA STATIC INFORMATION

After adding the environment information into the WiseNET ontology, the next step is to populate the static information about the Smart Camera Network (SCN).

There are two types of information that need to be populated concerning the SCN. Firstly, the setup information, which consists in describing the Smart Cameras (SCs) and their relation to the built environment. Secondly, the information generated each time the SCs perform a detection. The first one is inserted once, during the system configuration, therefore it is considered as a *static population* and it will be presented in this section. While the second one, needs to be inserted each time there is a detection, therefore it is considered as a *dynamic population* and it will be presented in Section 5.3.

SMART CAMERA SOFT CALIBRATION

The SCN setup requires the information about the position of the camera nodes in the environment and about the objects/regions of interest they observe. This setup process can be seen as a *soft camera-calibration process* because it only requires the knowledge of the location of the cameras in the building. This differs from standard calibration processes, that require the *extrinsic* and *intrinsic* parameters of the camera (transformations between the world coordinate to the camera's coordinate and to the 2D image coordinate), thus obtaining the precise information about the camera's locations and orientations, however leading to a time-consuming and skill-dependent calibration process [159].

For each camera node, the setup information is stored in a *camera-calibration* file. This

file is structured in a JSON format (a lightweight format to serialize structure data[1]) where each field corresponds to:

- *deviceID*: identification of the camera node.
- *isHostedBy*: space where the camera node is located.
- *ipAddress*: IP address of the camera node.
- *implements*: set of algorithms/procedures implemented by the camera.
  - *imageProcessing*: name of the image processing algorithm.
- *observes*: set of Regions Of Interest (ROIs) observed by the camera.
  - *regionOfInterest*: identification of the ROI.
  - *xywh*: position of the ROI in the camera's Field Of View (FOV), where (x,y) are the coordinates of the ROI's top-left point, and (w,h) are the width and height respectively.
  - *represents*: real object represented by the ROI.

An example of a camera-calibration file is presented in Listing 5.3. The information contained in the file can be summarize as: the `smartCamera_4` is located at `space_2`, it implements the `HOG_SVM` and `YOLOv3` image processing algorithms, and it observes three ROIs in the image scene which represent the `door_2`, `door_5`, and `door_6`.

There is a *semantic gap* between the visual information of an image and its physical representation. Thus, the selection of ROIs in the camera image is known as semantic-labelling and its goal is to bridge the gap between the real objects (or regions) and their projections in the camera view. As stated previously, we will only consider doors as ROIs due to their importance in a building environment, e.g., they connect two spaces and people have to pass through them to enter/exit a space, however, the system is able to consider other ROIs if required.

The camera-calibration file is stored in each camera node. Thus, each camera knows the position of each ROI in its FOV and its identification. Moreover, a software was developed that allows to draw ROIs directly in the camera view, hence facilitating the obtention of their coordinates.

After having defined the soft camera-calibration information for each camera node, the information needs to be populated into the WiseNET ontology. The population process is presented in Algorithm 1, and is explained as follows: firstly, in Line 2, a *camera-calibration* file is taken from the set of files (the FOR loop). For exemplification, consider the file presented in Listing 5.3. Afterwards, in Line 3, the smart camera and its FOV are defined and inserted by performing the query presented in Listing 5.4. Notice that the prefix `wni:` is used to define WiseNET instances. Then, in Line 6, the relationship between the smart camera and the algorithms is inserted by performing the query presented in Listing 5.5. This query needs to be performed for each algorithm defined in the *calibration-file* thus the need of the FOR loop (Line 5). Finally, in Line 9, the ROIs are defined and inserted by performing the query presented in Listing 5.6. This query needs to be performed for each ROI defined in the *calibration-file* thus the need of the FOR loop (Line 8).

Listing 5.3 – Camera-calibration file of smart camera #4.

```
{
    "deviceID": "smartCamera_4",
    "isHostedBy": "space_2",
    "ipAddress": "10.141.13.26",
    "implements": [
        {       "imageProcessing": "HOG_SVM"   },
        {       "imageProcessing": "YOLOv3"    }
    ],
    "observes": [
        {   "regionOfInterest": "regionOfInterest_8",
            "xywh": [552, 119, 98, 239],
            "represents": "door_2"
        },
        {   "regionOfInterest": "regionOfInterest_9",
            "xywh": [1048, 85, 125, 193],
            "represents": "door_6"
        },
        {
            "regionOfInterest": "regionOfInterest_10",
            "xywh": [805, 83, 79, 146],
            "represents": "door_5"
        }
    ]
}
```

Algorithm 1 Pseudocode for populating the static information of the cameras.

Input: Set of *camera-calibration* files
Output: Information populated into the WiseNET ontology

```
procedure SCN STATIC POPULATION
    for each camera-calibration file do
        insert camera and FOV instances (query in Listing 5.4)

        for each algorithm in the camera-calibration do
            insert camera-algorithm relationship (query in Listing 5.5)

        for each ROI in the camera-calibration do
            insert ROI instance (query in Listing 5.6)
```

SCN: Smart Camera Network
FOV: Field Of View
ROI: Region Of Interest

An example of the semantic-graph inserted during the static camera population process is presented in Fig 5.5. The left part of the graph corresponds to the `smartCamera_4` (Listing 5.3) while the right side corresponds to the `smartCamera_1`. The information about `smartCamera_1` was added to show the relationships that can be deducted after the static population of both the environment and the smart camera information. For instance, we can deduce the `bot:adjacentZone` relationship between both spaces be-

[1] JSON (JavaScript Object Notation) http://www.json.org/

Listing 5.4 – SPARQL query for inserting and defining a smart camera and its field of view.

```
INSERT DATA {
   # Define field of view
   wni:fieldOfView_4 rdf:type owl:NamedIndividual;
                     rdf:type wisenet:FieldOfView.

   # Define smart camera
   wni:smartCamera_4 rdf:type owl:NamedIndividual;
                     rdf:type wisenet:SmartCamera;
                     sosa:isHostedBy inst:space_2;
                     wisenet:ipAddress "10.141.13.26"^^xsd:string;
                     wisenet:hasFieldOfView wni:fieldOfView_4.
}
```

Listing 5.5 – SPARQL query for inserting an algorithm to the smart camera. Example with HOG_SVM algorithm.

```
INSERT DATA {
   # Relate the camera with the algorithm
   wni:smartCamera_4 ssn:implements wisenet:HOG_SVM.
}
```

cause they contain the same door (rule in Listing 4.1); the `wisenet:hasNearbySensor` relationship between both cameras can be deduced due to they are hosted by adjacent spaces (rule in Listing 4.3); the `wisenet:overlap` relationship between both FOVs can be deduces because they show the same door (rule in Listing 4.4 and Eq 4.27); and the relationship `wisenet:observes` that relates the smart cameras with the ROIs shown in their FOV's (Eq. 4.26). As stated previously, all the prefixes, vocabulary, rules and equations defining the WiseNET ontology can be found in Chapter 4.

FACILITATING THE SOFT CALIBRATION PROCESS

Figure 5.6 presents the *System Configuration Interface* (SCI) designed to facilitate the soft calibration process by visualizing and automatically suggesting pertinent elements. The SCI helps specifically to perform the following tasks:

- **Defining a smart camera:** the interface allows to insert the smart camera identification, along with its general information such as the IP address and the algorithms implemented. Moreover, the interface automatically proposes a set of spaces to attach the smart camera to. Those spaces are obtained by querying the `ifcowl` ABox.
- **Setting up ROIs:** the interface allows to manually label the camera image by drawing ROIs and assigning them a representation in the built environment. The ROI's $[x, y, h, w]$ coordinates are automatically obtained from the drawing. Moreover, the system automatically suggests a set of elements, in our case doors, according to the selected space. This information is obtained by querying the `ifcowl` ABox.
- **Extend IFC:** the interface allows to insert extra environment information, for exam-

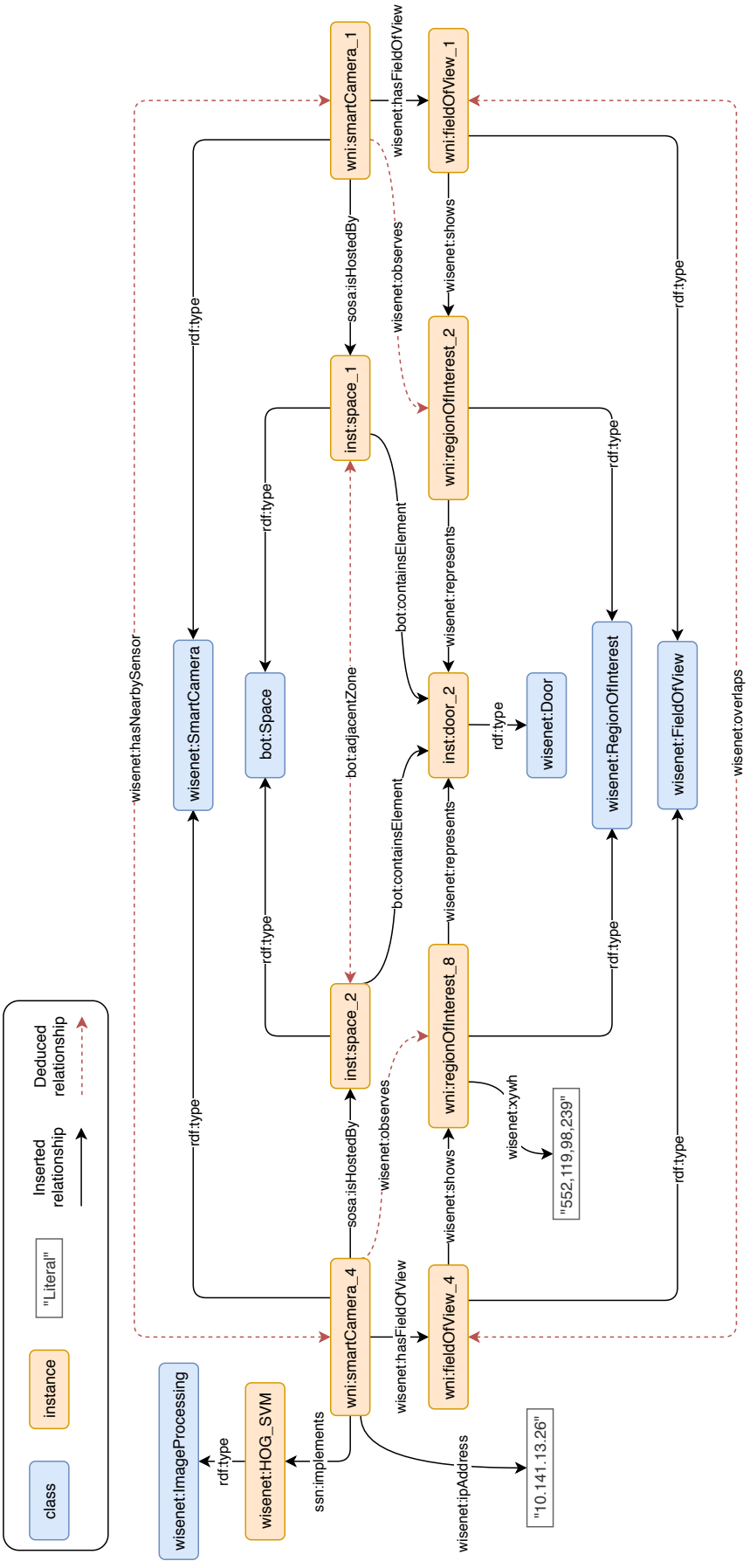


Figure 5.5 – Example of semantic-graph after populating the environment and camera-calibration information. The example considers some information concerning the `smartCamera_1` and `smartCamera_4`.

Listing 5.6 – SPARQL query for inserting and defining a region of interest. Example with `regionOfInterest_8`.

```
INSERT DATA {
   # Define region of interest
   wni:regionOfInterest_8 rdf:type owl:NamedIndividual;
                          rdf:type wisenet:RegionOfInterest;
                          wisenet:xywh "552,119,98,239"^^xsd:string;
                          wisenet:represents inst:door_2.

   # Relate FOV with ROI
   wni:fieldOfView_4 wisenet:shows wni:regionOfInterest_8.
}
```

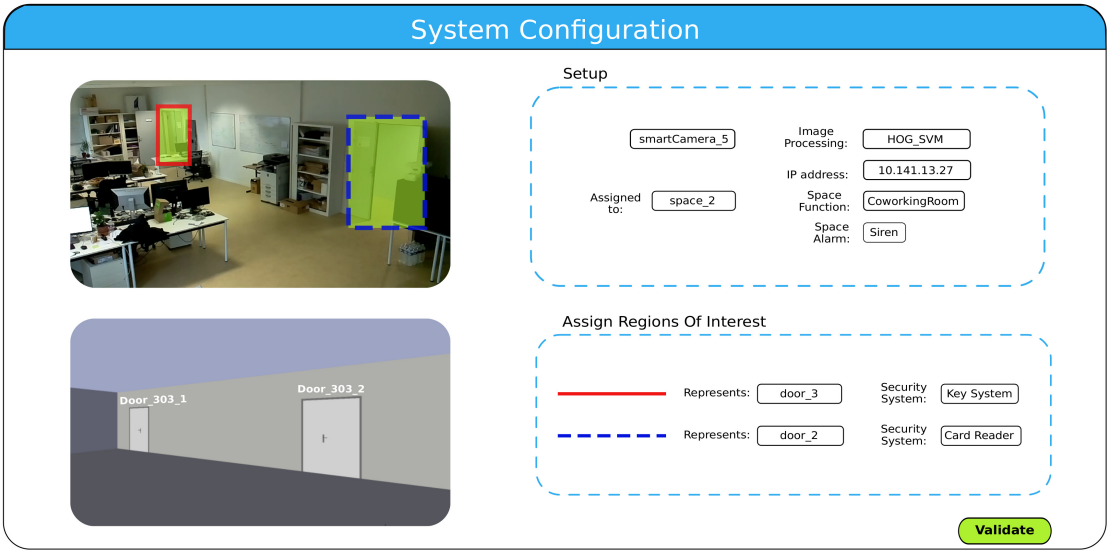


Figure 5.6 – System Configuration Interface. In this example the `smartCamera_5` is being configured to perform `HOG_SVM` algorithm and is being assigned to the `space_2`. Additionally, the `space_2` is being defined as a `CoworkingRoom` with an alarm of type `Siren`. Furthermore, the blue region of interest (located in the top-left image) is assigned to represent the `door_2` which has as security system *Card Reader*, similar for the red region of interest. The 3D view (bottom-left) was obtained from the IFC file.

ple, information about the space functionality, information about the presence and the type of alarm and information about the security systems of a door.

After introducing the information in the interface, the SCI needs to follow the population process presented in Algorithm 1 to insert the information into the WiseNET ontology. However, the connection between the SCI and the population process is still under development.

Furthermore, the SCI presents many advantages such as allowing the definition of the *camera-calibration* information in a semi-automated manner by using a graphical interface, visualizing the semantic-labelling of the ROIs, and using the already populated environment data to automatically suggest pertinent elements.

5.3/ DYNAMIC POPULATION

An IVS system should be able to perceive (accurately) the *dynamic* and evolving data of the environment. In our IVS system configuration, the perception is done thanks to the SCN, where each camera node detects pertinent information using different image processing algorithms, extracts the knowledge from it and send it to the central API (that is in charge of inserting the knowledge into the WiseNET ontology). Furthermore, the WiseNET framework furnishes a vocabulary—in the form of an ontology (see Chapter 4)—that allows each camera node to *express* what is happening in a *textual* manner, instead of (only) sending images. In other words, **we enable cameras to express what they are observing.**

Beyond the work of extracting information from the digital signal constructed by the cameras, our ambition is to deduce new knowledge by gathering all the messages coming from the camera nodes and combine them with other type of knowledge such as time, environment and human-skill rules. In this section, we will present the *dynamic population* procedure, that start with the extraction of knowledge from the camera images, follows by its conversion to a vocabulary understandable by our system, and finish by the processing and insertion of it into the WiseNET ontology. Figure 5.7 illustrates the overall procedures involved in the dynamic population, details of each step will be presented below.

5.3.1/ KNOWLEDGE EXTRACTION

As observed in Fig. 5.7 the knowledge extraction process consists of two procedures, *data extraction* and *image processing*. Both procedures are performed by the SCs.

Data extraction this step consists of extracting/acquiring data. In our case, the SCs observe *a part of* the real world and represent it as a *raw image* format.

Image processing afterwards, image processing algorithms are implemented to *filter the data* and to *extract the knowledge* from the raw image. The **data filtration** step, is similar to the data extraction, however in this step we decide which data to keep based on our interest. For an IVS application, the priority is to detect people—as explained in Chapter 2—thus the SCs perform *person detection algorithms* on the raw image to keep only the parts of the image (set of pixels) that correspond to pertinent information. The *data filtration* can be seen as the *localization* step of an object detector (see Section 2.3). The **knowledge extraction** consists on adding some knowledge to data. In our case, the image processing algorithms assigns some knowledge to the selected image parts. The *knowledge extraction* can be seen as the *classification* step of an object detector (see Section 2.3). The result of the image processing—data filtration + knowledge extraction—is a *processed image*. For example, consider the *processed image* shown in Fig 5.7, this image was obtained by passing the raw image through HOG_SVM person detector (see Section 2.3.1), which firstly selected some image parts (red and blue bounding boxes) and then assigned the knowledge that "those set of pixels corresponds to people". Moreover, the smart camera knew (a priori, thanks to the static population)

that the green bounding box represents a door, therefore the complete knowledge extracted from what the smart camera observes is: "At a specific time, two people were detected and one of them (the one inside the blue box) was around a door".

After extracting the knowledge of what the SCs observe, some processing should be done to translate this knowledge into a "language" understandable to our system, which then will enable its insertion into the WiseNET ontology.

5.3.2/ KNOWLEDGE PROCESSING

The knowledge processing consists of two procedures, *knowledge mapping* and *population* (see Fig. 5.7). Notice that the first procedure is performed by the smart camera node while the second one is performed by the central API.

Knowledge mapping consists on converting a knowledge representation to another one. In this case the knowledge extracted by the smart camera ("two people were detected at a specific time, and one of them was detected around a door") is then converted to the WiseNET ontology representation. This is done by describing the knowledge using the vocabulary defined in the WiseNET ontology. The result of the knowledge mapping is the **smart camera message** (SC-message) defined using the JSON syntax. In our example, the `smartCamera_2` sends the SC-message shown in Listing 5.7, where each field correspond to:

- *deviceID*: identity of the smart camera node that sends the message.
- *inXSDDateTime*: synchronized timestamp, shared by all the SCs. The synchronization was obtained by implementing a Network Time Protocol (NTP) server [117, 156].
- *detections*: array of detections observed in a time instance. In the example, there are two detections made at the same time.
  - *imageProcessing*: computer vision algorithm used to perform the detections. *The system is not depended on a particular computer vision algorithm*, and its choice depends on the application, the resources of the SC nodes and their accuracy. In Chapter 6 we will compare the results using different people detectors presented in the state of the art (see Section 2.3).
  - *class*: type of object detected by the computer vision algorithm.
  - *regionOfInterest*: states if the detection was made around a ROI. A detection is considered around a ROI if: (1) the center of its bounding box is inside the ROI and (2) if the bounding box is at a similar level than the ROI, i.e., if the highest and lowest points of the Bbox and the ROI are around the same height. In the example, one detection—the red bounding box—was made around a `regionOfInterest_5`, which represent the real object `door_4`, while the other detection—the blue bounding box—was not made around any ROI, thus its value is *null*.
  - *xywh*: coordinates of the detection bounding box. These coordinates can be used to project the detections in the building map and to draw them into the image, as shown in the example.

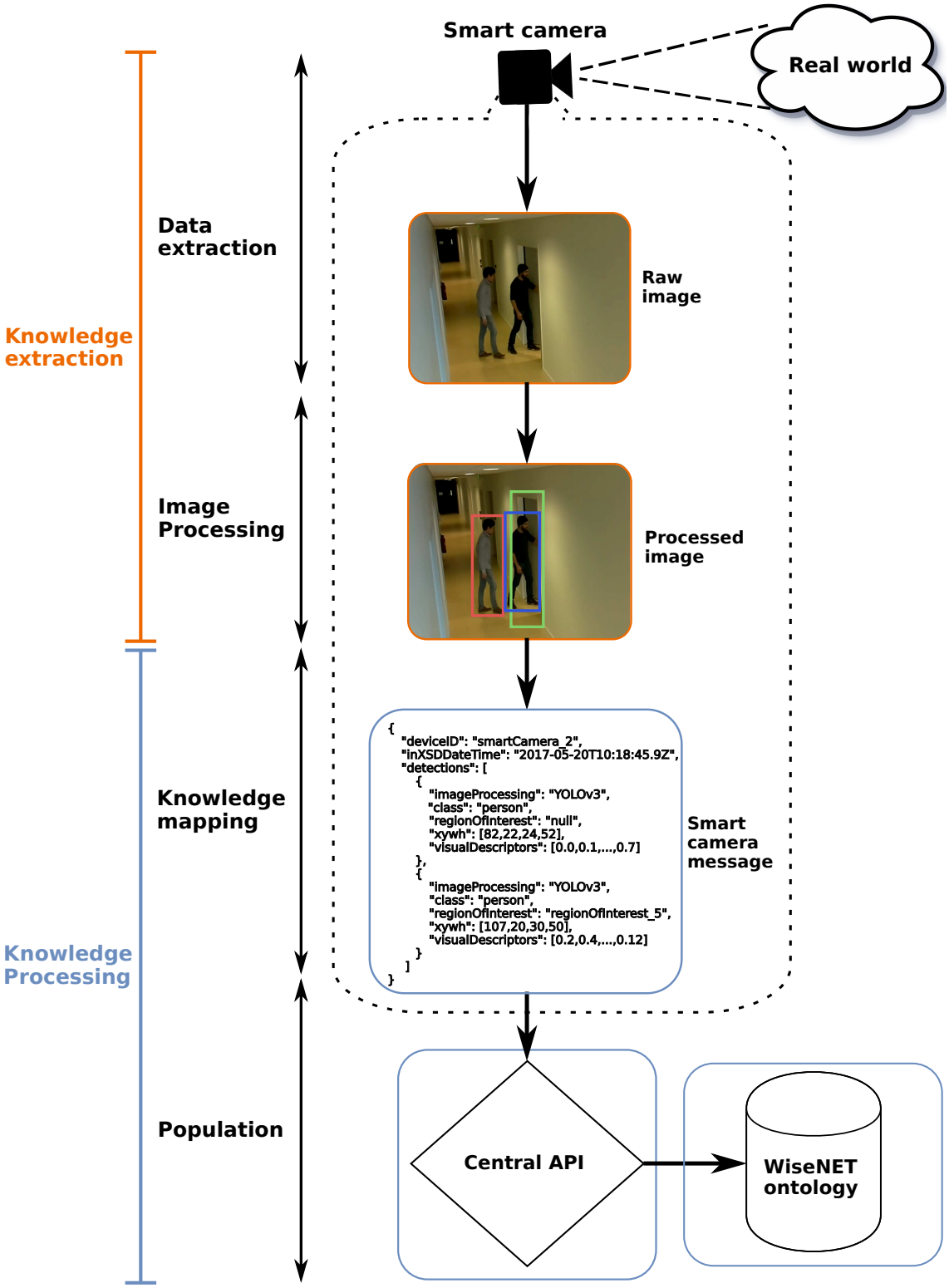


Figure 5.7 – Dynamic population: knowledge extraction and processing. The process starts by *extracting* some data from the real world in the form of a video. Follows by processing the data to extract some knowledge from it, this is done by applying *image processing* algorithms. Then, a *mapping* procedure is performed to express the extracted knowledge with the vocabulary defined in the WiseNET ontology. Finally, the *central API* inserts the resulting knowledge into the *WiseNET ontology* by performing a set of queries.

Listing 5.7 – Example of smart camera message (SC-message) sent to the central API. The message describes what the smart camera is observing in the example presented in Fig 5.7

```
{
  "deviceID": "smartCamera_2",
  "inXSDDateTime": "2017-05-20T10:18:45.922Z",
  "detections": [
    {
        "imageProcessing": "YOLOv3",
        "class": "person",
        "regionOfInterest": "regionOfInterest_5",
        "xywh": [107,20,30,50],
        "visualDescriptors": [0.2,0.4,...,0.12]
    },
    {
        "imageProcessing": "YOLOv3",
        "class": "person",
        "regionOfInterest": "null",
        "xywh": [82,22,24,52],
        "visualDescriptors": [0.0,0.1,...,0.7]
    }
  ]
}
```

– *visualDescriptors*: array of visual features used for describing a detection. *The system is not depended on a particular type of visual features*, and its choice depends on the application and the resources of the SC nodes. In Chapter 6 we will compare the results using different types of visual features presented in the state of the art (see Section 2.3.1).

The final output of the SC is the image knowledge not an image, in this way the privacy of the building users is protected. The addition of semantic meaning to what the camera observes narrows the *semantic gap* between the human interpretation of an image/video an the computer interpretation [169].

Finally, the SC-message—structured in a JSON format—is sent to the central API by using the standard HTTP POST method [57].

Population Once the central API receives the SC-message it creates and inserts the semantic-graph shown in Figure 5.8, for each detection in the message. The graph presented in the figure only concerns the first detection in the SC-message. The semantic-graph extends and combines the knowledge extracted by the SC with knowledge of the environment and knowledge about events—i.e., `Detection` and `PersonInSpace` events.

As explained in Chapter 4, a `Detection` is a type of event that occurs in a specific point in time/space. In the other hand, a `PersonInSpace` event (`PIS-E`) is a container of `Detections` relating a specific `Person` with a specific `Space` during a period of time (time `Interval`). If a `Person` is in a `Space`, the corresponding `PIS-E` is active (open) and `Detections` can be attached to it, however, when the `Person` leaves the `Space` the `PIS-E` is then closed and no more detections can be attached to it. Furthermore, two or more `PIS-Es` are related to each other if they involve the same `Person` instance.

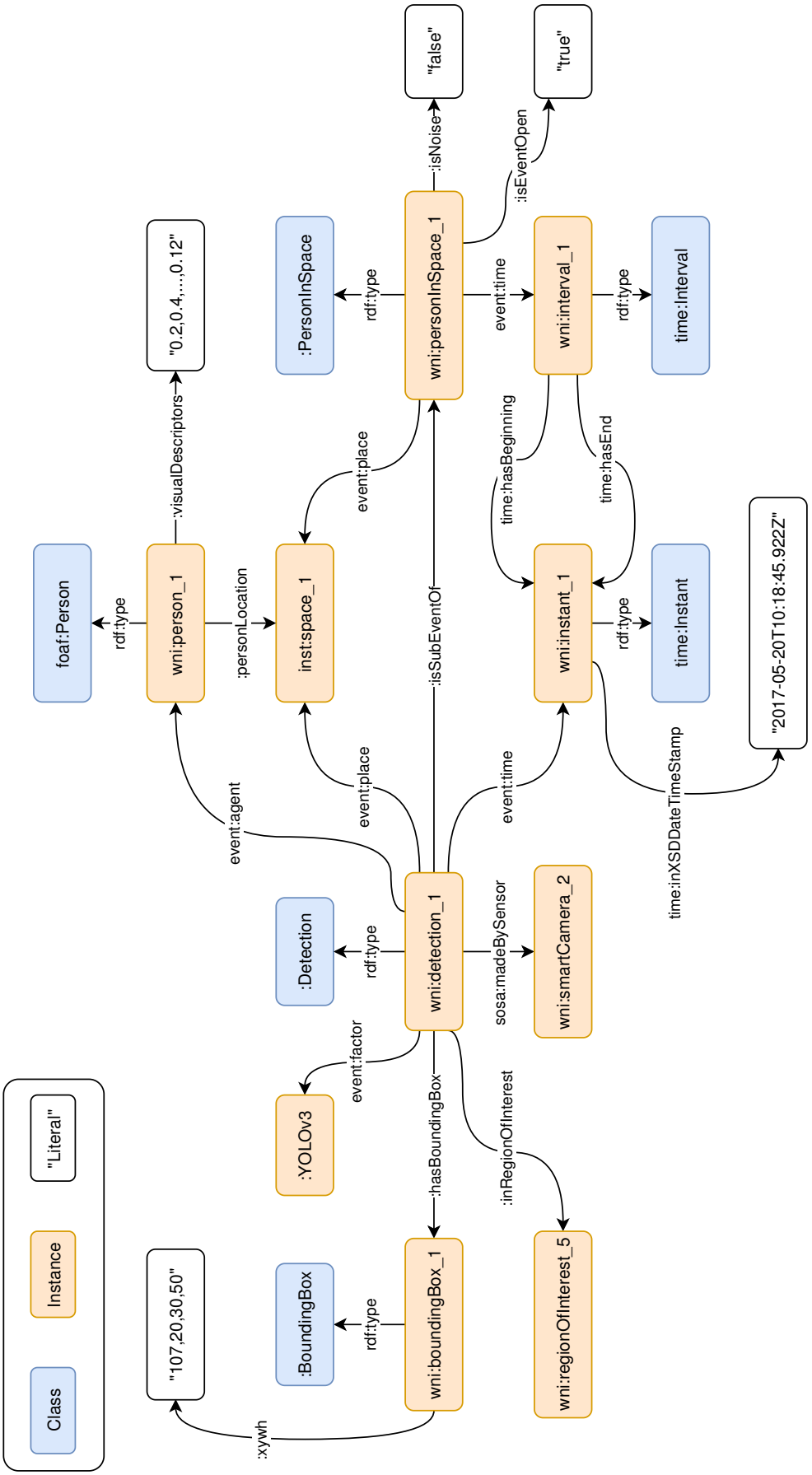


Figure 5.8 – Semantic-graph after populating the first detection of the smart camera message presented in Listing 5.7. For simplicity, the default prefix `":"` corresponds to the `wisenet:` prefix.

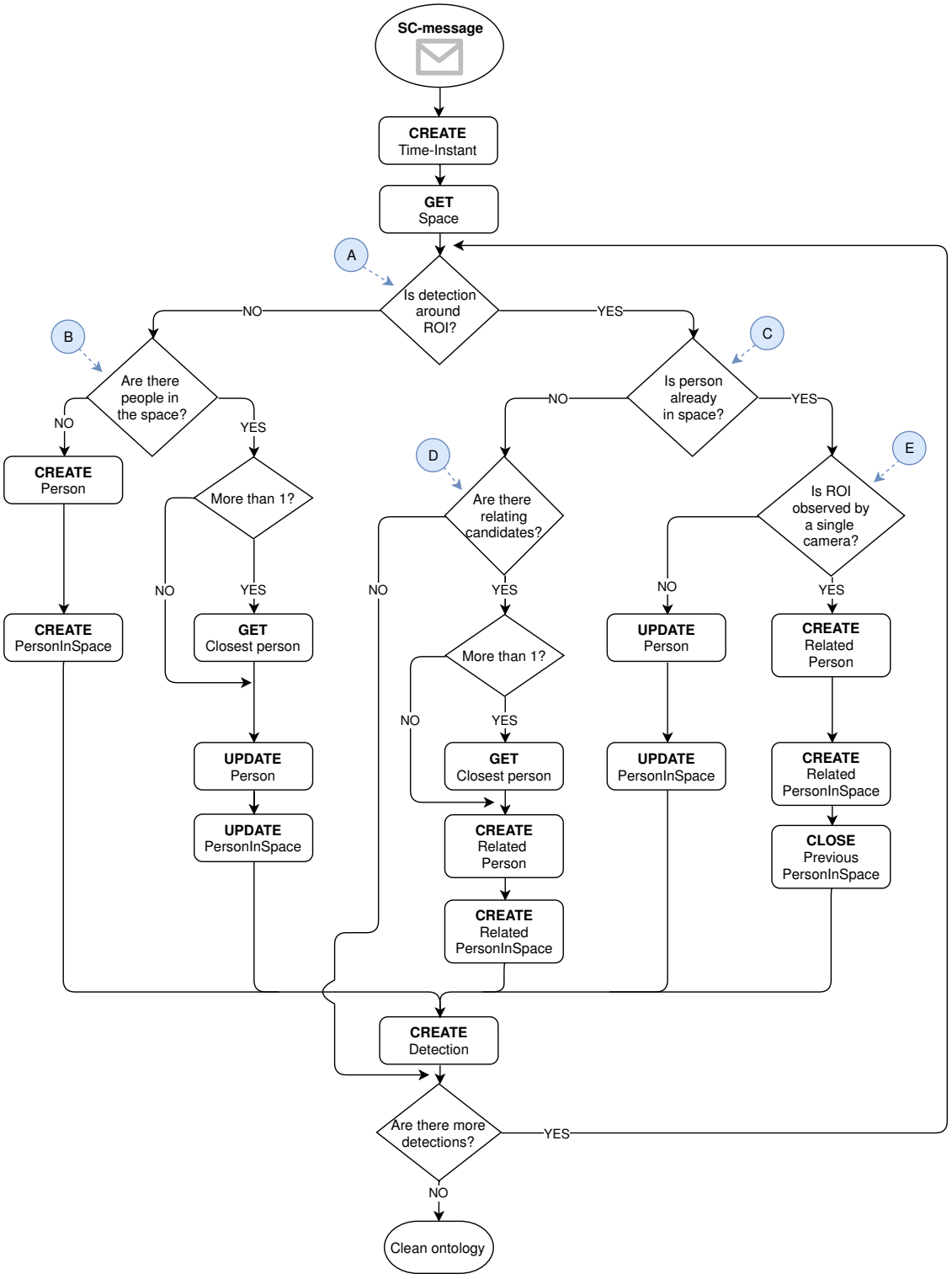


Figure 5.9 – Dynamic population process performed by the central API. The blue points—A, B, C, D, E and F—are used as reference in the text.

To achieve the creation/insertion of the previous semantic-graph, the central API performs the process presented in Figure 5.9. All the CREATE, GET and UPDATE processes, and most of the decision blocks involve SPARQL queries, as it will be presented next.

Listing 5.8 – Query to create a time-instant instance. In this example the `instant_1` is being created.

```
1 INSERT DATA {
2   wni:instant_1 rdf:type owl:NamedIndividual;
3                 rdf:type time:Instant;
4                 time:inXSDDateTimeStamp "2017-05-20T10:18:45.922Z"^^xsd:dateTime.
5 }
```

Listing 5.9 – Query to get the space hosting a sensor. This example gets the space hosting the `smartCamera_2`.

```
1 SELECT ?space
2 WHERE {
3    wni:smartCamera_2 sosa:isHostedBy ?space.
4 }
```

The dynamic population process starts by receiving a SC-message. For exemplification, consider the SC-message presented in Listing 5.7. Then, a time- `Instant` instance is created, by executing the query present in Listing 5.8. Afterwards, the `Space` instance where the situation is happening is obtained by executing the query presented in Listing 5.9. This query makes use of the property `sosa:isHostedBy`, which relates the sensor instance that sent the SC-message with the space where it was installed. In the example, the resulting is `space_1`. For the rest of the process, the resulting space will be considered as the *space of interest*.

Subsequently, the central API takes the first detection of the SC-message and checks if it was done around a ROI (i.e., a door), this corresponds to the **point A** in Fig. 5.9. This is performed by looking the value of the *regionOfInterest* field in the SC-message. If the value is *"null"*, then the detection was not made around a ROI (the "NO" branch of point A, we will refer to it as "*A.NO branch*"), otherwise the detection was made around a ROI (the A.YES branch). For example, the first detection of the SC-message presented in Listing 5.7 was made around a ROI, while the second one was not.

(A.NO branch) if the detection was not made around a ROI, then the central API checks if there are people already in the space of interest **(point B)**. This is performed by executing the query presented in Listing 5.10. The query gets all open `PIS-Es` in the space of interest (Lines 3-5), with their corresponding `Person` instances involved (Line 6) and their visual descriptors (Line 8).

(B.NO branch) if there is nobody in the space, then it means that the detection belongs to a person not seen by the system before, therefore a `Person` and a `PIS-E` instances are created. The `Person` instance is created by executing the query presented in Listing 5.11. The query inserts the person's visual descriptors (Line 4) obtained from the SC-message, and the person's location (Line 5) which corresponds to the space of interest. The `PIS-E` instance is created by executing the query presented in Listing 5.12. The query first creates a time- `Interval` instance (Lines 3-6), with the same beginning and end instant; then a `PIS-E` is created by relating space of interest (Line 11) with the previously created person (Line 12) and time interval (Line 13), moreover, the open and noise boolean properties are set to "true" and "false", respectively (Lines 14 and 15).

Listing 5.10 – Query to get people that have an open PIS-E, in a particular space. This example use the `space_1`.

```
1 SELECT ?person ?descriptors ?event
2 WHERE {
3    ?event rdf:type wisenet:PersonInSpace;
4           wisenet:isEventOpen "true"^^xsd:boolean;
5           event:place inst:space_1;
6           event:agent ?person.
7
8    ?person wisenet:visualDescriptors ?descriptors.
9 }
```

Listing 5.11 – Query to create a person instance. In this example, the `person_1` is being created.

```
1 INSERT DATA {
2    wni:person_1 rdf:type owl:NamedIndividual;
3                 rdf:type foaf:Person;
4                 wisenet:visualDescriptors "0.2,0.4,...,0.12"^^xsd:string;
5                 wisenet:personLocation inst:space_1.
6 }
```

(B.YES branch) if there is somebody in the space of interest, then the new detection should be attach to it, thus the `Person` instance and its corresponding `PIS-E` should be updated. If there is only one person in the space, then the detection corresponds to it, however, if there are more than one people in the space, a comparison of their visual descriptors with the detection's visual descriptors (obtained from the SC-message) is performed to determine which `Person` instance should be updated. The comparison of visual descriptors is performed by using distance metrics such as *cosine-distance* and *bhattacharyya-distance*. Details the distance metrics used, their thresholds and their influence in the results are presented in the evaluation chapter (Chapter 6). After the comparison, the `Person` instance with the closest visual descriptor is updated by executing the query in Listing 5.13. The query-updating process starts by obtaining the current/old person's visual descriptors (Line 7-10), then removing them (Lines 1-3) and finally the inserting the new descriptors in the place of the old ones (Lines 4-6). Moreover, the new descriptors are a naive average between the old visual descriptors and the detection's visual descriptors, this is done to reduce the influence of light changes [160]. Finally, the ending time of the corresponding `PIS-E` is also updated by executing the query in Listing 5.14.

(A.YES branch) If the detection was made around a ROI/door, then multiple scenarios might happen, for example a person can be passing by a door, or a person can be entering or leaving a space (see Fig 5.10). To determine what is happening, the central API will firstly check if the detected person is already in the space of interest or not **(point C)**. This is done by getting the `Person` instances located in the space (query in Listing 5.10) and then comparing their visual descriptors to the detection's visual descriptors.

(C.NO branch) if the detection does not belongs to somebody in the space, then it means that: a person is passing by the door or a person is entering the space. To determine this, the central API checks if there exist a *relating candidate* in neighbouring space, i.e., if

Listing 5.12 – Query to create a time interval and PIS-E instances. In this example, the `interval_1` and the `personInSpace_1` are being created. Lines that start with # are comments.

```
1  INSERT DATA {
2     # Create time interval
3     wni:interval_1 rdf:type owl:NamedIndividual;
4                    rdf:type  time:Interval;
5                    time:hasBeginning wni:instant_1;
6                    time:hasEnd wni:instant_1.
7
8     # Create person in space envent
9     wni:personInSpace_1 rdf:type owl:NamedIndividual;
10                        rdf:type wisenet:PersonInSpace;
11                        event:place inst:space_1;
12                        event:agent wni:person_1;
13                        event:time wni:interval_1;
14                        wisenet:isEventOpen "true"^^xsd:boolean;
15                        wisenet:isNoise "false"^^xsd:boolean.
16 }
```

there is a detection in a neighbour space around the same ROI and around the same time **(point D)**. This is performed by executing the query in Listing 5.15. The query starts by getting the `Door` instance represented by the known ROI (Line 4), then considers all the ROIs representing that door *less* the known ROI (Lines 5-6), the filtration of the known ROI is done to only consider detections from other cameras. Furthermore, the query gets the detections performed around the previously obtained ROIs (Lines 9-10) and only considers those that occurred 2 seconds before the current detection (Lines 11-13), the 2 seconds were chosen empirically. The query gets the `PIS-E` instances containing the previously obtained detections (Line 16), and the `Person` instance involve in it (Lines 17-18). Finally, the query checks if the person candidates are already in the space, if yes then they will be filtered out (Line 19).

(D.NO branch) if there is no relating candidate, then it means that the detection belongs to a person passing by a door, thus no instance will be created or updated. For example, in the *person passing by* scenario in Fig. 5.10, the `smartCamera_3` observes somebody around a door, however in the `smartCamera_2` this person is not around that door, thus the person is just *passing in front of the door*.

(D.YES branch) if there exist a relating candidate, then it means that a person is entering the space. Thus, a `Person` and `PIS-E` instances will be created, related to the candidate. If there exists more than one candidate, then the closest to the detection is obtained. The related `Person` instance is created by executing the query in Listing 5.11 with the addition of line:

```
wni:new_person owl:sameAs wni:candidate_person,
```

which asserts that the newly created `Person` instance refers to the `Person` candidate even if they have different names/identifiers. The related `PIS-E` instance is created by executing the query in Listing 5.12 with the addition of line:

```
wni:new_event wisenet:isRelatedTo wni:candidate_event,
```

which asserts that the newly created `PIS-E` is related to the `PIS-E` candidate. For exam-

Listing 5.13 – Query to update the visual descriptors of a person instance. In this example, the `person_1` is being updated.

```
DELETE {
   wni:person_1 wisenet:visualDescriptors ?descriptors.
}
INSERT {
   wni:person_1 wisenet:visualDescriptors "0.3,0.3,...,0.15"^^xsd:string.
}
WHERE
{
   wni:person_1 wisenet:visualDescriptors ?descriptors.
}
```

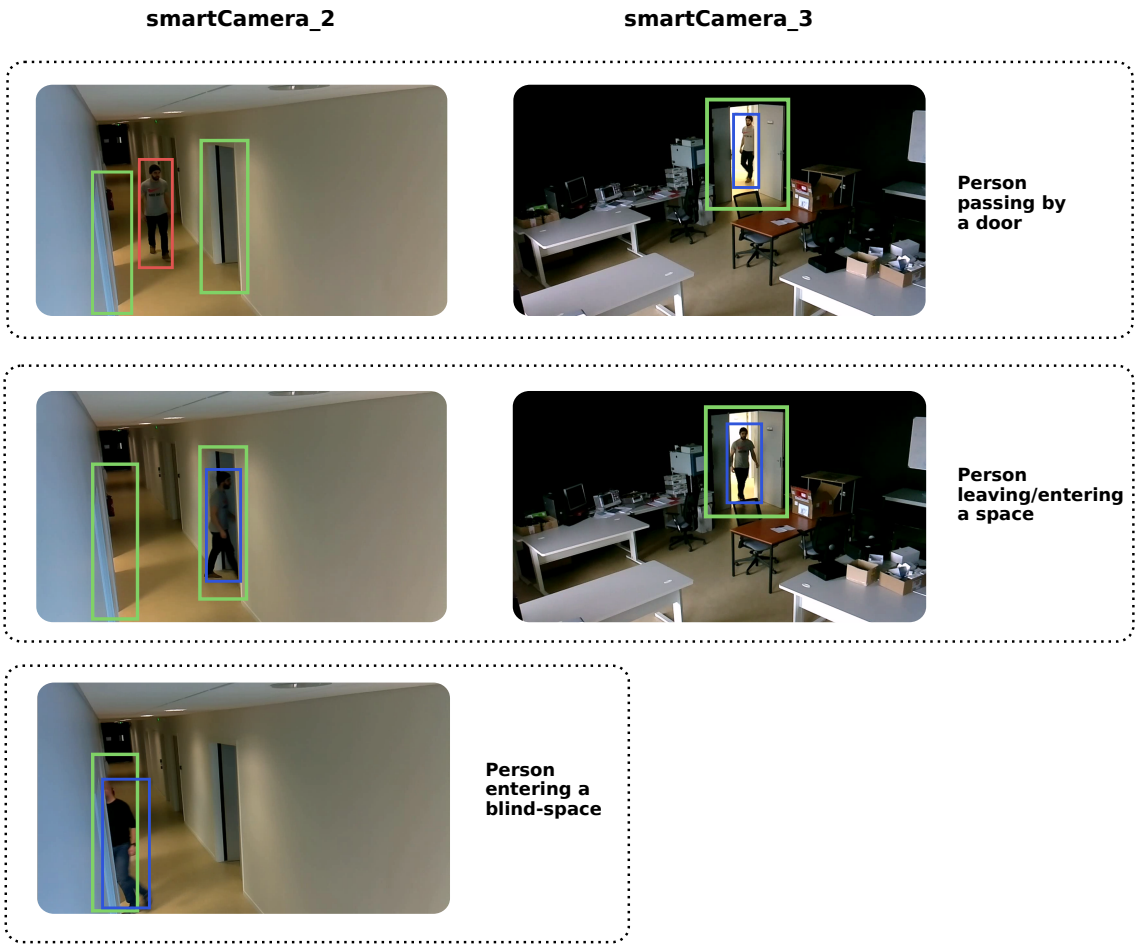


Figure 5.10 – Different scenarios of a person detected around a door. The green boxes represent Regions Of Interests (ROIs), while the blue and red boxes represent, respectively, detections made around and not around a ROI. In the first case, the person is just *passing by the door* (based on `smartCamera_3`, because it is not being detected around the door in the `smartCamera_2`. In the second case, the detected person is *leaving* the `space_1` (the space hosting `smartCamera_2`) and at the same time *entering* the `space_3`. In the final case, the person is entering the `space_6`, which does not have any cameras (*blind-space*) (see Fig. 5.1).

Listing 5.14 – Query to update the time interval of a person in space event. In this example, the `personInSpace_1` is being updated.

```
1  DELETE {
2     ?timeInterval time:hasEnd ?instantEnd.
3  }
4  INSERT {
5     ?timeInterval time:hasEnd wni:instant_2.
6  }
7  WHERE {
8     wni:personInSpace_1 event:time ?timeInterval.
9     ?timeInterval time:hasEnd ?instantEnd.
10 }
```

ple, in the *person entering* scenario in Fig. 5.10, the `smartCamera_3` observes somebody around a door and a relating candidate was found (person observed around the same door by `smartCamera_2`), thus new `Person` and `PIS-E` instances will be created related to the candidate in the `space_2`.

(C.YES branch) if the detection belongs to somebody in the space, it means that the person is leaving the space **(point E)**. The person can be going to a space where it will be observed by another camera or to a space where there is no cameras, referred as *blind-space*. To determine if the person is going to a blind-space, the central API checks if the ROI is observed by single camera, by executing the query presented in Listing 5.16. The query gets the spaces containing the door represented by the ROI (Lines 3-4), and then filters out the spaces hosting a camera (Lines 5-8), thus resulting in the blind-space.

(E.NO branch) if the detection was made around a ROI which is observed by multiple cameras, then it means that the person is not entering a blind-space, thus the `Person` and `PIS-E` instances are simply updated by executing the queries presented in Listings 5.13 and 5.14, respectively.

(E.YES branch) if the detection was made around a ROI observed by a single camera, then it means that the person is about to enter a blind-space. Therefore, a new `Person` and `PIS-E` instances will be created in the blind-space. These new instances will be related to the `Person` and `PIS-E` in the current space (result of point C). Moreover, the "old" instances need to be closed/updated to avoid multiple creation of instances in the blind-space, due to subsequent detections. The updating is performed by executing the query presented in Listing 5.17, where the "old" `PIS-E` is being closed (Line 6) and the "old" `Person` is being removed from the current space (Line 3). For example, consider the *blind-space* scenario in Fig. 5.10, where a person is entering the `space_6` and leaving the `space_1`, thus new `Person` and `PIS-E` instances will be created in `space_6`, while the `Person` and `PIS-E` instances in `space_1` will be closed and removed.

The dynamic population process continues by creating and inserting a `BoundingBox` and `Detection` instances. This is done by executing the query in Listing 5.18, where Lines 3-5 create the `BoundingBox` while the rest create a `Detection` and relate it to: the time `Instant` when it occurred (Line 10), the `PIS-E` that contains it (Line 11), the `SmartCamera` that performed it (Line 12), the `Person` to which the detection belongs to (Line 13), the `Space` where it happened (Line 14), the `ImageProcessing` algorithm used to obtain it (Line 15), the previously created that `BoundingBox`, and finally if the detection was made around a ROI then it will also be related to it (Line 17).

Listing 5.15 – Query to relate candidates (events and people) from other spaces.

```
1  SELECT DISTINCT ?event ?person ?descriptors
2  WHERE {
3     # Get Door instance and all other ROIs that represent it
4     wni:regionOfInterest_5 wisenet:represents ?door.
5     ?roi wisenet:represents ?door.
6     FILTER(?roi != wni:regionOfInterest_5)
7
8     # Get detections around the same time and around the same door
9     ?detection rdf:type wisenet:Detection;
10                wisenet:inRegionOfInterest ?roi;
11                event:time ?timeInstant.
12    ?timeInstant time:inXSDDateTimeStamp ?detectionTime.
13    FILTER (?detectionTime > "2017-05-20T10:18:43.922Z"^^xsd:dateTime)
14
15    # Get their PIS_E, their person and their descriptors
16    ?detection wisenet:isSubEventOf ?event;
17                event:agent ?person.
18    ?person wisenet:visualDescriptors ?descriptors.
19    FILTER NOT EXISTS {?person wisenet:personLocation inst:space_1}
20 }
```

After creating the `Detection` instance, the process is repeated for each detection in the SC-message.

Finally, the central API performs a cleaning process which consists of detecting noisy events and closing events:

- **Detect noisy events:** generally, false detections (i.e., wrong detections) occurred randomly, caused by light changes, and they last for few seconds. Therefore, events are considered "noisy" if they were generated by and contain false detections. In that manner, the central API tags events as noisy if they have not being populated for certain time and they only contain few detections. To perform this, the central API executes the query presented in Listing 5.19. The query first gets all the events which last detection occurred 2 seconds before the current time, along with the number of detections contained (Lines 12-28). If the number of detections is lower than a threshold (Line 30), then the event will be closed and considered as noisy (Lines 1-9). The minimum number of detections was empirically set to 40, meaning that the person was detected for at least few seconds.

- **Closing events:** an event is closed if its last detection occurred some time ago and this detection was made around a door, i.e., if the person left the space. The central API close events by performing the query presented in Listing 5.20, which Lines 9-17 get the last detection of events, Line 18 checks if that detection was made around a ROI and Lines 19-22 filter those events which last detection occurred 2 seconds before the current time. Moreover, Lines 1-7 close the `PIS-E` and removes its related `Person` instance from the `Space`. This way of closing events allows to leave open events where the related person has not been "seen" leaving the space, for example the case of a person being outside the FOV of a camera.

Notice that the central API executes the dynamic population process by demand, it does not run in the background, *i.e.*, that it only runs each time a SC-message is received.

Listing 5.16 – Query to check if a door is contained by a space without a camera.

```
SELECT DISTINCT ?spaces
WHERE {
   wni:regionOfInterest_5 wisenet:represents ?door.
   ?spaces bot:containsElement ?door.
   ?spaces_withCameras bot:containsElement ?door.
   ?camera rdf:type wisenet:SmartCamera;
           sosa:isHostedBy ?spaces_withCameras.
   FILTER (?spaces_withCameras != ?spaces)
}
```

5.3.3/ USE CASE

To better observe the different steps performed by the central API, consider the scenario depicted in Fig. 5.11, where two people are walking in the I3M building. The SCN extracts the pertinent information from the scene, and then sends the SC-messages to the central API. The central API processes those messages and generates the new knowledge that will be inserted into the ontology by following the diagram shown in Fig. 5.9. Table 5.2 shows the dynamic population performed by the central API at each timestamps. For the sake of clarity, the following information was omitted in the table:

- The first time a person is detected in a new space a `Person` and `PIS-E` instances are created (timestamps 1, 3, 6 and 8). The `PIS-E` instance is created in the space where the detection was performed.
- For each detection performed by a SC, a `Detection` instance is created and attached to a `Person` instance and to a `PIS-E` instance.
- If a detection is made around an door (i.e. a ROI), then the new `PIS-E` may be related to an existing `PIS-E` (timestamps 3, 6 and 8), meaning that they involve the same `Person` instance.
- The time of the use case is continuous, meaning that between each timestamps there exists many others. For example between timestamps 3 and 4 there is a detection (at timestamp 3.5 for example) that explains the creation of `person_4` and `PIS-E_4`, which was omitted for conciseness.
- Each time a `Detection` instance is attached to a `Person` instances, then the `Person` instance is updated. In the same manner, each time a `Detection` instance is defined as sub-event of a `PIS-E` instance, then the `PIS-E` is updated.

At timestamp 1, both people are observed by the first time, thus two `Person` and `PIS-E` instances are created. At timestamp 2, the detections are not around a door and the people are already in the space, thus the previously created instances are updated. At timestamp 3, the `detection_5` was made around a door and a relating candidate was found, thus a new `Person` and `PIS-E` instances are created in the new space. At time stamp 4, the detections are not around a door and the people are already in the space, thus they are updated. At timestamp 5, the second person is not being detected however, notice that at timestamp 6 no new `Person` instance was created, this is because

Listing 5.17 – Query to close/update a PIS-E which was related to a blind-space. In this example, the `personInSpace_1` event is being updated.

```
DELETE {
   wni:personInSpace_1 wisenet:isEventOpen "true"^^xsd:boolean.
   ?person wisenet:personLocation ?space.
}
INSERT {
   wni:personInSpace_1 wisenet:isEventOpen "false"^^xsd:boolean.
}
WHERE {
   wni:personInSpace_1 event:place ?space;
                       event:agent ?person.
 }
```

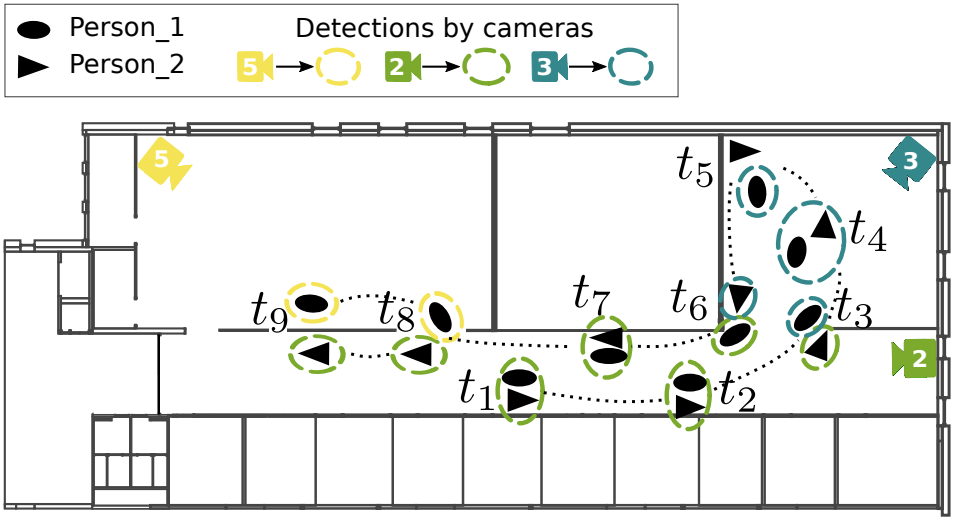


Figure 5.11 – Dynamic population use case. Two people are walking in a built environment while being detected by a SCN. Nine timestamps are being highlighted in the scenario. Notice that at timestamp 5, `Person_2` is not being detected. The dynamic population at each timestamp can be found in Table 5.2.

the system deduced that the person *has not left the space*. At timestamp 6, both detections—`detection_10` and `detection_11`—are made around a door. However, a relating candidate was found only for `detection_10`, thus no new `PIS-E` was created for `detection_11`. At timestamp 7, the detections are not around a door and the people are already in the space, thus they are updated. At timestamp 8, the `detection_14` was made around a door and a relating candidate was found, thus a new `Person` and `PIS-E` instances are created in the new space. Finally, at timestamp 9, the detections are not around a door and the people are already in the space, thus they are updated.

The dynamic population was performed in a machine with the following configuration: Intel Core i7-4790 CPU @3.6GHz × 4, 16GB of RAM and a "Java Heap" size set to 200MB.

The time required for the central API to process the SC-messages at each timestamp is shown in the Table 5.2. From that processing time, it can be observed that the central API requires more time to process related detections (i.e., detections which are around

Listing 5.18 – Query to create a bounding box and detection instances. In this example, the `boundingBox_1` and the `detection_1` are being created.

```
INSERT DATA {
   # Create bounding box
   wni:boundingBox_1 rdf:type owl:NamedIndividual;
                     rdf:type wisenet:BoundingBox;
                     wisenet:xywh "107,20,30,50"^^xsd:string.

   # Create detection
   wni:detection_1 rdf:type owl:NamedIndividual;
                   rdf:type wisenet:Detection;
                   event:time wni:instant_1;
                   wisenet:isSubEventOf wni:personInSpace_1;
                   sosa:madeBySensor wni:smartCamera_2;
                   event:agent wni:person_1;
                   event:place inst:space_1;
                   event:factor wisenet:YOLOv3;
                   wisenet:hasBoundingBox wni:boundingBox_1;
                   wisenet:inRegionOfInterest wni:regionOfInterest_5.
}
```

a door, like in t_3, t_6 and t_8) compared to detections which are not around a region of interest. Furthermore, we can observe that the processing time of a SC-message with a single detection (t_5) is lower than the rest.

Listing 5.19 – Query to detect and close noisy events.

```
DELETE {
   ?event wisenet:isNoise "false"^^xsd:boolean;
          wisenet:isEventOpen "true"^^xsd:boolean.
   ?person wisenet:personLocation ?space.
}
INSERT {
   ?event wisenet:isNoise "true"^^xsd:boolean;
          wisenet:isEventOpen "false"^^xsd:boolean.
}
WHERE {
   {
      SELECT ?event (count(?detections) as ?numDetections)
      WHERE {
         ?event rdf:type wisenet:PersonInSpace;
                wisenet:isEventOpen "true"^^xsd:boolean;
                event:agent ?person;
                event:place ?space;
                event:time ?timeInterval.
         ?timeInterval time:hasEnd ?endInstant.
         ?endInstant time:inXSDDateTimeStamp ?endTime.

         ?detections wisenet:isSubEventOf ?event.
         ?lasDetection wisenet:isSubEventOf ?event;
                       event:time ?timeInstant.
         ?timeInstant time:inXSDDateTimeStamp ?endTime.
         FILTER (?endTime < "2017-05-20T10:18:43.922Z"^^xsd:dateTime)
      }
      GROUP BY (?event)
   }
   FILTER (?numDetections < "40"^^xsd:int)
}
```

Listing 5.20 – Query to close events.

```
DELETE {
   ?event wisenet:isEventOpen "true"^^xsd:boolean.
   ?person wisenet:personLocation ?space.
}
INSERT {
   ?event wisenet:isEventOpen "false"^^xsd:boolean.
}
WHERE {
   ?event rdf:type wisenet:PersonInSpace;
          wisenet:isEventOpen "true"^^xsd:boolean;
          event:agent ?person;
          event:place ?space;
          event:time  ?timeInterval.
   ?timeInterval time:hasEnd ?endInstant.
   ?endInstant time:inXSDDateTimeStamp ?endTime.

   ?lastDetection wisenet:isSubEventOf ?event;
                  wisenet:inRegionOfInterest ?roi;
                  event:time ?timeInstant.
   ?timeInstant time:inXSDDateTimeStamp ?endTime.

   FILTER (?endTime < "2017-05-20T10:18:43.922Z"^^xsd:dateTime)
}
```

Table 5.2 – Dynamic population performed during the people tracking scenario shown in Fig 5.11. The time denotes the processing time required by the central API to perform the dynamic population.

	Dynamic population	Time (s)
t_1	person_1 created detection_1 created attached to: person_1 subEventOf: PIS-E_1 PIS-E_1 created attached to: person_1	0.147
	person_2 created detection_2 created attached to: person_2 subEventOf: PIS-E_2 PIS-E_2 created attached to: person_2	
t_2	detection_3 created attached to: person_1 subEventOf: PIS-E_1	0.161
	detection_4 created attached to: person_2 subEventOf: PIS-E_2	
t_3	detection_5 created around: door_4 attached to: person_3 subEventOf: PIS-E_3 person_3 created same as: person_1 PIS-E_3 created related to: PIS-E_1	0.623
	detection_6 created attached to: person_2 subEventOf: PIS-E_2	
t_4	detection_7 created attached to: person_3 subEventOf: PIS-E_3	0.158
	detection_8 created attached to: person_4 subEventOf: PIS-E_4	
t_5	detection_9 created attached to: person_3 subEventOf: PIS-E_3	0.068
t_6	detection_10 created around: door_4 attached to: person_5 subEventOf: PIS-E_5 person_5 created same as: person_3 PIS-E_5 created related to: PIS-E_3	0.657
	detection_11 created around: door_4 attached to: person_4 subEventOf: PIS-E_4	
t_7	detection_12 created attached to: person_5 subEventOf: PIS-E_5	0.166
	detection_13 created attached to: person_6 subEventOf: PIS-E_6	
t_8	detection_14 created around: door_3 attached to: person_7 subEventOf: PIS-E_7 person_7 created same as: person_5 PIS-E_7 created related to: PIS-E_5	0.635
	detection_15 created attached to: person_6 subEventOf: PIS-E_6	
t_9	detection_16 created attached to: person_7 subEventOf: PIS-E_7	0.172
	detection_17 created attached to: person_6 subEventOf: PIS-E_6	

5.4/ CONCLUSION

In this chapter we presented a set of processes, developed for automatically populating an ontology with static and dynamic contextual data.

The proposed frameworks are part of an Artificial Intelligence (AI) system, WiseNET, that draw conclusions based on observations and context. This AI creates semantic-links between different knowledges in order to take decisions. Thus, allowing humans to directly design and understand the different algorithms for intelligent decision making, for example the dynamic process performed by the central API (Fig 5.9). i.e., it is a transparent AI.

Moreover, thanks to this "transparency", semantic-based models allows to pinpoint the cause of errors (unexpected results/ wrong decisions) and to solved them by modifying the semantic-links. This is a big advantage compared to deep-learning AI models which are "black boxes", where (for the moment) is almost impossible to know which part of the network is causing an unexpected decision. However, the main drawback of semantic-based models is that each situation of interest should be defined "manually", for example, the "person in blind-space" and the "person passing by a door" situations presented before. In contrast, deep-learning models could (probably) deduce, by using big amount of data, many of the situations "manually" defined, however, their definition can not be controlled by humans.

Furthermore, we developed an innovative method to constantly insert data into an ontology—which are normally used with static data—by using an API bridge between the data sources and the ontology. The API uses multiple semantic web technologies which enable the interaction with the ontology. We applied the processes into an smart building problem.

Static population the proposed static population process, consisted in inserting into an ontology the knowledge that (normally) stays unchangeable, such as information of the environment and the calibration information of a sensor network.

The proposed *IFC to WiseNET process* (Fig. 5.3), allows to automatically extract pertinent information from an IFC file and insert it into the WiseNET ontology by using a simple and intuitive structure. The process is based on semantic web technologies, specifically it uses a set of queries for extracting and inserting the pertinent information.

The *camera-calibration process* (Algorithm 1), is a sequence of queries that enables the automatic insertion of the camera-calibration information, such as its location, the computer vision algorithms it performs and the regions of interest it observes.

As the building structure and the location of the cameras remain (normally) unchangeable after its construction/installation, the population of the static information needs to be performed once at the initialization of the system. However, if the building is modified (e.g., if it is extended, or a door or a space is added) the environment population should be re-performed using the new IFC file, or the new elements should be manually inserted in the ontology by executing the population query (Listing 5.2). In a similar manner, to add a different sensor or modified an existing one, its calibration file should be defined and then simply inserted by following the Algorithm 1.

Furthermore, the processes were designed in a modular and generic way, meaning that

they are not dependent of the previous modules. For example, if the IFC file of a building is not available (which is the case for most of old buildings), the environment knowledge can still be inserted using our framework by manually executing the population query (Listing 5.2). However, if there is a high amount of information a more suitable solution should be considered.

Is important to notice that we use as a running example the I3M building and the network of smart cameras deployed in it, however, **the frameworks are not dependent of the building environment nor the network of cameras**. Thus, it could be applied to any type of environment equipped (or not) with any type of sensors.

Dynamic population the proposed dynamic population process, consisted in automatically extracting, structuring and inserting the data observed by each camera node, into the WiseNET ontology. In a few words, it enables smart cameras to express, in a structured manner, what they are looking. Moreover, **the dynamic population enables the interaction between knowledge extracted by computer vision algorithms with contextual data**. The dynamic population process is divided into knowledge extraction and processing (see Fig. 5.7).

The *knowledge extraction* consists in acquiring the image and performing some image processing algorithm in it, to give some knowledge to (understand) what the camera is observing.

The *knowledge processing* consists in representing the extracted knowledge using the vocabulary understood by the WiseNET system and then inserting it according to the current and previous information in the system by following the diagram presented in Fig. 5.9. The process focuses in the information if a detection was made around a door (ROI), because this information determines if a person is entering/leaving a space, and can also be used for person re-identification between multiple cameras. Which are principal knowledges on a intelligent video surveillance systems.

The result of the dynamic population is the insertion of a semantic-graph relating information of the smart camera, with the detection performed, the events generated, the person involved and time (see Fig 5.8).

Regarding the privacy protection, it is important to remark that the SCN used in the WiseNET system does not send or save any images, thereby protecting the privacy of the individuals. However, one exception could be made if the ontology infers that an illegal act is occurring, in which case, the central API can use that inferred knowledge to send a message to the SC telling it to start recording and to save the images locally (to have them as proof).

This chapter defined the ways for automatically inserting data into the WiseNET system. In the next chapter, the system will be populated with real data from a smart camera network, which will allow to evaluate its performance.

Remark. *Most of the information presented in this chapter was validated in the following papers [111, 112].*

6

Dataset and Evaluations

The main motivation of a Intelligent Video Surveillance (IVS) system is to ease the management and monitoring of an environment. This can be done by filtering the data, and providing just the pertinent information about the building structure, the elements contain in it, and about the building usage (i.e., what has happened and what is happening). In this section, we will validate that our semantic-based system (WiseNET) is able to provide all these informations in an intuitive way.

Datasets play an extremely important role in validating a newly proposed system or algorithm. Therefore, Section 6.1 will present the dataset used during the complete validation procedure. Furthermore, the evaluation procedure is articulated in two parts. The first one, presented in Section 6.2.1, consist in a proof of concept of the system by using perfect detection conditions. In this part, the kernel of our semantic-based system (i.e., the ontology) will be evaluated, as well as the system's performance on tracking people. The second part, presented in Section 6.3, consist in going beyond the proof of concept, and evaluating the system performance and limits by using non-ideal detectors.

Remark. *All the CQs, prefixes, vocabulary, rules and equations used and referred to in this section were defined in Chapter 4.*

6.1/ Multi-camera multi-space datasets

Dataset are important in the development and evaluation of a system. In contrast to real-time data (i.e., not stored), a dataset allows to test different configurations of the systems, using the same data-conditions. Furthermore, a dataset provides a baseline, which can be used to evaluate the performance of an algorithm. The best baseline is the "truth", which in computer vision is referred "ground truth".

As presented previously, we are interesting in combining the information coming from a camera network, with contextual information of the environment. Therefore, we require a dataset that includes all the necessary information.

Nowadays, camera networks are part of our every-day life environments, consequently, they represent a massive source of information for monitoring human activities and to propose new services to the building users. To perform human activity monitoring, people have to be detected and the analysis has to be done according to the information relative to the environment and the context. Available multi-camera multi-space datasets furnish videos with few (or none) information of the environment where the network was

deployed. Thus, we developed the WiseNET dataset, which provides multi-camera multi-space video sets along with the complete contextual information of the environment.

This section is articulated in two parts. The first part presents some existing multi-camera datasets, while the second part presents the WiseNET dataset.

6.1.1/ EXISTING DATASETS

The existing multi-camera datasets can be categorized by the scene they observe—indoor, outdoor or both—by the type of data provided—full-frames or only cropped images—by their main application—people detection, re-identification, tracking or activity recognition—and even by the distribution of the cameras in the environment—cameras located in a single or multiple spaces.

In this section, we briefly summarize some publicly available datasets deployed in *multiple indoor spaces* and that provide *full-frame* data. Table 6.1 summarizes the relevant datasets. For each dataset we indicate the number of video sets (#Sets), cameras (#Cams), people (#People) and the annotation method used for obtaining the bounding boxes: manually (hand) or automatically (auto) methods. We also indicate the presence of contextual data, such as environmental data and scene semantic information. For environmental data, we considered the inclusion of: 2D schema (2D), 3D schema (3D) or extra environmental data (E+), for instance, the topology of the environment and information about the different elements contained in the spaces. For scene semantic information, we considered the presence of any semantic labeling, for instance the labeling of enter/exit regions. Moreover, each dataset was annotated with the following challenging attributes: view-point variations (VV)—target appears at different poses and viewpoints—illumination variations (IV)—changes on light conditions—detection errors (DE)—occurs (mostly) when detections are performed by an automatic method—occlusion (OCC)—target totally or partially non-visible—and background clutter (BC)—background filled with targets.

CAVIAR [58] was acquired using two cameras with overlapping Fields Of View (FOVs). It consists of 26 small video sets involving 72 people and manually labeled bounding boxes. CAVIAR suffers from viewpoint variations due to the position of the cameras. V47 [173] was constructed from two indoor cameras with overlapped FOVs. It consists of 47 people walking in two directions (in and out). However, no contextual information is provided. V47 suffers from viewpoint variations and partial occlusion. SAIVT [13] was collected by eight surveillance cameras. It consists of ten video sets involving 152 people. The annotations were performed manually and a 2D schema of the environment is provided. SAIVT suffers from viewpoint variation, illumination variation and background clutter. Dana36 [136] was acquired from 36 camera views in a mixed scene—27 cameras outdoors and nine indoors. It consists of 15 people following predefined activities around a building. Moreover, no contextual information is provided. Dana36 suffers from viewpoint and illumination variations, as well as from partial occlusion. HDA+ [123] consists of a single 30-minutes long video set recorded using 13 indoor cameras. The dataset involves 85 people moving around three floors of a building. HDA+ provides both manually and automatically annotated bounding boxes. The automatic annotations were obtained using the Aggregated Channel Features (ACF) people detector [46]. HDA+ also provides a 2D schema of the environment. Moreover, HDA+ suffers from viewpoint and illumination variations, as well as from occlusion and detection errors due to the automatic people detector. NLPR [31] consists of four subsets from which we consider the third one (NLPR-3)

Table 6.1 – Characteristics of indoor multi-camera multi-space datasets. People appearing at multiple video sets are considered as different people.

Dataset	#Sets	#Cams	#People	Annotation	Environment	Semantics	Attributes	Year
CAVIAR	26	2	72	hand	✗	✗	VV	2004
V47	2	2	47	hand	✗	✗	VV,OCC	2011
SAIVT	10	8	152	hand	2D	✗	VV,IV,BC	2012
Dana36	1	36[†]	15	hand	✗	✗	VV,IV,OCC	2012
HDA+	1	13	85	hand/auto	2D	✗	VV,IV,OCC,DE	2014
NLPR-3	1	4	14	hand	2D	✓	VV,IV,OCC	2015
CamNeT	4	8[‡]	50	hand	2D	✗	VV,IV,OCC	2015
WiseNET	11	6	77	hand/auto	2D,3D,E+	✓	VV,IV,OCC,BC,DE	2019

[†] 9 indoors.
[‡] 5 indoors.

due to the indoor scene. NLPR-3 has a single video set, which captures 14 people using four cameras installed at different spaces. The datasets provides a 2D schema of the environment and semantic information about the enter/exit regions of each camera scene. CamNeT [191] is a mixed environment dataset composed of three outdoor cameras and five indoors. It involves 50 people and it provides the 2D schema of the environment. CamNeT suffers from viewpoint and illumination variations, as well as from occlusion.

Finally, the WiseNET dataset provides 11 video sets recorded using 6 indoor cameras deployed on multiple spaces. The video sets represent more than one hour of video footage, include 77 people tracks and captured different human actions such as walking around, standing/sitting, motionless, entering/leaving a space and group merging/splitting. Moreover, each video has been manually and automatically annotated to include people detection and tracking meta-information. The automatic people detection annotations were obtained by using different complexity and robustness detectors, from machine learning to state-of-art deep convolutional neural network (CNN) models. Concerning the contextual information, the Industry Foundation Classes (IFC) file that represents the environment's Building Information Modeling (BIM) data provided, as well as semantic information relating real objects with enter/exit zones (i.e., doors). The BIM/IFC file describes the complete structure of the environment, it's topology and the elements contained in it, moreover, it can be used to generate 2D and 3D schema. Furthermore, the WiseNET dataset provides all challenging attributes, making it an interesting candidate for benchmarking people detection and tracking algorithms.

To our knowledge, the WiseNET dataset is the first to provide a set videos along with the complete information of the environment. More details about the WiseNET dataset will be given in the next section.

6.1.2/ WISENET DATASET

The WiseNET dataset was created using an indoor network composed of 6 Smart Cameras (SCs) deployed on the third floor of the Institut Marey et Maison de la Métallurgie (I3M) building located in Dijon, France [113]. The dataset consists of three main elements: (1) video sets, (2) information of the environment and (3) annotations for people detection and people tracking.

1. The video sets were recorded using from 5 to 6 cameras simultaneously. The videos captured different human actions such as walking around, standing/sitting, motion-

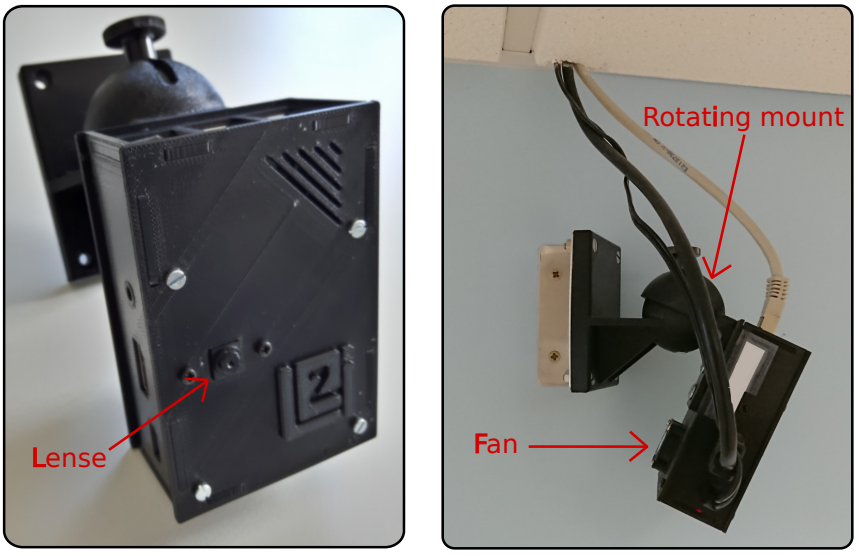


Figure 6.1 – Raspberry Pi 3 used as smart camera. The camera case was specially designed to include a fan and a rotating mount.

less, entering/leaving a space and group merging/splitting. In addition, one view only includes shadows of people moving around.

2. The information of the environment is crucial for smart building applications. For this reason, the dataset includes the IFC file of the I3M building (referred as I3M-IFC) and a camera-calibration file for each camera node.

3. The dataset also includes people detection manual and automatic annotations (PD-MAN and PD-AUT respectively), as well as people tracking manual annotations (PT-MAN). The use of automatic people annotation aims not only to propose an alternative to the time- consuming manual annotation but also to evaluate the complexity of each video (in terms of difficulty to detect people) using state-of-art people detectors.

Figure 5.1 presents the distribution of the SCs in the environment and some examples of their images. A demonstration of the deployed network can be found at http://wisenet.checksem.fr/#/demo. The SCs used were the Raspberry Pi 3 model B1 and its camera module v2.12, that contains a Sony IMX219 8-megapixels sensor.[1] the Raspberry Pi was chosen due to its high-performance and low-cost. The Raspberry Pi is a *multi-core*, general-purpose mini computer with the capability to attach a high definition camera and to execute multiple programs. Some of its specifications are: quad-core 64-bit ARM Cortex A53 processor at 1.2GHz, 1GB LPDDR2-900 SDRAM, fast Ethernet (100Mbit/s), 802.11n Wireless LAN, Bluetooth 4.1 (including Bluetooth Low Energy), 400MHz VideoCore IV multimedia and a 8-megapixels camera sensor (Sony IMX219). A case for the Raspberry Pi was specially designed to include a cooling system and a rotating mount that facilitates its installation (see Fig.6.1). The cooling system is important to enable the Raspberry Pi to record for long periods of time without overheating.

[1] https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

Table 6.2 – Description of WiseNET video sets. *#Videos* indicates the number of cameras used in the set; *#Frames (Time)* indicates the number of frames and the recording time of a complete set; *#People* indicates the number of people present in the set; *#PD-MAN* refers to the number of manually annotated people detection bounding boxes.

Set #	Resolution	FPS	#Videos	#Frames (Time)	#People	#PD-MAN
1	1280×720	30	5	8749 (01'00"×5)	5	13777
2	1280×720	30	5	17767 (02'00"×5)	2	8590
3	1280×720	30	5	17758 (02'00"×5)	2	9777
4	1280×720	30	5	35747 (04'00"×5)	3	17489
5	640×480	25	6	6000 (00'40"×6)	14	11495
6	640×480	25	6	6000 (00'40"×6)	6	6545
7	640×480	25	6	6000 (00'40"×6)	7	7109
8	640×480	25	6	6000 (00'40"×6)	6	4605
9	640×480	25	6	6000 (00'40"×6)	8	11520
10	640×480	25	6	6000 (00'40"×6)	9	10375
11	640×480	25	6	6000 (00'40"×6)	15	10631
			62	**122021 (1h13'00")**	**77**	**111913**

VIDEO SETS

Multiple video sets were recorded at different times. A description of each video set is presented on Table 6.2. In summary, there are 11 video sets, composed of 62 videos that cover more than 1 hour of video footage, 122K frames, 77 people tracks,[2] and around 112000 PD-MAN annotations. The video sets were captured at two resolutions—HD 720 (1280 × 720) or VGA (640 × 480)—different Frames Per Second (FPS)—30 or 25—various recording time—40 seconds or 1, 2 and 4 minutes—and using two video codecs—MPEG-4 and Planar 4:2:0 YUV. The different recording characteristics leads to a richer and more diversified dataset.

Notice that the videos recorded by the `smartCamera_6` do no recorded any person. However, they were left in the dataset because they include shadows of people moving, which might be useful for people interested in *shadow detection*.

CONTEXTUAL INFORMATION

The contextual information of the WiseNET network is composed of two parts, the information of the environment where the network was deployed and the information concerning the camera nodes.

The information about the environment is contained in the I3M-IFC file. The I3M-IFC file was obtained from the company in charge of the construction of the I3M. IFC is a data model standard, use to describe architectural, building and construction industry data. The IFC data model facilitates the interoperability between the different agents involved in a building construction. The I3M-IFC contains large amount of information concerning the I3M building, e.g., information about all the elements composing the building, their geometrical information, their position and their relation to other elements. Figure 6.2 shows

[2] Each people that appear in the video sets signed informed written consent before participating.

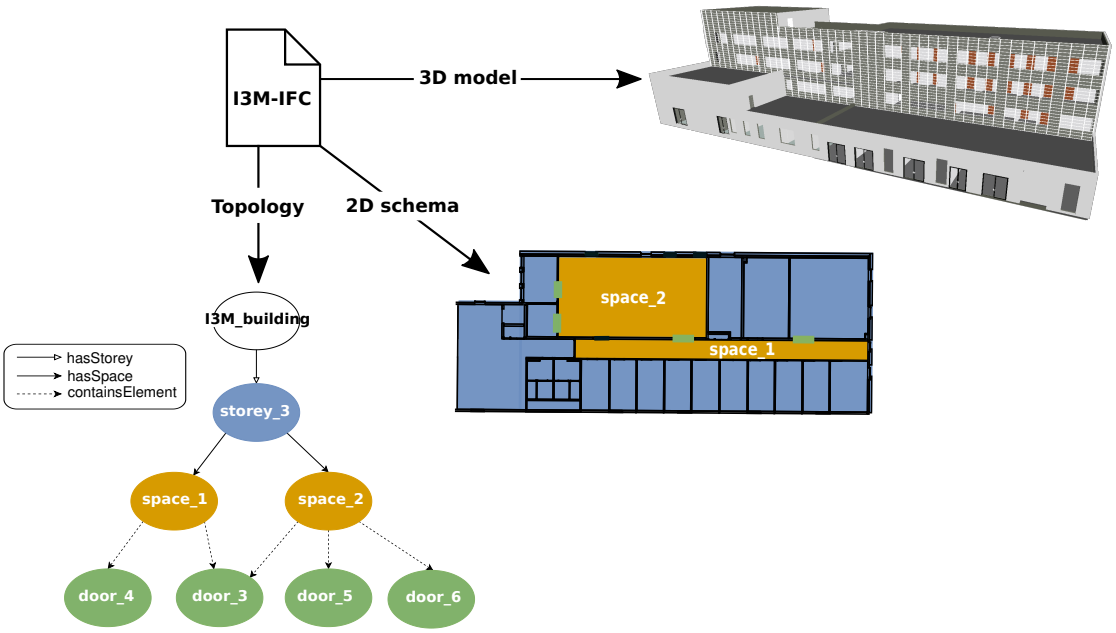


Figure 6.2 – Data generated from the IFC file of the I3M building (I3M-IFC). The 2D schema and the topology graph focus only on the third storey and on some spaces and doors of interest.

some examples of data that can be generated from the I3M-IFC file. For example, the environment's topology can be obtained from an IFC file by following the process presented in Chapter 5, while the 2D and 3D schema can be generated using Computer-Aided Design (CAD) software such as Solibri.[3]. Another example of data that can be obtained from an IFC file are, the dimensions of the spaces where the cameras were installed and the dimensions of the doors they observe, as presented in Table 6.3. However, to be able to extract detailed information—such as the dimensions of elements—some knowledge of the IFC standard is required.

The camera-calibration files contain the position of each camera node in the environment and the information about the Regions Of Interest (ROIs) they observe. The position of each camera node can be observed in Fig. 5.1, moreover Fig 6.3 shows the ROIs observed by each node and their assigned unique identifier (ID). Furthermore, Table 6.4 presents the doors *represented by* each ROI. Notice that a door can be represented by multiple ROIs, this was done to make each camera-calibration file independent of each other, and to facilitate the procedure.

The camera-calibration files, which were already presented in Section 5.2.2, take all the information presented in Fig.6.3 and Table 6.4 and synthesize it in JSON structured files. An example of a camera-calibration file can be found in Listing 5.3.

People detection annotations

The people detection manual (PD-MAN) annotations were obtained by manually enclosing a bounding box (Bbox) around each person that appears in a video frame, assigning

[3]https://www.solibri.com/bim-ifc

Table 6.3 – Dimensions of the spaces and doors of interest depicted on Fig. 5.1. These dimensions were extracted from the I3M-IFC file. The door's dimensions are defined as *width*×*height*, while the space's dimensions are defined as *length*×*width*×*height*.

Element	Dimension (m)
door_1-door_4	1.5×2.075
door_5-door_7	1×2.075
space_1	$27.7 \times 1.747 \times 3.52$
space_2	$14.33 \times 7.04 \times 3.52$
space_3	$7.5 \times 7.04 \times 3.52$
space_4	$2.77 \times 2.91 \times 3.52$
space_5	$3.16 \times 4.14 \times 3.52$
space_6	$2.48 \times 4.47 \times 3.52$

them an ID and stating if they are around a ROI. The process was performed by using a software developed in Python[4] using OpenCV library[5]—the code is provided in the dataset. The PD-MAN annotation rules were as follows:

1. On each video set, a unique ID should be associated to each person. For example, the person "Mario" should have the same ID in the five videos composing the set 1. Moreover, if "Mario" appears in another set, he might be assigned a different ID.
2. The Bbox should be created by selecting its top-left and bottom-right corners.
3. If only a person's limb is visible, then no Bbox should be drawn.
4. If a person is partially occluded, then the Bbox should enclose only the visible parts.
5. If a person torso is no visible, e.g., only its head is visible, then no Bbox should be drawn.
6. If a person is not visible for the human eye—because is totally occluded by an object, is outside the camera's FOV or the scene is too dark—then no Bbox should be drawn. Even if the person's position could be deduced from previous frames.
7. A person is considered around a ROI if: (1) the center of its bounding box is inside the ROI and (2) if the bounding box is at a similar level than the ROI, i.e., if the highest and lowest points of the Bbox and the ROI are around the same height.

Performing manual annotations is a time-consuming task, therefore, it was only performed on every fifth frame starting from frame 0. However, the information was propagated to the missing frames, e.g., the annotation in frame 0 was propagated into frames 1, 2, 3 and 4, in the same manner, the annotation of frame 5 was propagated to frames 6, 7, 8 and 9, and so on. Nevertheless, it took around 26 hours for 3 people to obtain the PD-MAN annotations.

The annotation's meta-data was stored in JSON files, structured following the logic that a *video has a set of frames and some of those frames present a set of Bboxes (detections)*. Listing 6.1 presents an extract of a resulting JSON file, where the fields correspond to:

[4] https://www.python.org/
[5] http://opencv.org/

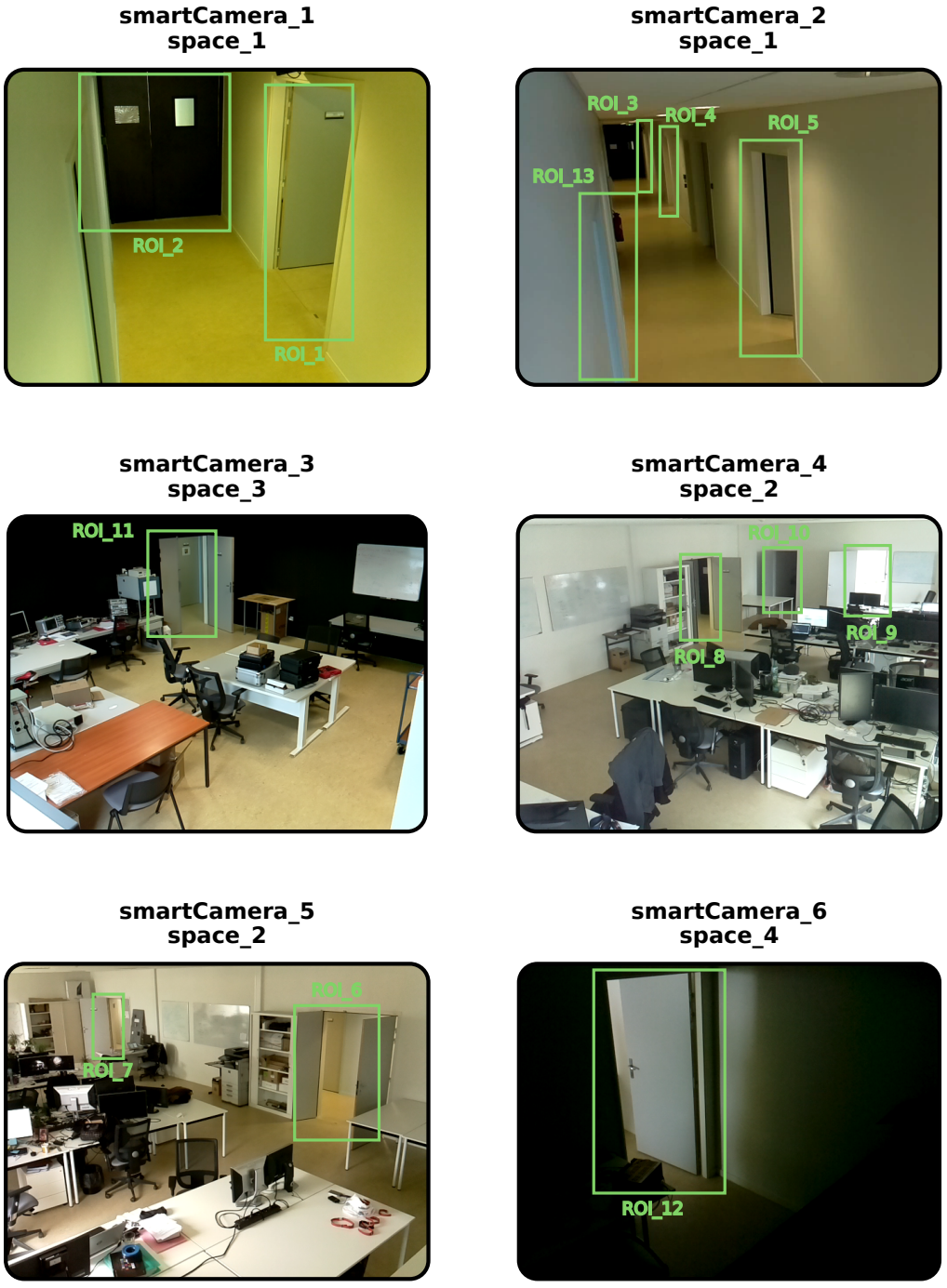


Figure 6.3 – Regions Of Interest (ROIs) observed by each camera node.

- *video:* name of the video file from which the meta-data was obtained. In the example, *video3_2.avi* means that the video comes from *set 3* and *camera node 2*.
- *resolution:* video resolution.
- *frames:* set of frames with Bboxes. The frames with no Bboxes are not considered.
  - *framNumber:* frame number.
  - *deviceID*: ID of the smart camera observing the scene.

Table 6.4 – Doors represented by Regions Of Interest (ROIs).

	door_1	door_2	door_3	door_4	door_5	door_6	door_7
ROI_1	✓	-	-	-	-	-	-
ROI_2	-	✓	-	-	-	-	-
ROI_3	-	✓	-	-	-	-	-
ROI_4	-	-	✓	-	-		-
ROI_5	-	-	-	✓	-	-	-
ROI_6	-	✓	-	-	-	-	-
ROI_7	-	-	✓	-	-		-
ROI_8	-	✓	-	-	-	-	-
ROI_9	-	-	-	-	-	✓	-
ROI_10	-	-	-	-	✓	-	-
ROI_11	-	-	-	✓	-	-	-
ROI_12	-	-	-	-	✓	-	-
ROI_13	-	-	-	-	-	-	✓

  - *inXSDDateTime*: synchronized timestamp, obtained from the smart camera.
  - *detections*: set of Bboxes (detections) made in the same frame.
    * *imageProcessing*: algorithm used to perform the detections. We consider the manual detections as a *ground truth* method.
    * *class*: type of object detected by the image processing algorithm.
    * *regionOfInterest*: ROI's ID. If the detection is not around a ROI then the value is *null*.
    * *xywh*: detection's top-left point (x,y), width (w) and height (h).
    * *visualDescriptors*: array of visual features describing a detection.
    * *id*: person's ID.

The PD-MAN annotations are sufficient to test and evaluate the WiseNET system, as it will be presented in Section 6.2. However, we decided to go a step forward and to test the system using real people detectors methods, as it will be presented in Section 6.3. The use of real detectors allow us to obtain automatically generated annotations (PD-AUT), and to evaluate the influence of the detection-quality in the WiseNET system. We decided to presented the methods to obtain the PD-AUT afterwards (in Section 6.3) because other explanations are required for comparing them, such as the different metrics used.

PEOPLE TRACKING ANNOTATIONS

People tracking manual annotations (PT-MAN) consists in manually stating the space location of each person at all times, during a complete video set. This is done by considering the people's ID and the time they enter and leave each space.

For each video set, the tracking meta-data is stored in a JSON file, where the fields corresponds to:

- *set:* video set number from which the PT-MAN was obtained.

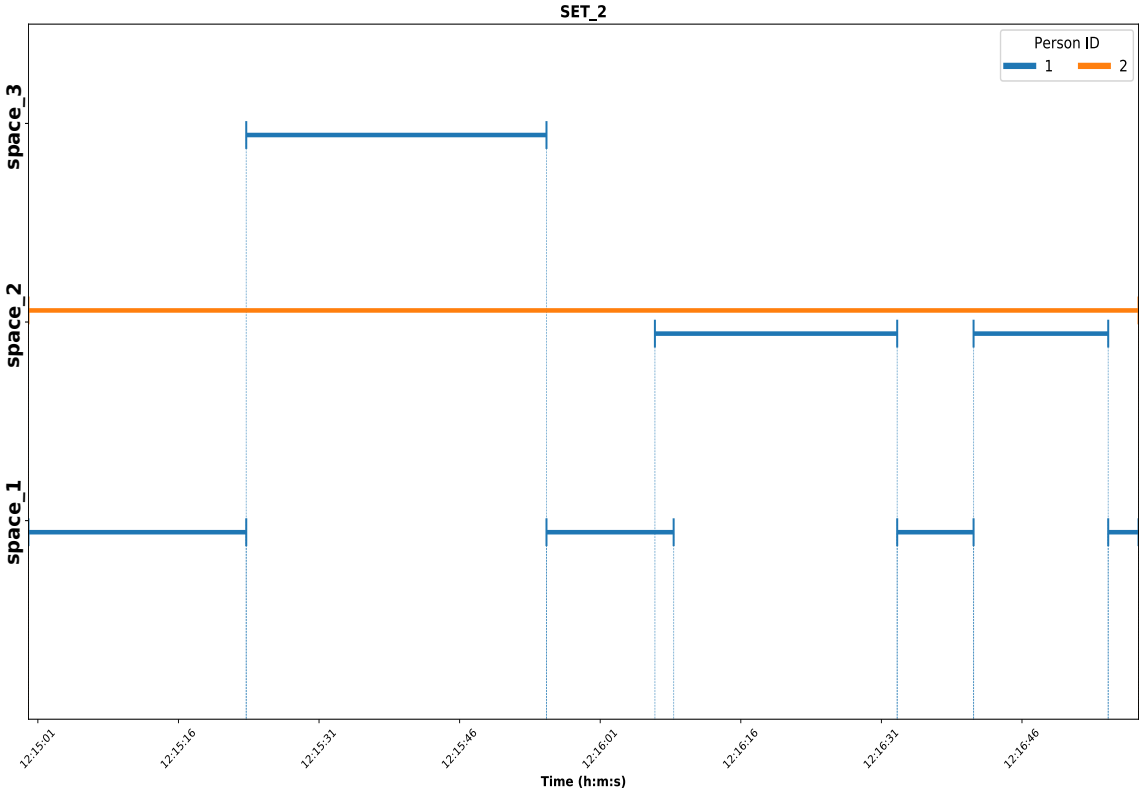


Figure 6.4 – Space-time graph representing the tracking ground truth of video set 2.

- *tracks:* set of people tracks. A track relates a person with a set of spaces at some periods of time. A track is divided into a set of tracklets.
  - *id:* person's ID.
  - *tracklets:* set of tracking segments of a person.
    - *location:* tracklet's space location.
    - *start:* tracklet's starting time.
    - *end:* tracklet's end time.

Furthermore, a space-time graph generated from the tracking meta-data is also provided in the dataset. The space-time graph is an intuitive way of presenting the location and the changes of spaces of all people during a period of time. For exemplification, Listing 6.2 presents the tracking data for video set 2, while Fig. 6.4 presents a graphical representation of the data. In the space-time graph, each blue line represents the different *tracklets* of persons #1, while the complete set of blue lines represent the *track* of person #1. From the space-time graph it can be easily observed that there were 2 people recorded on video set 2; that person #1 moved between spaces 1, 2 and 3, while person #2 stayed at space 2 during the whole recording time. Notice that sometimes the tracklets overlap in time, this occurs when a person is passing from one space to another thus it is detected around a door (e.g., around time 12:16:07).

The PT-MAN can be used for evaluating tracking algorithms, as it will be presented in Section 6.2.2.

VALUE OF THE DATASET

To summarize, the value of the WiseNET dataset can be recapitulated in 5 points [113]:

- Large, diverse and high-quality video sets recorded using an indoor multi-camera multi- space network (6 cameras deployed on 4 spaces, more than 1 hour of video footage, 11 video sets, 62 videos, 77 people).
- The dataset could be used as a benchmark for people detection, people re-identification and multi-space people tracking, thanks to the given automatic and manual annotations.
- The video sets could be used for human-action recognition such as walking around, standing/sitting, motionless, entering/leaving a space and group merging/splitting. Moreover, they could be also be used for office-objects detections such as tables, monitors, chairs, etc. Furthermore, one camera view only includes shadows of people moving around.
- Each frame was timestamped and annotated using a JSON format, making the meta- data easy to read, understand and re-use.
- Non-conventional information about the network's environment is provided, such as: (1) the position of the camera nodes, (2) the doors observed by them, (3) the dimensions of the doors and their position with respect the camera's field of view, (4) the dimensions of the spaces containing the camera nodes and (5) an IFC file containing the complete information of the environment.

The complete dataset is publicly available at https://data.4tu.nl/repository/uuid:c1fb5962-e939-4c51-bfd5-eac6f2935d44. It can be also downloaded from the project's website http://wisenet.checksem.fr/#/dataset.

Listing 6.1 – Example of person detection file of video3_2.avi. For simplicity only one frame is shown.

```
{
    "video": "video3_2.avi",
    "resolution": [
        {
            "width": 1280.0, "height": 720.0
        }
    ],
    "frames":[
        .
        .
        .,
        {
            "frameNumber": 1639,
            "deviceID": "smartCamera_2",
            "inXSDDateTime": "2017-05-20T12:18:45.922Z",
            "detections": [
                {
                   "imageProcessing": "groundtruth",
                   "class": "person",
                   "regionOfInterest": "regionOfInterest_5",
                   "xywh": [107, 20, 30, 50],
                   "visualDescriptors": [0.2, 0.4,..., 0.12],
                   "id": 1,

                },
                {
                   "imageProcessing": "groundtruth",
                   "class": "person",
                   "regionOfInterest": "null",
                   "xywh": [88, 22, 24, 52],
                   "visualDescriptors": [0.0, 0.1,..., 0.7],
                   "id": 2,
                }
            ]
        },
        .
        .
        .
    ],
}
```

Listing 6.2 – Tracking ground truth file of video set 2.


```
{
  "set": 2,
   "tracks":[
     {
       "id": 1,
       "tracklets": [
         {
           "location": "space_1",
           "start": "2017-04-24T12:15:00.001Z", "end": "2017-04-24T12:15:23.234Z"
         },
         {
           "location": "space_3",
           "start": "2017-04-24T12:15:23.234Z", "end": "2017-04-24T12:15:55.267Z"
         },
         {
           "location": "space_1",
           "start": "2017-04-24T12:15:55.267Z", "end": "2017-04-24T12:16:08.834Z"
         },
         {
           "location": "space_2",
           "start": "2017-04-24T12:16:06.834Z", "end": "2017-04-24T12:16:32.687Z"
         },
         {
           "location": "space_1",
           "start": "2017-04-24T12:16:32.687Z", "end": "2017-04-24T12:16:40.834Z"
         },
         {
           "location": "space_2",
           "start": "2017-04-24T12:16:40.834Z", "end": "2017-04-24T12:16:55.201Z"
         },
         {
           "location": "space_1",
           "start": "2017-04-24T12:16:55.201Z", "end": "2017-04-24T12:16:58.434Z"
         }
       ]
     },
     {
       "id": 2,
       "tracklets": [
         {
           "location": "space_2",
           "start": "2017-04-24T12:15:00.001Z", "end": "2017-04-24T12:16:58.434Z"
         }
       ]
     }
  ]
}
```

6.2/ System evaluation

Once the WiseNET ontology has been defined (Chapter 4); the framework for inserting data into the ontology has been explained (Chapter 5); and a dataset made of the environment and SCN information has been acquired (Section 6.1); then, we can proceed to the evaluation of the system. The evaluation consists in verifying if the system satisfy its intent. We will perform two types of evaluations. The first one, is a qualitative evaluation and focus on the WiseNET ontology which is the core element of the system (presented in Section 6.2.1). The second one, is a quantitative evaluation and focus on applying the system to solve a computer vision problem, specifically *people tracking* problem (presented in Section 6.2.2).

6.2.1/ Ontology evaluation

The WiseNET ontology plays a central role in the WiseNET system, it is in charge of combining the information coming from the SCN with the contextual information. Moreover, all the system's results are stored in it. According to Hitzler et al. [71], the accuracy criteria is a central requirement for ontology evaluation. This criteria consists in verifying if the ontology accurately captures the aspects of the modeled domain for which it has been designed for. The development of the WiseNET ontology was based on 41 Competency Questions (CQs) that the ontology should be able to answered (see Table 4.1). Therefore, the ontology evaluation consists in showing that those questions can be answered by the ontology. Moreover, the CQs were divided into 2 sub-sets: *static CQs* which answers do not change over time (Section 6.2.1.1), and *dynamic CQs* which answer might change over time (Section 6.2.1.2).

The evaluations were performed after inserting the complete contextual information, i.e., the information of the I3M environment and the general information of the camera nodes. This was done by using the static population process presented in Section 5.2).

Concerning the dynamic information, the video set 3 was used for evaluating the CQs. This video set was chosen because is suitable for exemplification and it has a adequate level of complexity. In this video set, there are two people moving around multiple spaces, while being observed by multiple cameras. Figure 6.5 presents a graphical representation of a part of video set 3. For conciseness, only seven timestamps, taken approximately each 10 seconds, will be consider during the evaluation: $t_1 \approx$ 12:15:44, $t_2 \approx$ 12:15:55, $t_3 \approx$ 12:16:07, $t_4 \approx$ 12:16:13, $t_5 \approx$ 12:16:28, $t_6 \approx$ 12:16:39 and $t_7 \approx$ 12:16:50. Furthermore, Fig 6.6 presents the manually generated (i.e., ground truth) space-time graph of video set 3. From this graph, it can be observed the position of each person at each time, during the complete video set. Notice that the person with ID 2 is not visible by any camera for around 10 seconds, from 12:16:10 to 12:16:19 (as shown in the graph).

The people detection manual annotations from video set 3, were automatically inserted by using the dynamic population process presented in Section 5.3. Moreover, the dynamic population was stopped at the different timestamps, to execute some CQs. This was performed to show the evolution/changes of the answers in time as it will be presented in Section 6.2.1.2.

Furthermore, the query execution times presented below, were obtained by executing the queries 10 consecutive times and then performing an average. The evaluation procedure

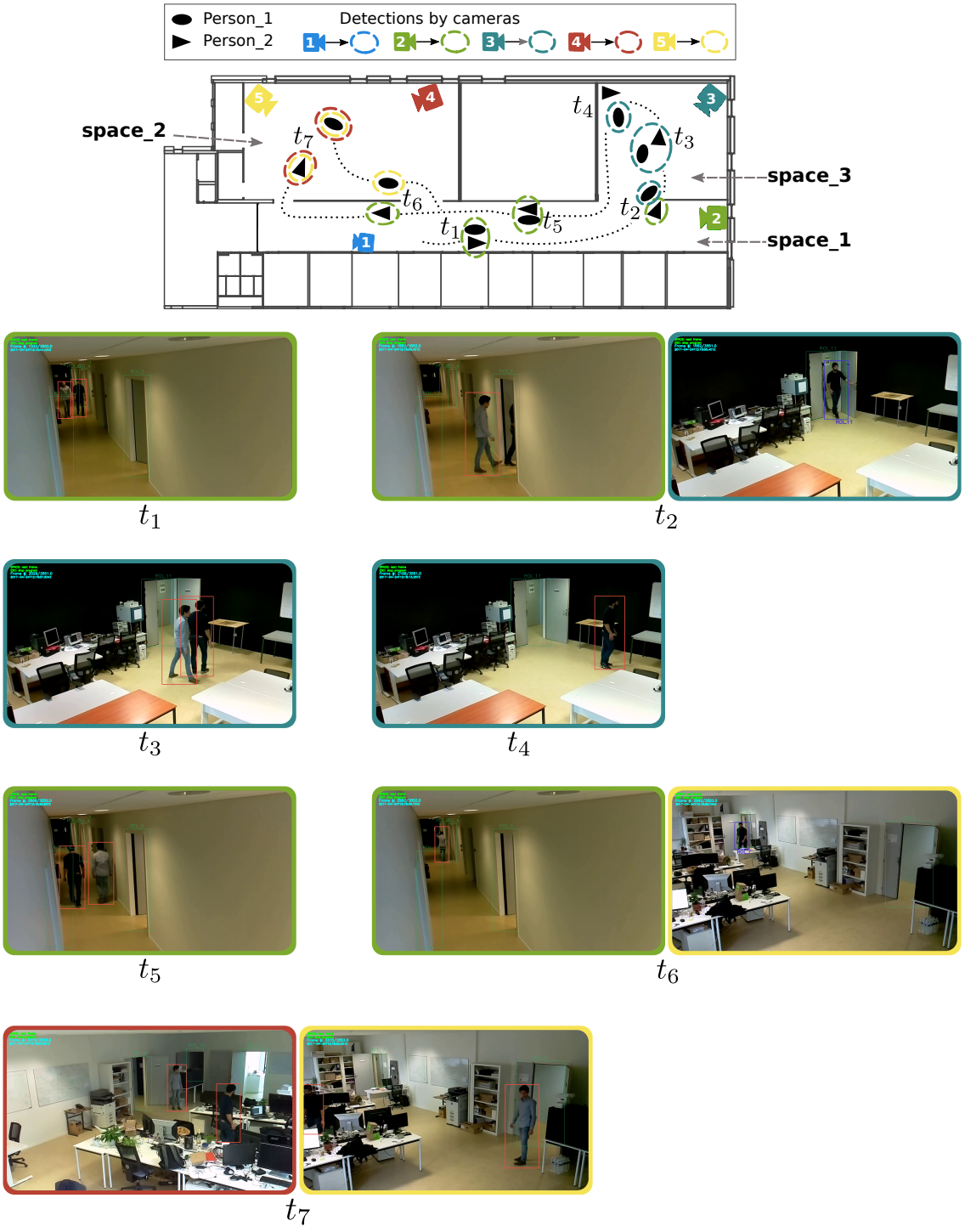


Figure 6.5 – Extract of video set 3, where two people are walking around three spaces, while being detected by a SCN. Seven timestamps of interest are being shown. The images information shown in the top-left corner is not relevant.

was performed in a machine with the following configuration: Intel Core i7-4790 CPU @3.6GHz $\times$ 4, 16GB of RAM and a "Java Heap" size set to 200MB.

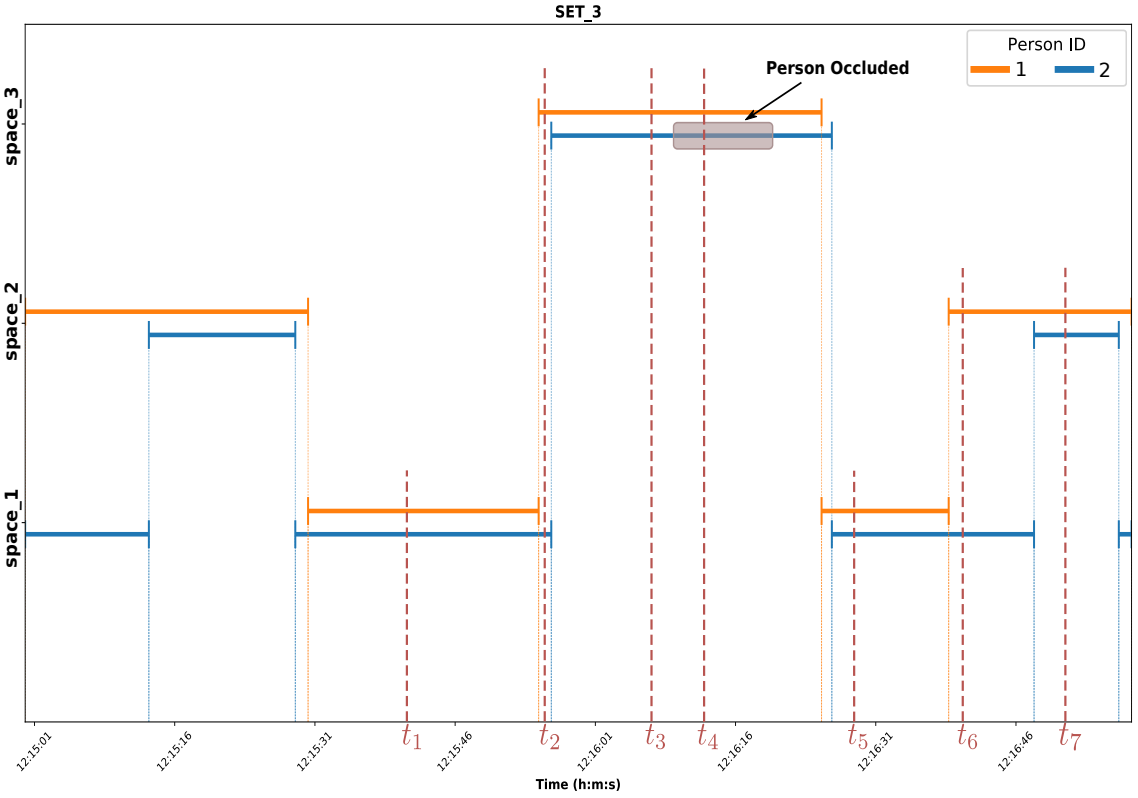


Figure 6.6 – Space-time graph representing the tracking ground truth of video set 3 and the position of the timestamps of interest.

6.2.1.1/ STATIC CQS

The static CQs are those which answers do not change over time. This type of CQs concern mostly the information about the structure of the building, its topology and the different elements contained in the spaces. Furthermore, the static CQs are independent of the data coming from the SCN. Thus, if we perform the static CQs at the beginning, during or at the end of any video set, the answer will be the same. Therefore, we decided to evaluate the static CQs after populating the complete data of video set 3.

In Table 4.1, the CQs 1 to 19 are static. Due to space constrains, only some relevant CQs will be presented:

- **CQ2: how many doors are contained in each storey?** Listing 6.3 presents the query used for answering the question, where Line 3 gets all the storeys, Lines 4-5 get all the doors contained in the storeys, Line 6 groups each door by storey and the command COUNT in Line 1 calculates the total number of door instances by storey. The query uses the storey-element relation (`hasElement`) which is not directly inserted in the ontology, but can be deduced by the subsumption hierarchies $\texttt{containsElement} \sqsubseteq \texttt{hasElement}$ and $\texttt{hasSpace} \sqsubseteq \texttt{containsZone}$ and the property chain $\texttt{containsZone} \circ \texttt{hasElement} \sqsubseteq \texttt{hasElement}$ (Eq.4.19). Because deduction is needed, then the reasoner is required to obtain the results. The query execution time is around 130 ms. Based on the real I3M data, the query returns the correct results:

Listing 6.3 – SPARQL query for getting the number of doors contained in each storey (CQ2).

```
1 SELECT  ?storeys (COUNT (?doors) AS ?numberOfDoors)
2 WHERE {
3    ?storeys rdf:type bot:Storey;
4             bot:hasElement ?doors.
5    ?doors rdf:type wisenet:Door.
6 } GROUP BY ?storeys
```

Listing 6.4 – SPARQL query for getting the number of spaces contained in each storey (CQ3).

```
1 SELECT  ?storeys  (COUNT (?spaces) AS ?numberOfSpaces)
2 WHERE {
3    ?storeys rdf:type bot:Storey;
4             bot:hasSpace ?spaces.
5 } GROUP BY ?storeys
```

?storeys	?numberOfDoors
inst:storey_3	40

.

- **CQ3: how many spaces are contained in each storey?** Listing 6.4 presents the query for answering the question, where Line 3 and 4 get all the storeys and the spaces contained by them, Line 5 group them by storeys and Line 1 count the number of spaces. This query does not need the reasoner to obtain the results, and its execution time is around 10 ms. The query returns the correct results:

?storeys	?numberOfSpaces
inst:storey_3	33

.

  A similar query could be used to obtain all the spaces contained in the storeys, by removing the COUNT-AS and GROUP BY commands.

- **CQ8: what are all the spaces connected to a space?** Listing 6.5 presents the query for getting all the spaces connected to `space_2`. This query deduces that two spaces are adjacent/connected due to the rule stating that: *two spaces are adjacent if they contain the same door* (Rule 1 presented in Listing 4.1). Line 4 filters out the space itself to only consider the other spaces. This query requires the reasoner and its execution time is around 20 ms. The query returns the correct results:

?connectedSpaces
inst:space_1
inst:space_4
inst:space_5

  A similar query could be used to count the number of connecting spaces of each space by replacing: Line 1 by "`SELECT ?spaces (COUNT(?connectedSpaces) AS ?numConnectedSpaces)`" and the entry "`inst:space_2`" by the general variable "`?spaces`".

Listing 6.5 – SPARQL query for getting the spaces connected to `inst:space_2` (CQ8).

```
1 SELECT  ?connectedSpaces
2 WHERE {
3    inst:space_2 bot:adjacentZone ?connectedSpaces
4    FILTER (?connectedSpaces != inst:space_2)
5 }
```

Listing 6.6 – SPARQL query for getting the type of sensor deployed and their locations (CQ12-13).

```
1 SELECT  ?sensor ?type ?location
2 WHERE {
3    ?sensor rdf:type sosa:Sensor;
4            rdf:type ?type;
5            sosa:isHostedBy ?location.
6    FILTER (?type!=owl:Thing && ?type!=bot:Element && ?type!=sosa:Sensor)
7 }
```

- **CQ12-13 which type of sensors are deployed in the system and where are they?** Listing 6.6 answer the question by getting all the sensor instances (Line 3), their types (Line 4) and their locations (Line 5). Moreover, the general classes of sensors are filter out (Line 6), leaving only the detailed types, such as smart camera, thermostat, humidity sensor, etc. However, in our system there are only smart cameras as shown in the results. This query requires the reasoner for deducing the association to the upper-class `Sensor` and to deduce the location (`isHostedBy`). The query execution time is around 400 ms, and it returns the following results:

?sensor	?type	?location
wni:smartCamera_1	wisenet:SmartCamera	inst:space_1
wni:smartCamera_2	wisenet:SmartCamera	inst:space_1
wni:smartCamera_3	wisenet:SmartCamera	inst:space_3
wni:smartCamera_4	wisenet:SmartCamera	inst:space_2
wni:smartCamera_5	wisenet:SmartCamera	inst:space_2
wni:smartCamera_6	wisenet:SmartCamera	inst:space_4

.

- **CQ14: which and where are the nearest sensors to a sensor?** Listing 6.7 presents the query to get the nearest sensor to `smartCamera_1`, where Line 3 gets all its nearby sensors, Line 4 gets their location and Line 5 filter out the same sensor. This query deduces the relation between sensors (`hasNearBySensor`) due to the rules stating that: *two sensors are nearby if they are located in the same space or if they are located in adjacent spaces* (Rules 2 and 3, presented in Listings 4.2 and 4.3 respectively). This query requires the reasoner for the deduction of the sensor relation, and its execution time is around 100 ms. The results of the query are the nearest sensors to `smartCamera_1`:

Listing 6.7 – SPARQL query for getting the nearest sensors to the `wni:smartCamera_1` and their location (CQ14).

```
SELECT  ?nearestSensor ?location
WHERE {
   wni:smartCamera_1 wisenet:hasNearbySensor ?nearestSensor.
   ?nearestSensor sosa:isHostedBy ?location.
   FILTER (?nearestSensor != wni:smartCamera_1)
}
```

Listing 6.8 – SPARQL query for getting the elements observed by the `wni:smartCamera_1` (CQ16).

```
SELECT ?elements
WHERE {
   wni:smartCamera_1 sosa:observes ?elements.
   ?elements rdf:type bot:Element.
}
```

?nearestSensor	?location
wni:smartCamera_2	inst:space_1
wni:smartCamera_3	inst:space_3
wni:smartCamera_4	inst:space_2
wni:smartCamera_5	inst:space_2

.

Notice that if we are interested in a specific type of sensors, this can be easily done by adding a line restricting the type of the `?nearestSensor` variable. This can be useful, for example, when a smart camera needs to broadcast information only to nearby cameras.

- **CQ16: which building elements does a camera observes?** Listing 6.8 present the query to get the elements observed by the `smartCamera_1`. This query looks simple but it needs a concatenation of deductions. Firstly, the deduction that a camera's Field Of View (FOV) `shows` and element represented by a ROI, defined by the property chain $\texttt{shows} \circ \texttt{represents} \sqsubseteq \texttt{shows}$ (Eq. 4.27). Afterwards, the deduction that a camera `observes` the elements `shown` by it's FOV, defined by the property chain $\texttt{hasFieldOfView} \circ \texttt{shows} \sqsubseteq \texttt{observes}$ (Eq. 4.26). This query requires the reasoner and its execution time is around 1100 ms. The results of the query are:

?elements
inst:door_1
inst:door_2

.

The static CQs concerning only the building information—e.g., CQs 2,3,8—are answered by using the data extracted directly from the IFC file of the building (see Section 5.2 for the extraction procedure). Similarly, the static CQs concerning the sensors and their relation with the environment—e.g., CQs 12,13,14,16—are answered by using the sensor static information (i.e., sensor's calibration, see Section 5.2.2)

As presented in this section, **the system is able to correctly answer all the CQs concerning the built environment and the sensors deployed in it.** Furthermore, **the proposed formalism for defining the CQs allows to question, in real-time, the different elements composing a smart building.**

Moreover, the proposed CQs have many applications. Specially, for building managers which would like to quickly know: the building's topology, how many elements are located in a building, their position, their dimensions, etc. Or a path guidance application, which can use the connection between spaces to compute the closest path between two spaces. This type of application could be use in, for example, hospitals or even for disaster-evacuation guidance. Moreover, the proposed CQs have many applications. Specially, for building managers which would like to quickly know: the building's topology, how many elements are located in a building, their position, their dimensions, etc. Or a path guidance application, which can use the connection between spaces to compute the closest path between two spaces. This type of application could be use in, for example, hospitals or even for disaster-evacuation guidance.

6.2.1.2/ DYNAMIC CQS

In contrast to the static CQs, the answer of the dynamic CQs might change over time, according to the data sent by the SCN.

In Table 4.1, the CQs 20 to 41 are dynamic, and as it can be observed, they concern mostly the building usage information, i.e., what is happening (or happened) in the building at a precise time.

Therefore, the evaluation of the dynamic CQs will be performed at different timestamps, as shown in Fig. 6.6. Furthermore, the space-time graph presented in the figure summarizes the building usage during the whole video set, thus it will be used as ground truth, i.e., the results of the queries below should be compared to the information in the graph.

Let us start by answering two important questions: **CQ27: how many people are in a space?** and **CQ28: is a space empty/occupied?** Listing 6.9 presents the query for getting the number of people in each space. The query first gets all the `PersonInSpace` instances which are open (Lines 3-5) and which are not noise (Line 6), then it group them by each space (Line 7) and finally count them (Line 1). This query does not the reasoner and its execution takes around 20 ms. Listing 6.10 presents the query for getting the spaces which are occupied. The query uses the boolean property `isOccupied`, which is deduced by the rule stating that: *if a zone/space contains a person then that zone/space is occupied* (Rule 6 presented in Listing 4.6). This query requires the reasoner and its execution time is around 143 ms. Furthermore, the query in Listing 6.9 will be referred as the *heatmap-query*, because it gives the occupancy map of people in the spaces.

Table 6.5 presents the results after performing both queries (one after the other) at each timestamps. All the results are correct. For example, at timestamp t_2, the query in Listing 6.9 returns that there is 1 person in `space_1` and 1 person in `space_3`, similarly, the query in Listing 6.10 returns that `space_1` and `space_3` are occupied. The spaces which are not returned as results are consider empty, thus with 0 people in them. Notice that, at timestamp t_4 there is one person located in `space_3` which is not visible by any camera, however the system is able to deduce that the person is still in the space, due to its last detection was not made around a door, i.e., the person has not left the space thus

Listing 6.9 – SPARQL query for getting the number of people in each space (CQ27). This query is referred as *heatmap query*.

```
1 SELECT ?spaces (COUNT (?x) AS ?numberOfPeople)
2 WHERE{
3    ?x rdf:type wisenet:PersonInSpace;
4       event:place ?spaces;
5       wisenet:isEventOpen "true"^^xsd:boolean;
6       wisenet:isNoise "false"^^xsd:boolean.
7 } GROUP BY ?spaces
```

Listing 6.10 – SPARQL query for getting the spaces which are occupied (CQ28).

```
1 SELECT ?spaces
2 WHERE {
3    ?spaces wisenet:isOccupied "true"^^xsd:boolean.
4 }
```

its `PersonInSpace` event is open.

The following questions **CQ34: what is the most visited space in the building?** and **CQ37: how many people entered a space in a period of time?**, can be answered by slightly modifying the heatmap-query, as shown in the *accumulated heatmap-query* presented in Listing 6.11. In contrast to the heatmap-query which counts the number of people *that are* in each space at a precise time, the accumulated heatmap-query counts the number of people *that have been* in each space *until* a precise time. Where the word *until* refers to a period of time, that is why is consider as accumulated. This is done by considering both open and closed `PersonInSpace` instances. This query does not required the reasoner and its execution time is around 18 ms. Table 6.6 shows the results after performing the accumulated heatmap-query at different timestamps. All the results are correct. Notice that before t_1, 1 person has already been in `space_1` and 2 people have been in `space_2` (see Fig. 6.6). Thus, when executing the query at t_1, 2 more people are in `space_1` which makes a total of 3 people that have been in the `space_1`. Similarly, at t_5, the 2 people re-enter the `space_1` thus resulting in 5 people that have been in `space_1`; and until the same time, 2 people have been in `space_2` and 2 in `space_3`. Finally, if we re-consider the questions and by looking the results, we can state that at each timestamp the `space_1` is the most visited space, and until t_7, 5 people have been there. Furthermore, the most visited space could also be directly obtained from the query by adding the line "`ORDER BY DESC(?numberOfPeople) LIMIT 1`", which will order the result in a descending order according to the number of people and it will return only the first result (i.e., the one with more people), the line should be added after Line 6.

Listing 6.12 presents the *space-time query* used to obtain the tracks for each person, i.e., the time they enter and left each space. The query gets all the `PersonInSpace` instances which are not noise (Lines 3-4), and then gets all the information related to the event, such as the people involved in them, their location, the time the event started and the time it ended (Lines 5-10). Finally, the results are order according to their starting time (Line 11). This query requires the reasoner to deduce that multiple `Person` instances are the same if they are related by the `owl:sameAs` property, consequently for each person we will get a single ID instead of multiple IDs. The execution time of the query is around

Table 6.5 – Heatmap results - Number of people in the spaces at different timestamps. The cells in gray represent the spaces which are occupied at each timestamp.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7
space_1	2	1	0	0	2	1	0
space_2	0	0	0	0	0	1	2
space_3	0	1	2	2	0	0	0
space_4	0	0	0	0	0	0	0
space_5	0	0	0	0	0	0	0
space_6	0	0	0	0	0	0	0

Table 6.6 – Accumulated heatmap results - Number of people that are and have been in the spaces at different timestamps.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7
space_1	3	3	3	3	5	5	5
space_2	2	2	2	2	2	3	4
space_3	0	1	2	2	2	2	2
space_4	0	0	0	0	0	0	0
space_5	0	0	0	0	0	0	0
space_6	0	0	0	0	0	0	0

220 ms.

The resulting table of the space-time query can be represented in a graphical way as shown in Figure 6.7. Where each horizontal line is a *tracklet*, i.e., a `PersonInSpace` instance which relates a `Person` with a `Space` during a period of time; and the set of tracklets from the same person (e.g., all the blue lines) is a *track*. Notice that if a `PersonInSpace` is open, then its end is plotted with a circle (see the last two tracklets in Fig. 6.7). Moreover, the people IDs are high—18 and 24—because multiple people have been created. Some are considered as "noise", thus, they are ignored, and others have been related and the reasoner just considers the latest ID of `Person` instance.

The space-time graph in Fig. 6.7 allows to answer many CQs, such as:

- **CQ23: where is a person located?** For example, at t_6 it can be observed that `Person_24` (which corresponds to *Person_2* in Fig. 6.5) is located at `space_1`, while at t_7 is located at `space_2`.
- **CQ24: where was a person in the last few minutes?** For example, at t_6 it can be stated that `Person_24` has been in `space_1` and `space_3`, for the last minute.
- **CQ25: For how long a person has been in a space?** For example, it can be stated that `Person_18` was in `space_3` around 34 seconds, from 12:15:54 to 12:16:28.
- **CQ26: where were all the people at a specific time?** For example, at t_1 it can be observed that both people are located at `space_1`, while at t_4, both people are locate at `space_3`.
- **CQ31: at what time does a person entered/left a space?** For example, it can be observed that `Person_24` entered the `space_3` at 12:15:56 and left it at 12:16:27. Notice that the person is considered in the space even thought is occluded for around 10 seconds.

Listing 6.11 – SPARQL query for getting the number of people that enter each space. This query is referred as *accumulated heatmap-query*.

```
SELECT ?spaces (COUNT (?x) AS ?numberOfPeople)
WHERE{
   ?x rdf:type wisenet:PersonInSpace;
      event:place ?spaces;
      wisenet:isNoise "false"^^xsd:boolean.
} GROUP BY ?spaces
```

Listing 6.12 – SPARQL query for getting time each person enter and left the spaces. This query is referred as *space-time query*.

```
SELECT ?person ?place ?start ?end ?isopen
WHERE {
   ?events rdf:type wisenet:PersonInSpace;
           wisenet:isNoise "false"^^xsd:boolean;
           event:agent ?person;
           event:place ?place;
           wisenet:isEventOpen ?isopen;
           event:time ?interval.
   ?interval time:hasBeginning/time:inXSDDateTimeStamp ?start.
   ?interval time:hasEnd/time:inXSDDateTimeStamp ?end.
} ORDER BY ?start
```

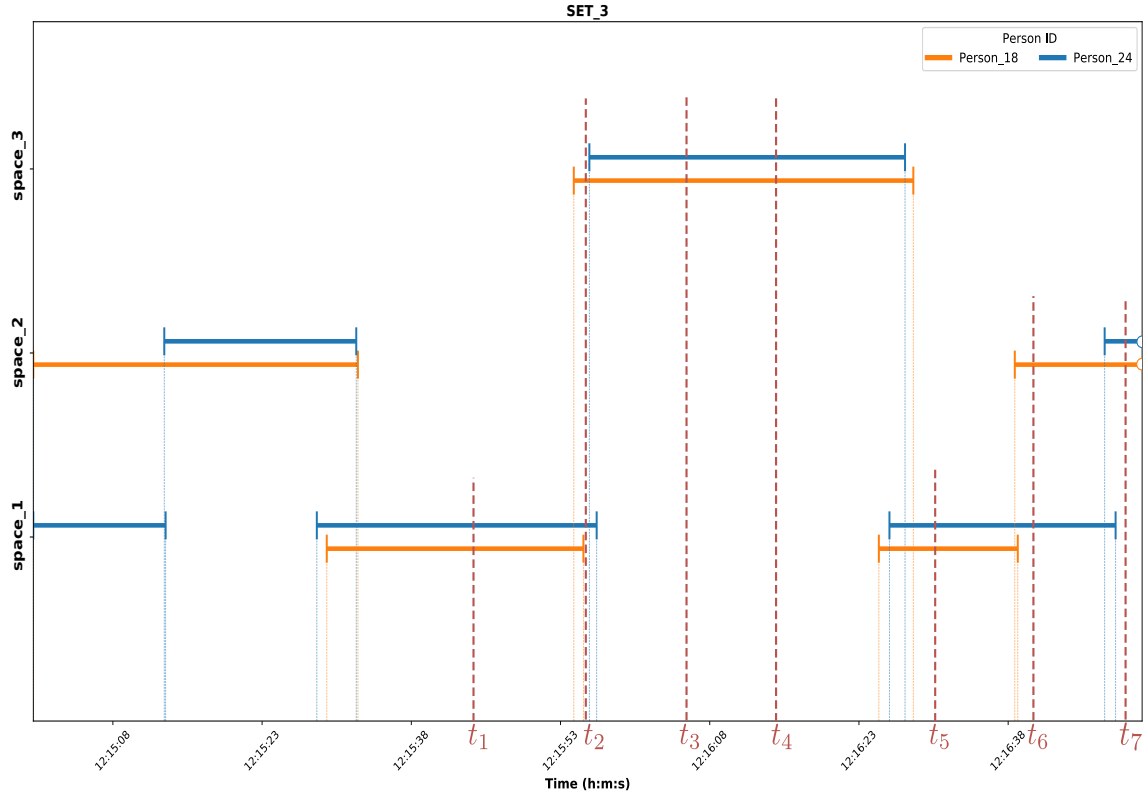


Figure 6.7 – Space-time graph obtained by executing the space-time query (Listing 6.12) few seconds after t_7.

Listing 6.13 – SPARQL query for getting the number of times a person has passed/used each door.

```
1  SELECT ?door (COUNT (?detections) AS ?usage)
2  WHERE{
3     ?events  rdf:type wisenet:PersonInSpace;
4              wisenet:isNoise "false"^^xsd:boolean;
5              event:sub_event ?detections;
6     event:time/time:hasBeginning ?timeInstant.
7     ?detections event:time ?timeInstant;
8                 wisenet:inRegionOfInterest/wisenet:represents ?door.
9     ?door rdf:type wisenet:Door.
10 } GROUP BY ?door
```

- **CQ33: where does a person stayed the longest time?** For example, consider the `Person_18`, if the question is asked at timestamps t_1, t_2, t_3 or t_4 the answered will be `space_2`, however if it is asked at timestamps t_5, t_6 or t_7 the answered will be `space_3`.
- **CQ36: at what time there are more people in a space?** For example, it can be stated that there are more people in `space_3` from 12:15:56 to 12:16:27.

The resulting space-time graph presented in Fig. 6.7, is similar to the one presented in Fig. 6.6, however the latter one was obtained manually, without any detections just by looking the videos (i.e., it is a ground truth), while the former one was obtained after automatically passing each camera's detection through the WiseNET system and then executing the space-time query. Furthermore, some differences can be observed in the degree of overlap between tracklets. A quantitative comparison between the ground truth and the resulting space-time graphs will be presented in Section 6.2.2.

Finally, the question **CQ35: which is the most used door in the building?**, can be answered by executing the query presented in Listing 6.13. When a person cross a door from one space to another, many detections are performed but (normally) only one `PersonInSpace` event is created. Therefore, the door usage is determined by counting the number of `PersonInSpace` events that stareted around a door. This is done by getting all the `PersonInSpace` instances which are not noise (Lines 3-4), then getting their first detections (Lines 5-7) and checking if they were made around a door (Lines 8-9). Finally, the number of detections are counted (Line 1) and grouped by doors (Line 10). This query requires the reasoner and it takes around 220 ms to be executed. The results of executing the query at the different timestamps are presented in Table 6.7. From the table it can be observed that, from t_1 to t_2 the `door_2` was the most used, from t_3 to t_4 it was the `door_2` and `door_3`, while from t_5 to t_7 the `door_4` was the most used. Notice that is possible to automatically obtain *only* the door most used at each timestamp by adding the following line "`ORDER BY DESC(?usage) LIMIT 1`", at the end of the query.

As presented in this section, **the system is able to correctly answer all the CQs concerning the building usage.** Furthermore, **the proposed formalism for defining the dynamic CQs allow to interaction with an smart building, and to monitor what is happening in real-time**. Moreover, the semantic-based system enables the smart building to understand what is happening inside itself *without* any human interaction. One important example, is the deduction that somebody is still present in a space even if no-

Table 6.7 – Door usage at different timestamps. The cells in gray show the most used door at each timestamp.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7
door_1	0	0	0	0	0	0	0
door_2	2	2	2	2	2	2	3
door_3	1	1	1	1	1	2	2
door_4	0	1	2	2	4	4	4
door_5	0	0	0	0	0	0	0
door_6	0	0	0	0	0	0	0
door_7	0	0	0	0	0	0	0

body is detected by any camera (person occluded). This type of deduction shows how our **semantic-based system overcomes a big limitation of computer vision systems —they only know what they observe—by using contextual information and logic rules.**

The CQs presented in this section can be used for many smart building applications. For example, it can be used for *maintenance*, where is important to know the usage of the different building elements (e.g., spaces, doors), to plan their cleaning or changing. They can also be used for *smart light control*, where the building will automatically turn on/off light according to the person's presence in spaces. However, the application we are more interested in is the *Intelligent Video Surveillance (IVS)*, which consist in helping the building managers to know (monitoring) what is happening in the environment. For example, by using the heatmap, accumulated heatmap and spac-time query, the system is able to count how many people are/have been in the building, where are the people now, where they have been, etc.

The following section will present the *monitor unit*, which is an IVS application that ease the monitoring task by graphically showing the results of some CQs.

6.2.1.3/ MONITOR UNIT

The monitor unit is in charge of presenting the status of the WiseNET system in an intuitive form, thus helping the monitoring and management of a built environment. The monitor unit is an user interface, designed thinking on the building managers which (most probably) may not be fluent with information technology, thus, they might not know how to create and execute SPARQL queries.

The main task of the monitor unit is to automatically execute each 2 seconds the heatmap-query (Listing 6.9), the accumulated heatmap-query (Listing 6.11) and the space-time query (Listing 6.12) in order to present in real-time what is happening in real-time. Moreover, it presents the query results in a graphical manner as shown in Fig 6.8. The examples shown in the figure were obtained at around t_7.

The three views—heatmap, accumulated heatmap and space-time graph—combine the contextual information of the environment with the environment usage and present the results in graphical and intuitive way. Moreover, these views can be very useful for people in charge of monitoring a built environment. For example, if somebody wants to know the position of the people in the building in real-time or where a person was some time before,

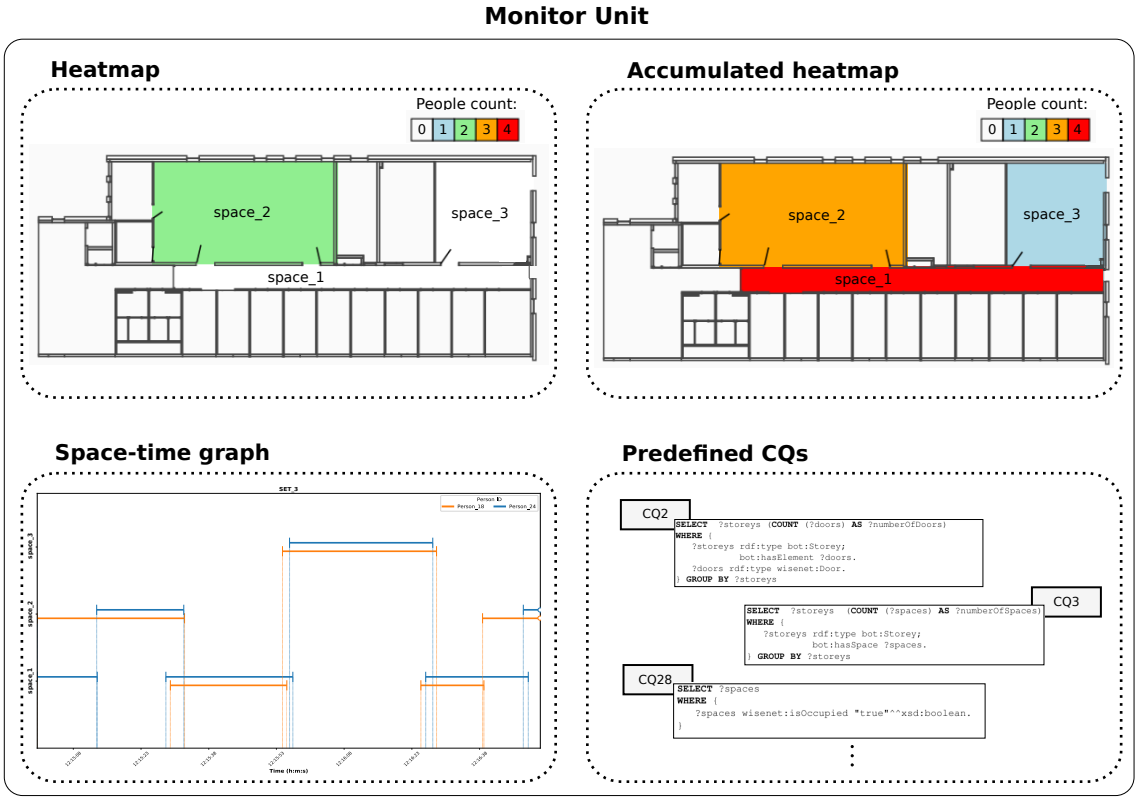


Figure 6.8 – Components of the monitor unit. The heatmap shows the number of people in each space at a given time. The accumulated heatmap shows the number of people that have been in each space. The space-time graph shows the location of people with respect to time. The Competency Questions (CQs) are a set of predefined SPARQL queries that can be executed at any time.

they just need to observe the heatmap or the space-time graph, instead of looking all the camera screens. Another example, if somebody would like to schedule cleaning task, they could look the accumulated heatmap to avoid cleaning spaces which have not been visited. Furthermore, these views protect the privacy of the building users, due to the people monitoring the environment do not need to observe the images from the cameras.

The monitor unit also collects a set of predefined SPARQL queries, that allow to answer some CQs presented previously. The queries are present in a graphical interface where the user just needs to press a button to execute them. In this way, users can easily execute queries even if they do not have any knowledge about SPARQL.

As a summary, the monitor unit is an interface of the WiseNET system. The main objective of the monitor unit ease the human-machine interaction. Specifically, **the Artificial Intelligence (AI) behind the monitor unit, creates semantic links between the different data and extracts only the pertinent information. Then, the extracted information is presented in an intuitive way to the users, in order to reduce their cognitive load.** By enabling this, **we can conclude that the WiseNET system is an IVS system.**

Moreover, the monitor unit presents many advantages, such as ease the building monitoring, visualizing in real-time what is happening in the environment, and protecting the private life of building users by not showing any image.

An interface showing the heatmap, the accumulated heatmap and the space-time graph can be found at http://wisenet.checksem.fr/#/monitor. While an interface showing the predefined SPARQL queries can be found at http://wisenet.checksem.fr/#/query.

6.2.2/ TRACKING WITH SEMANTICS - EVALUATION

In this section, we will measure the performance of our semantic-based system in tracking people. As stated in Chapter 2, people tracking is an important task in computer vision, especially for surveillance applications. People tracking consists of automatically assigning an Unique Identifier (ID) to multiple-people and maintaining it through time while they move through multiple-space and appeared in multiple-cameras. This specific task is also known as *Multi-target, Multi-Camera* (MTMC) tracking.

The tracking evaluation consists in comparing the ground truth and computed tracks. To better understand what we will evaluate, Fig. 6.9, presents the manually obtained people tracks of video set 3. In contrast, (b) presents the people tracks generated after passing the set of detections through the WiseNET system.

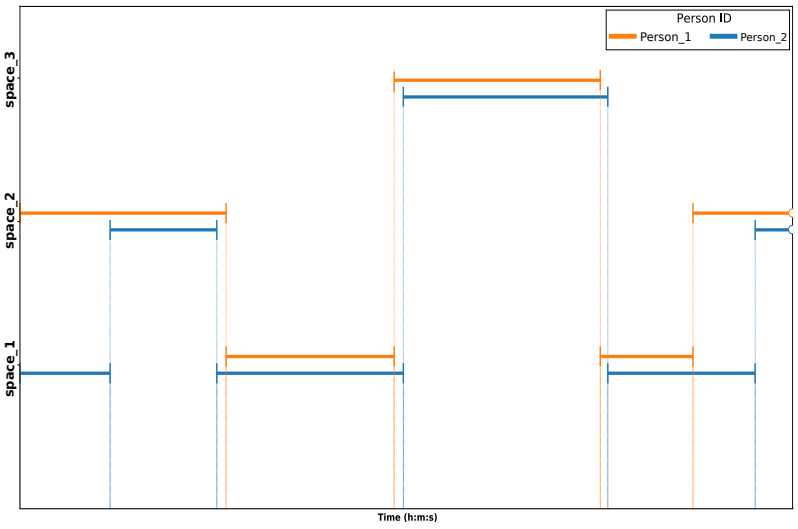
To better understand the tracking evaluation, let us consider Fig. 6.9. The people space-time tracks presented in (a) were obtained manually, by looking directly the video set 3. In contrast, the people tracks presented in (b) were generated automatically, after passing a set of detections through the WiseNET system. Therefore, the goal of the tracking evaluation, is to measure "how good" are the tracks computed by the WiseNET system, based on the real people tracks (i.e., the manually obtained/ground truth ones). The evaluation consist in firstly deciding the best match between the computed tracks and the ground truth tracks. In this example, the computed track of `Person_18` should be match to the ground truth track of `Person_1`, and the track of `Person_24` with the track of `Person_2`. Once the match is done, then every frame in which the ground truth tracks are assigned to a wrong computed track will be consider as an error. More details will be presented below.

TRACKING METRICS

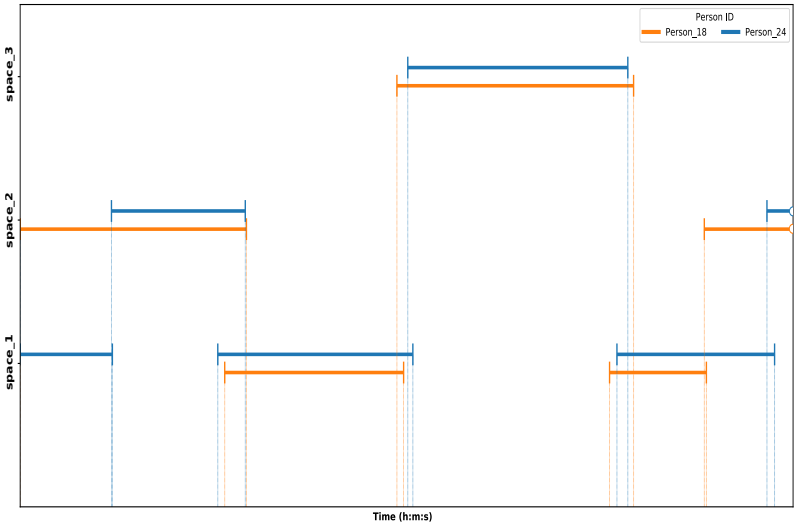
There exits two types of metrics used to for evaluating the performance of MTMC trackers:

- **Event-base metrics:** focus on how often a tracker makes wrong decisions. This type of metrics help to pinpoint the source of some errors. The well-known CLEAR MOT metrics are an example of this measurements [11]. These metrics measures the performance by considering the number of track's *fragmentation* (i.e., if a tracker switches/changes the identity), tracks *merging* (i.e, if two tracks with different identity are merged into a single one, is the opposite of fragmentation), and track *mismatch* (i.e., the total number of wrong decisions, that is the sum of fragmentations and merging).

- **Identity-based metrics:** focus on how well a tracker can determine *who* is *where* at all times. This type of metrics are more interested in the preservation of identities, thus they evaluate how well computed identities conform to true identities, while disregarding where or why mistakes occur. These types of metrics were introduced recently by Ristani et al. in [146].



(a) Ground truth tracks



(b) Computed tracks

Figure 6.9 – Comparison of ground truth tracks (a) and computed tracks (b) of video set 3.

To observe the difference between the two types of metrics, we will considered the scenario abstractly presented in Fig 6.10, which was taken from [146]:

> In the scenario, airport security is following suspect A spotted in the airport lobby. They need to choose between two trackers, (a) and (b). Both tag the suspect as identity 1 and track him up to the security checkpoint. System (a) makes a single mistake at the checkpoint and henceforth tags the suspect as identity 2, so it loses the suspect at the checkpoint. After the checkpoint, system (b) repeatedly flips the tags for suspect A between 1 and 2, thereby giving police the correct location of the suspect several times also between the checkpoint and the gate, and for a greater overall fraction of the time. Even though system (a) incurs only one ID switch, airport security is likely to prefer system (b), which reports the suspect's position longer—multiple ID switches notwithstanding—and ultimately leads to his arrest at the gate.

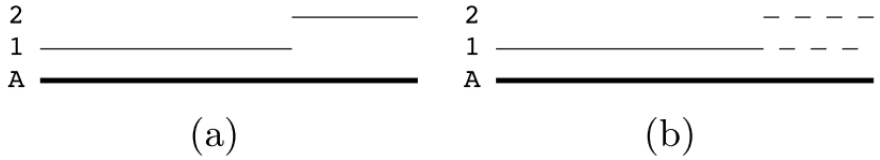


Figure 6.10 – Comparison of two trackers (a) and (b). There is one true identity **A** (thick line, with time in the horizontal direction), a tracker may mistakenly compute identities 1 and 2 (thin lines) broken into two fragments (a) or into eight (b). Identity 1 covers 67% of the true identity's trajectory in (a) and 83% of it in (b). The figure was taken and adapted from [146].

In the presented scenario, *event-based metrics* charge one fragmentation error to (a) and 7 to (b), thus (a) is considered a much better tracker than (b). In the other hand, *identity-based metrics* charge 33% of the length of **A** as tracking error to (a) and 17% to (b), thus (b) is considered a better tracker than (a). Moreover, as stated in [146], there is no one type of metric better than the other, but rather each of them serve different purposes. Thus, the choice of metrics depends on what needs to be measured and on the desired application. Consequently, **based on our Intelligent Video Surveillance (IVS) application, where we are interested in knowing the position of people at all time, we decided to use the *identity-based metrics* to measure the performance of our system**.

The identity-based metrics use a *truth-to-result matching* criterion to measure the performance not by *how often* mismatches occur, but by *how long* the tracker correctly identifies the targets [146]. Where *truth* are the ground truth tracks and *result* are the computed tracks.

As observed in Fig 6.10, the key mistake made by the trackers is to see two identities where there is one. To quantify the extent of the mistake, we need to decide which of the two computed identities we should match with A for the purpose of performance evaluation. Once that choice is made, every frame in which A is assigned to the wrong computed identity is a frame in which the tracker is in error. In both cases in Fig 6.10, the most favorable choice is to tie A to 1, because this choice explains the largest fraction of A. Once this choice is made, we measure the number of frames over which the tracker is wrong—in the example, the number of frames of A that are not matched to 1. In Fig 6.10, this measure makes (b) better than (a). This penalty is consistent because it reflects precisely what the choice made above maximizes, namely, the number of frames over which the tracker is correct about who is where. In (a) the tracker matches ground truth 67% of the time, and in (b) it matches it 83% of the time [146].

To compute the optimal truth-to-result match, we constructed and solved a bipartite graph as shown in [146]. Roughly, the procedure is as follows. First, consider a set of true tracks V_T, which has a track $\tau \in V_T$; and a set of computed tracks V_C, which has a computed track $\gamma \in V_C$. Then, two tracks have a match cost $e \in E$ if their trajectories overlap in time, where E is the set of matching costs.

The cost of the matching $(\tau, \gamma) \in E$ tallies the number of false negative and false positive frames that would be incurred if that match were chosen as correct. Specifically, let $\tau(t)$ be the the sequence of detections for true track τ, one detection for each frame t in the set $\mathcal{T}_\tau$ over which τ extends, and define $\gamma(t)$ for $t \in \mathcal{T}_\gamma$ similarly for computed trajectories [146].

The two simultaneous detections $\tau(t)$ and $\gamma(t)$ are a consider as *miss* if they do not overlap in the space, and is written as

$$m(\tau, \gamma, t, \Delta) = 1 \ . \tag{6.1}$$

While, if there is no miss, is written as

$$m(\tau, \gamma, t, \Delta) = 0 \ . \tag{6.2}$$

Where Δ is the overlapping threshold between detections. For example, it can be the minimum overlapping area between the detections bounding boxes, or the minimum distance between the detections coordinates. However, in our particular case Δ is not used, because we consider a lower space resolution, meaning that we declare a *miss* if the detections *are not performed in the same space location*.

With this definition, the cost of matching $(\tau, \gamma \in E)$ is defined as follows:

$$c(\tau, \gamma, \Delta) = \underbrace{\sum_{\tau \in \mathcal{T}_\tau} m(\tau, \gamma, t, \Delta)}_{\text{False Negative}} + \underbrace{\sum_{\tau \in \mathcal{T}_\gamma} m(\tau, \gamma, t, \Delta)}_{\text{False Positive}} \ . \tag{6.3}$$

Where the first part of the equation measures the number of *misses* by considering the detections in the true track, i.e., the false negatives. While the second part measures the number of *misses* by considering the detections in the computed track, i.e., the number of false positives. Finally, one-to-one matching between the true and computed tracks are obtained by determining a minimum-cost solution that minimizes the cumulative false negative and false positive errors [146].

Every (τ, γ) match is a True Positive ID ($IDTP$). Every γ without a match is a False Positive ID ($IDFP$). And every τ without a match is a False Negative ID ($IDFN$). In general not every trajectory is matched. The sets $MT = \{\tau | (\tau, \gamma) \in IDTP\}$ and $MC = \{\gamma | (\tau, \gamma) \in IDTP\}$ contain the matched ground truth tracks and matched computed tracks, respectively. The pairs in $IDTP$ can be viewed as a bijection between MT and MC. In other words, the bipartite match implies functions $\gamma = \gamma_m(\tau)$ from MT to MC and $\tau = \tau_m(\gamma)$ from MC to MT [146].

Finally, the counts of $IDFN$, $IDFP$ and $IDTP$, will be used to compute the Identification Precision (IDP), the Identification Recall (IDR), and the corresponding F_1 score (IDF_1), as follows:

$$IDFN = \sum_{\tau \in MT} \sum_{t \in \mathcal{T}_\tau} m(\tau, \gamma_m(\tau), t, \Delta) \tag{6.4}$$

$$IDFP = \sum_{\gamma \in MC} \sum_{t \in \mathcal{T}_\gamma} m(\tau_m(\gamma), \gamma, t, \Delta) \tag{6.5}$$

$$IDTP = \sum_{\tau \in MT} len(\tau) - IDFN = \sum_{\gamma \in MC} len(\gamma) - IDFP \tag{6.6}$$

$$IDP = \frac{IDTP}{IDTP + IDFP} \tag{6.7}$$

$$IDR = \frac{IDTP}{IDTP + IDFN} \tag{6.8}$$

$$IDF_1 = \frac{2\ IDTP}{2\ IDTP + IDFP + IDFN} \tag{6.9}$$

Identification precision (recall) is the fraction of computed (ground truth) detections that are correctly identified. While, Identification F_1 score is the ratio of correctly identified detections over the average number of ground truth and computed detections [146]. Therefore, we will use these three metrics to evaluate the tracking performance of our system.

RESULTS AND DISCUSSION

To perform the evaluations, we modified a Matlab[6] implementation of the metrics developed by the authors Ristani et al. [146].

The evaluations were performed on each video set of the WiseNET dataset. For each video set the *people tracking manual annotations* are considered as the *ground truth tracks* (see Section 6.1.2). While the *computed tracks* are obtained after passing the *people detection manual annotations* through the dynamic population process presented in Section 5.3. Furthermore, the evaluations are performed after inserting the complete contextual information, i.e., the information of the I3M environment and the general information of the camera nodes.

The videos of the dataset are quite challenging, they have sudden changes in light conditions, people suddenly appearing/disappearing from the camera's field of view, people are partially observed, different view-point variations, large distance between the target and the cameras, etc. The WiseNET tracking results on each video set are presented in Table 6.8. For the three metrics, a high value is better. In general, the results show that **the WiseNET system manage to successfully track people while they moved through different spaces**. Specially, the system presents high results in video **sets 1, 2, 3, 4, 8,** and **9**. In these video sets, there are people which are seated throughout the set and people which are moving, alone or in groups of two, through multiple spaces. These video sets present standard indoor scenarios, which are considered by the WiseNET system. However, the rest of the video sets, present some scenarios which were not considered during the system design, thus, they are interesting for discussion:

- The video **sets 5** and **11**, are the most challenging video sets in the dataset. There are 14 and 15 people moving as a group through multiple spaces. Both video sets start by all people going out from a blind space. A situation which was not considered in the WiseNET system, thus those people tracks were missed. This translate in a high count of $IDFN$, thus resulting in a low value of IDR. Moreover, throughout the video sets there are multiple occlusions due to people being in front of others, which resulted in multiple fragmentations of people tracks and multiple tracks which were missed. This translate in a high count of $IDFP$ and $IDFN$, thus resulting in low values of IDP and IDR.

[6] https://www.mathworks.com/products/matlab.html

Table 6.8 – Tracking results on each video set, using identity-base metrics.

	$IDP \uparrow$	$IDR \uparrow$	$IDF_1 \uparrow$
Set 1	97.97	100.00	98.97
Set 2	96.12	99.30	97.69
Set 3	93.21	100.00	96.49
Set 4	97.90	99.07	98.48
Set 5	73.89	56.85	64.26
Set 6	100.00	86.82	92.95
Set 7	100.00	82.75	90.56
Set 8	98.64	97.63	98.13
Set 9	99.97	99.19	99.58
Set 10	92.57	88.24	90.35
Set 11	73.98	54.68	62.88

- Before starting the video **set 6**, there were two people which were not visible by any camera. And they become visible after some time. In the ground truth, those people were considered inside the space, through the whole video set. However, it is impossible for the WiseNET system to know what happened before the video set. Thus, it only considered those people when they appeared in the cameras. As a result the system "missed" a part of the tracks ($IDFN$), which had an impact in its IDR value. Moreover, each generated people track had a correct match in ground truth, resulting in perfect IDP value.

- Similarly to video set 6, in video **set 7**, there was one person in a blind space before starting the recording, and it stayed there for around 80% of the time. It was impossible for the WiseNET system to know that there was somebody inside a blind space, thus it "missed" a part of its track, which had a negative impact in its IDR value.

- In video **set 10**, there is one person which enters to a blind space, stays there some time, and then go out of it. This situation of "somebody leaving a blind space" was not considered in our system, thus, (1) the person track is always considered in the blind space, and (2) a new person track is created when it re-enters the non-blind space. Both actions, translate in high $IDFP$ count, which results in lowering the IDP value. Furthermore, a person is also not visible at the beginning of the video, thus impacting in the IDR value.

To conclude, people tracking using semantic information is possible and gives promising results. The WiseNET system allow to know the location of people even if they are not visible by any camera. However, if the scenarios of the video sets were not consider in the system (e.g., people going out of a blind space), then the performance decreases.

Video sets 5 and 11 test the system's limitation by their complexity. However, to test further the limitations, the next section will evaluate the influence of using real people detections, instead of the manual ones used in this section.

6.3/ PEOPLE DETECTOR IMPACT

The WiseNET system is independent of the method used to obtained the people detections. In the previous section, we evaluated the system performance by using the *people detections* obtained *manually* (referred as PD-MAN in Section 6.1.2). Instead, in this section we will evaluate the system by using *automatically* generated *people detections*, obtained by using real people detector models.

The use of real people detectors aims to: (1) propose an alternative to the time consuming manual annotation task, (2) to evaluate the complexity of each video (in terms of difficulty to detect people) by using state-of-art people detectors (Section 6.3.1), and (3) to evaluate the influence of the people detections in the WiseNET system (Section 6.3.2).

The automatic people detection (PD-AUT) annotations were obtained by passing each video frame through a set of pre-trained people detector models. This resulted in a set of automatically generated Bboxes, that (ideally) represent people. We used three off-the-shelf detector models, the well-known Histogram of Oriented Gradients (HOG_SVM) [42], as well as two state-of-the-art CNN-based models: Single Shot Detector (SSD) [103], and the You-Only-Look-Once version 3 (YOLOv3) [142].

The HOG_SVM detector is based on HOG feature descriptors and Support Vector Machine (SVM) in order to detect people [42]. We used the implementation provided by the OpenCV library.[7] We chose this detector due to its popularity in the computer vision community and its low complexity, which results in a reduced processing time.

The SSD is a one-stage detector that extracts the feature map of the complete image, then applies a sequence of multi-scale convolutional layers and anchor boxes in order to classify the different regions of the feature map [103]. We used the pre-trained model—configuration and weights—provided by the authors.[8] Specifically, we used the model with input image size of 512 × 512, thus, we will referred to it as SSD_512. This model was first trained on the COCO dataset (Common Objects in Context)[9] and then fine-tune on the union of PASCAL VOC2007 and VOC2012 dataset.[10] We chose this detector model due to it's high precision as presented in [103].

The YOLOv3 uses a single neural network that predicts Bboxes and class probabilities directly from full images. We used the pre-trained model—configuration and weights—provided by the authors.[11] Specifically, we used the model with size 608 × 608, thus, we will referred to it as YOLOv3_608. This model was trained on the COCO dataset. We chose this detector model due its high precision and low inference time [142], which are two crucial factors for a real-time surveillance system.

For the SSD and YOLO detectors, only the *person* class was considered, i.e., the rest of objects were simply ignored. More details on each detector can be found in the people detection background presented in Section 2.3.1.

A software was developed that takes the different detectors and videos, and automatically generates files similar to Listing 6.1, where the value of the *imageProcessing* field is the name of the detector —HOG_SVM, SSD_512 or YOLOv3_608. **Thus, for each**

[7] https://docs.opencv.org/3.4.1/d5/d33/structcv_1_1HOGDescriptor.html
[8] https://github.com/chuanqi305/ssd
[9] http://cocodataset.org/
[10] http://host.robots.ox.ac.uk/pascal/VOC/
[11] https://github.com/pjreddie/darknet/blob/master/cfg/yolov3.cfg

video in the dataset there are four people detection files, one obtained by manually detecting people (PD-MAN), and three obtained by automatically detecting people (PD-AUT) using the HOG_SVM, SSD_512 and YOLOv3_608 detectors. The software that plugs-in the detectors was developed in Python using OpenCV library, and the code is provided in the WiseNET dataset. Moreover, the automatic detections were obtained using a machine with the following configuration: Intel Core i7-4790 CPU @3.6GHz $\times$ 4, 16GB of RAM and a "Java Heap" size set to 200MB.

6.3.1/ COMPARISON OF DETECTORS

The choice of detectors differs in complexity and robustness, which we consider an interesting evaluation factor for the WiseNET system. Therefore, we will evaluate the performance of the three automatic detectors—HOG_SVM, SSD_512 and YOLOv3_608—with respect to the manual annotations, which are considered as the ground truth. This evaluation allows to (1) compare the different detectors methods, (2) to validate the usability and quality of the WiseNET dataset, and to (3) evaluate the complexity of each video in terms of their difficulty to detect people.

To carry out the evaluation, a *matching value* needs to be obtained between the resulting Bboxes (coming from a PD-AUT method) and the ground truth Bboxes (coming from the PD-MAN). The possible outcomes of a match are:

- True positive (TP), a person Bbox is present in the ground truth and in the PD-AUT annotations (also called correct detection).
- False positive (FP), a person Bbox is present in the PD-AUT annotations, but is *not* present in the ground truth (also called false alarm).
- False negative (FN), a person Bbox is present in the ground truth, but is not present in the PD-AUT annotations (also called missed detection).
- True negative (TN), a person is not present in the ground truth, neither in the PD-AUT annotations.

The matching value depends of the overlapping between pairs of Bboxes. In a few words, if there are two Bboxes, one coming from the PD-MAN annotations (B_{man}) and the other one from the PD-AUT annotations (B_{aut}), the measure of Intersection Over Union (IOU) between their areas is defined as,

$$IOU = \frac{area(B_{man} \cap B_{aut})}{area(B_{man} \cup B_{aut})}. \tag{6.10}$$

Consequently, the matching between B_{man} and B_{aut} is defined as a TP or FP depending on the IOU value and an overlapping threshold IOU_{Th}, as follows:

$$matching(B_{man}, B_{aut}) = \begin{cases} TP & \text{if } IOU \geq IOU_{Th} \\ FP & \text{if } IOU \leq IOU_{Th} \end{cases} \tag{6.11}$$

After having the matching value for each pair of Bboxes, the performance of the detector is evaluated by computing different metrics. The metrics used for the evaluation were

the standard *Precision/Recall curve* and the *Average Precision*. These are well-known metrics proposed by the Pascal VOC challenge [52]. To determine these metrics, the detector's *precision* and *recall* values need to be computed. *Precision* is the fraction of detected items that are correct, and is defined as:

$$Precision = \frac{TP}{TP + FP} \tag{6.12}$$

Recall is the fraction of items that were correctly detected among all the items that should have been detected, and is defined as:

$$Recall = \frac{TP}{TP + FN} \tag{6.13}$$

Furthermore, the *confidence* of each detection also needs to be considered. Confidence is the detector's belief in the correctness of the detection.

The **Precision/Recall curve (PR-curve)** is obtained by ordering the detector's results in a descending manner according to their confidence value, and then computing and plotting the accumulated precision and recall values. A detector is considered good if its precision stays high as the recall increases, meaning that if you vary the confidence threshold the precision and recall will stay high. In the other hand, poor detectors will start with high precision value but will decrease as the recall increases.

To compare different PR-curves (detectors) in the same plot is not an easy task, due to the curves tend to cross each other frequently. Thus, a "summary" of the curve will better for comparing.

The **Average Precision (AP)** is a numerical metric that simplifies the comparison between different detectors. AP is obtained by computing the area under the PR-curve, i.e., averaging the precision across all recall values between 0 and 1. The AP value goes from 0 to 1, where higher is better.

Both metrics—PR-curve and AP—depends on a IOU threshold (IOU_{Th}) that determines if a detected instance is a correct or false detection. The Pascal VOC Challenge [52], proposes to set the threshold to 50%. However, the MS Common Objects in Context (COCO) challenge [100] proposes to compute the AP using different thresholds, for example from 50% to 95%, and then to compute an average of the resulting APs to provide a single number as result. We decided to set the IOU_{Th} threshold at 50%, because in this way we take into account the inaccuracies in the manually annotated bounding boxes, as stated in [52].

To perform the evaluations, we used the Python implementation of the metrics available on-line.[12]

The evaluations were performed for each video in the dataset. For exemplification, Fig. 6.11 presents the resulting PR-curves of the three detector on the video set 3. As previously stated, it is difficult to compare the detectors by observing the curves, thus we will focus on the AP metric. The AP values were computed from each PR-curve. Figure 6.12 presents the comparison of the resulting APs for all the videos. Notice that there are some videos without detections (i.e., nobody appeared in the camera's view), thus they were ignored during the evaluation (e.g., the videos from camera 6 in sets 5-11).

Figure 6.13 presents the mean AP per detector. As it can be observed YOLOv3_608 has a better performance followed by the SSD_512, and in the last place HOG_SVM.

[12] https://github.com/rafaelpadilla/Object-Detection-Metrics

YOLOv3_608 has a median AP of around 0.8, and half of the AP values are between 0.6 and 1, which is overall good performance. In the other hand, HOG_SVM has a median AP of around 0.1, and most of its AP values (around 75%) are lower than 0.4.

The detectors can also be compared by the number of Frames Process per Second (FPS). The HOG_SVM has around 20 FPS, YOLOv3_608 around 1.4 FPS, and SSD_512 around 1.1 FPS. Thus, it can be conclude that deep-learning models —YOLOv3_608 and SSD_512—are more complex, thus they required more time to process a frame, than the machine learning one (HOG_SVM).

Furthermore, from the AP results, it is complicated to evaluate the difficulty/challenge degree of each video in the dataset. A statistical analysis should be performed to make conclusions about this (which is left as future work). However, it can observed that video sets 2 and 4 are the less challenging ones, while video set 5 is the most challenging one. Moreover, it can be observed that people detection in cameras 2 and 4 are more challenging than in camera 3.

Finally, it is important to notice that the results presented in this section strongly depend on the quality of the ground truth (PD-MAN annotations), which was done by multiple humans, thus is prompt to subjectivity and errors. Furthermore, to obtain the PD-MAN annotations is a time-consuming task. In contrast, automatic annotations (PD-AUT) are not *subjective* and they can be performed almost without human intervention. Moreover, the results of YOLOv3_608 and SSD_512 are very promising just by using them off-the-shelf, which can be enough for many applications. But if a fine-tuning step using an small amount of data is performed, the results would be much better. Therefore, for all those reasons **we believe that ideal ground truth detections could be obtained by fine-tunning a deep-learning model (e.g., YOLOv3) in an specific scenario. This will required much less time and lower human resources, than doing all the annotations manually**.

6.3.2/ INFLUENCE OF DETECTORS IN THE WISENET SYSTEM

After evaluating each detector independently, we will evaluate the performance of the WiseNET using the different detectors. The evaluation will consists on (1) obtaining the number of people at different timestamps, as presented in the *Dynamic CQs* section (6.2.1.2), and (2) checking the people tracking performance of the system, as presented in the *Tracking with Semantics* section (6.2.2). Notice that we will *not* consider the *Static CQs* because their answers do *not* depend on detections performed, thus the type of detector used has no influence on them.

The output of the automatic detectors is a set of Bboxes (B_{aut}). However, there are some parameters that need to be defined in order to use them in the WiseNET system:

- **Visual descriptors:** any type of visual features could be use to describe the detection Bbox. We decided to use the combination of Hue-Saturation (HS) and Red-Green-Blue (RGB) color histograms. This choice was done due to, as presented in Section 2.3.1, color information is widely used and is considered as the most important feature for describing people detections [61, 65, 91, 101, 107, 114, 186]. Furthermore, the HS color channels are considered as the most discriminative cues given the illumination changes. The visual descriptor is composed of 128 features, 64 from the HS histogram and 64 from the RGB. In this way, both color histograms

have the same weight.

- **Distance metric:** to determine the correspondence/similarity between two visual descriptors, a distance metrics needs to be defined. We decided to use the *unsupervised* Bhattacharyya distance metric. As presented in Section 2.3.1, for histogram-based features the Bhattacharyya distance is the most used metric [28, 65, 54]. The Bhattacharyya distance d_B, between two visual descriptors P and Q is defined as,

$$d_B = -ln \sum_{i=1}^{n} \sqrt{P_i Q_i} \,, \tag{6.14}$$

where n is the total number of features in the descriptor (in our case 128). Moreover, a distance threshold was empirically set to 50%.

Afterwards, for each detector and for each video set, the dynamic population process is performed.

PEOPLE PRESENCE EVALUATION

The first evaluation consists in determining the people presence by using the heatmap-query. As presented in the *Dynamic CQs* section, the goal of this evaluation is to observe if the system is able to deduce the correct number of people in each space, at different times. As done previously, the video set 3 will be used for this evaluation, along with the timestamps. Table 6.9 presents the results of the heatmap-query for each detector. From the results, it can be observed that:

- At `space_1`, the WiseNET system using YOLOv3_608 and SSD_512 detectors, correctly answered the question at (mostly) all times. However, the use of HOG_-SVM detector resulted in wrong answers. Mostly, due to a merge in a people tracks (it considered mostly one person in all the set) and a constant false detection due to a particular light reflection.

- At `space_2`, the system using YOLOv3_608 and SSD_512 returned a wrong answered at t_6. These detectors were able to detect people before entering the `space_2` (few time before t_6), which resulted in more time to consider that a person entered the space. However, at t_7 they answered correctly. In contrast, HOG_-SVM was not able to perform detections before t_6 (the detections were too small for HOG_SVM to detect them), which resulted in a faster consideration that a person was in `space_2`.

- At `space_3`, the WiseNET system using YOLOv3_608 and SSD_512 detectors, correctly answered the question at (mostly) all times. At t_2 and t_5, YOLOv3_608 took a bit longer during the transition between spaces, thus resulting in wrong answers. Moreover, notice that at t_4 the system was able to understand that the person was still in the space, even if it was not detected from some seconds. In the other hand, HOG_SVM merged both people tracks, thus resulting in a single person in the space.

Table 6.9 – Heatmap results using different detectors. The results were obtained using video set 3. The spaces 4, 5 and 6 where omitted since no person was presence on them throughout the set.

	Detector	t_1	t_2	t_3	t_4	t_5	t_6	t_7
space_1	**ground_truth**	**2**	**1**	**0**	**0**	**2**	**1**	**0**
	HOG_SVM	1	1	1	1	1	1	1
	YOLOv3_608	2	1	0	0	2	1	0
	SSD_512	1	1	0	0	1	1	0
space_2	**ground_truth**	**0**	**0**	**0**	**0**	**0**	**1**	**2**
	HOG_SVM	0	0	0	0	0	1	2
	YOLOv3_608	0	0	0	0	0	0	2
	SSD_512	0	0	0	0	0	0	2
space_3	**ground_truth**	**0**	**1**	**2**	**2**	**0**	**0**	**0**
	HOG_SVM	0	0	1	1	1	0	0
	YOLOv3_608	0	0	2	2	1	0	0
	SSD_512	0	1	2	2	0	0	0

From this evaluation it can be concluded that the WiseNET system using the YOLOv3_-608 and SSD_512 detectors, were able to correctly deduce the presence of people at different times. Moreover, the results of HOG_SVM were not as good as the rest. This could be due to its low detection rate, specially in the `space_1` (see set 3-cam1-2 in Fig 6.12).

PEOPLE TRACKING EVALUATION

The second evaluation consists in measuring the tracking performance of the WiseNET using the different detectors. The evaluation process is the same as followed in the Section 6.2.2. Table 6.10 presents the results obtained using the different detectors. From the results, it can observed that:

- The system using **HOG_SVM** presents in all video sets a better IDP value in compare to the IDR. Meaning that most of the tracks generated were correct but it misses many of them. For example, in sets 6, 7, 8 and 9, it obtained an a perfect IDP values and a low IDR value, thus resulting in a low IDF_1 value. Moreover, by considering the low detection rate of HOG_SVM (see Fig. 6.12), we expected to have a low IDR value.
- The system using the **YOLOv3_608** presents, mostly good results, meaning that it manage to correctly track people. In sets 1, 5 and 11, the IDR values are quite low, which could also be expected by observing the Fig. 6.12. Furthermore, the IDP values of sets 5 and 11 are quite low, this was also expected due to the high complexity of the video sets.
- The system using the **SSD_512** presents, in general, good results. Similarly to the previous case, the IDR values are quite low for sets 1, 5 and 11; as well as the IDP for sets 5 and 11.

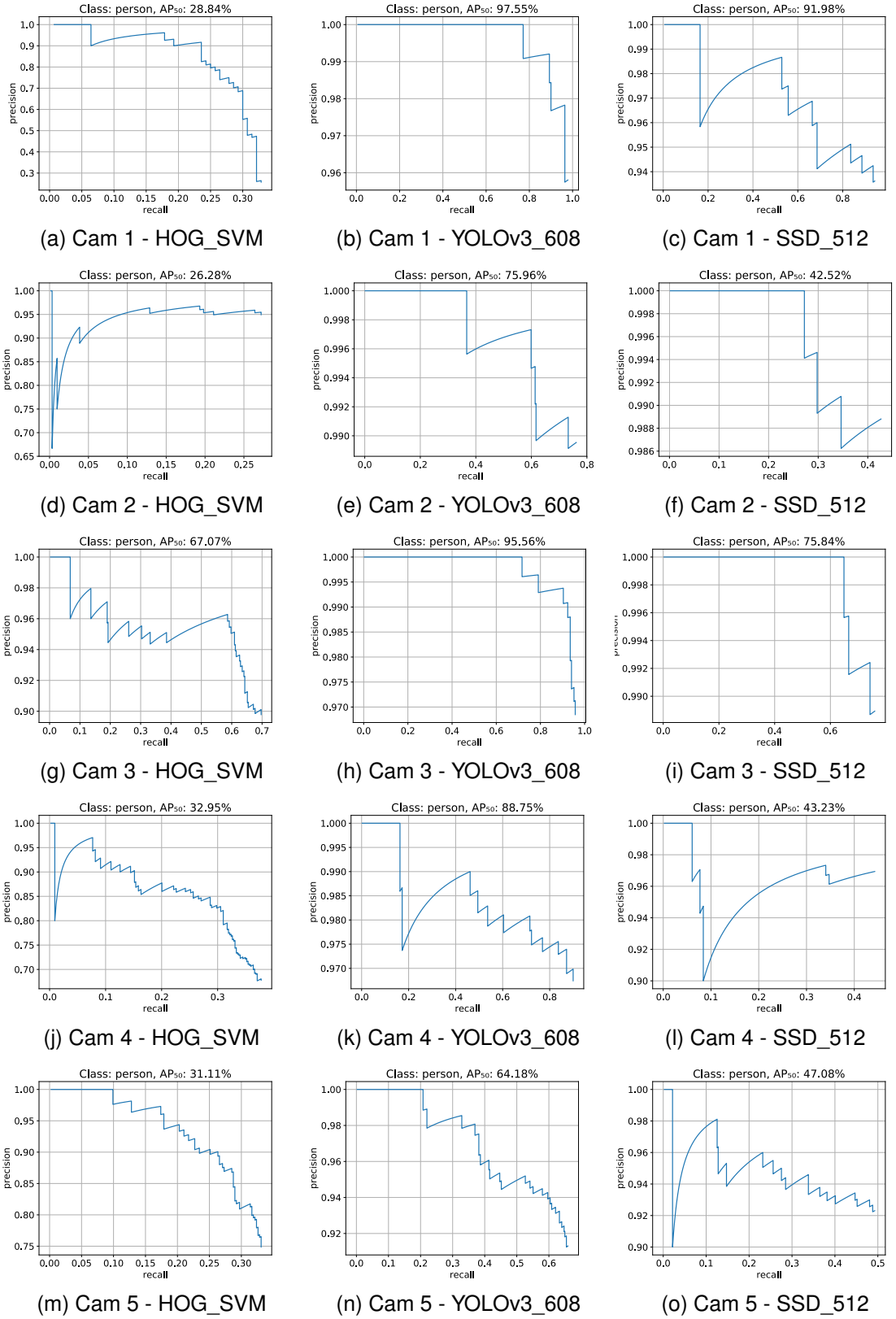
From the results, it can be concluded that **is very important to have multiple metrics during a comparison, specially one that balance different factors/metrics, in our**

case the IDF_1 metric. For example, consider sets 6, 7, 8 and 9, if just the IDP metric will be considered, then HOG_SVM will have the better performance, which is by far not the case. Also, consider sets 9 and 10, where HOG_SVM has the best IDP, YOLOv3_-608 has the best IDR, and SSD_512 has the best balance between both metrics thus resulting in the best IDF_1 value.

Also, we can conclude that **people tracking using a semantic-based system with real detectors is possible. This results allow to validate the proof of concept of our system, which fusions the results of computer vision with contextual information**. However, the detector quality do have an influence on the performance of the system. For example, YOLOv3_608 and SSD_512 are much better detectors than HOG_SVM, and this difference can be also be seen in the results of the WiseNET system.

Table 6.10 – Tracking results using the different detectors. We highlighted in bold the best result for each video set.

	Detector	$IDP \uparrow$	$IDR \uparrow$	$IDF_1 \uparrow$
Set 1	HOG_SVM	51.39	10.28	17.13
	YOLOv3_608	**71.52**	**29.63**	**41.90**
	SSD_512	58.82	23.74	33.83
Set 2	HOG_SVM	64.17	53.40	58.29
	YOLOv3_608	**97.93**	**91.32**	**94.51**
	SSD_512	55.15	64.86	59.61
Set 3	HOG_SVM	49.62	6.52	11.52
	YOLOv3_608	**91.60**	46.68	**61.84**
	SSD_512	58.07	**48.35**	52.77
Set 4	HOG_SVM	45.65	6.40	11.23
	YOLOv3_608	**78.42**	29.98	43.38
	SSD_512	76.12	**59.80**	**66.98**
Set 5	HOG_SVM	32.27	3.17	5.78
	YOLOv3_608	28.95	**40.95**	33.92
	SSD_512	**38.37**	36.97	**37.66**
Set 6	HOG_SVM	**100.00**	16.67	28.57
	YOLOv3_608	82.67	**96.03**	**88.85**
	SSD_512	81.91	77.18	79.47
Set 7	HOG_SVM	**100.00**	4.95	9.43
	YOLOv3_608	81.68	**77.42**	**79.49**
	SSD_512	88.41	68.73	77.34
Set 8	HOG_SVM	**100.00**	28.03	43.79
	YOLOv3_608	74.33	69.33	71.74
	SSD_512	80.65	**72.18**	**76.18**
Set 9	HOG_SVM	**100.00**	21.70	35.66
	YOLOv3_608	72.62	**89.94**	80.35
	SSD_512	92.39	88.24	**90.26**
Set 10	HOG_SVM	**90.43**	71.41	79.58
	YOLOv3_608	57.94	**92.07**	71.12
	SSD_512	71.67	91.79	**80.49**
Set 11	HOG_SVM	**56.36**	23.82	33.49
	YOLOv3_608	36.56	**37.25**	**36.90**
	SSD_512	39.95	33.84	36.64



(a) Cam 1 - HOG_SVM (b) Cam 1 - YOLOv3_608 (c) Cam 1 - SSD_512

(d) Cam 2 - HOG_SVM (e) Cam 2 - YOLOv3_608 (f) Cam 2 - SSD_512

(g) Cam 3 - HOG_SVM (h) Cam 3 - YOLOv3_608 (i) Cam 3 - SSD_512

(j) Cam 4 - HOG_SVM (k) Cam 4 - YOLOv3_608 (l) Cam 4 - SSD_512

(m) Cam 5 - HOG_SVM (n) Cam 5 - YOLOv3_608 (o) Cam 5 - SSD_512

Figure 6.11 – Precision/Recall curves for each camera video (rows) in set 3 obtained using HOG_SVM, YOLOv3_608 and SSD_512 detectors (each column, respectively).

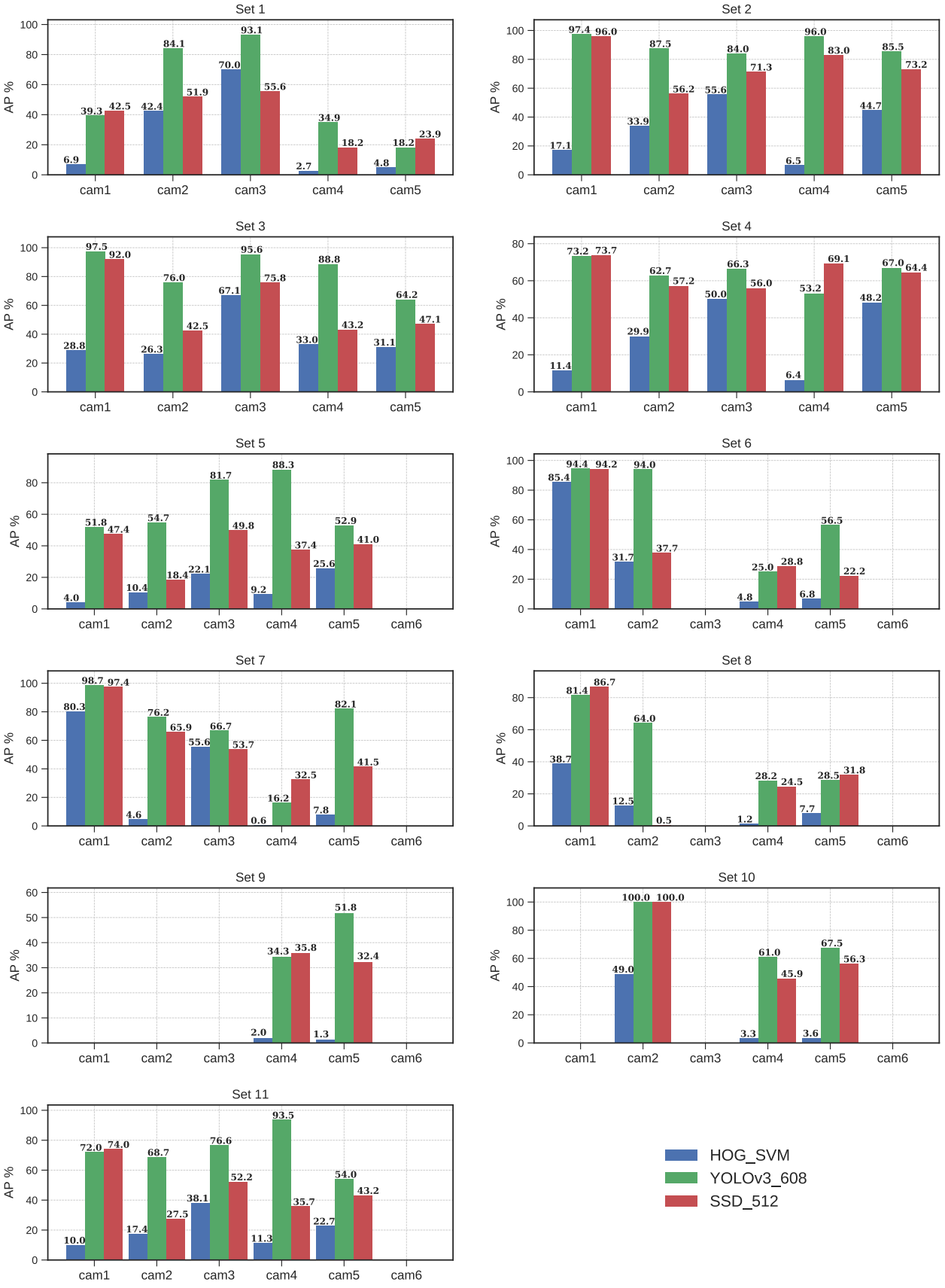


Figure 6.12 – Average Precision (AP) comparison of HOG_SVM (in blue), YOLOv3_608 (in green) and SSD_512 (in red) people detectors, in all the video sets. The videos without detections were ignored, thus no result is shown.

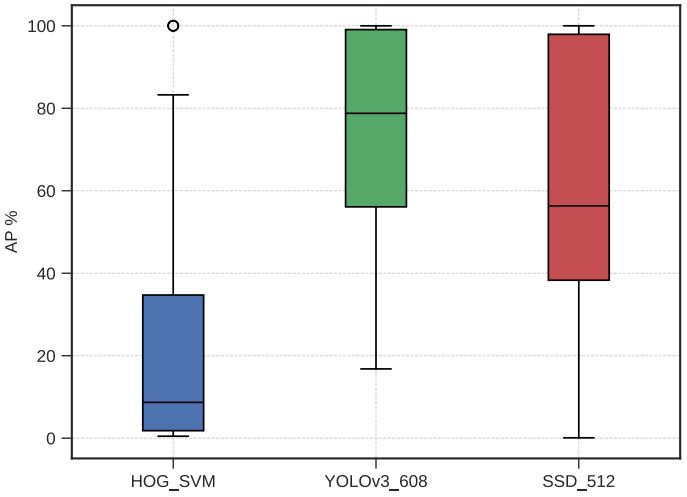


Figure 6.13 – Mean Average Precision (AP) of HOG_SVM (in blue), YOLOv3_608 (in green) and SSD_512 (in red) people detectors.

6.4/ CONCLUSION

In this section we presented different evaluations of the WiseNET system, as well as a dataset which was created for the evaluation.

The WiseNET dataset, consists of three main elements: (1) video sets, (2) information of the environment and (3) annotations for people detection and people tracking. To our knowledge, the WiseNET dataset is the only multi-camera multi-space dataset that provides the complete environment and contextual information. Moreover, the WiseNET dataset is a contribution to the computer vision community, due to its large panel of applications, such as, people detection, re-identification, tracking, human-action recognition, office object detections, etc.

The first WiseNET evaluation, consisted in checking if the system could correctly answer the question it was designed for. The questions concerned the built environment, the sensors deployed in it and building usage. These type of questions allow to interact with an smart building and to monitor in real-time what is happening. Due to the semantic nature of the system, the questions were defined using the SPARQL formalism. This formalism allows to interact and extract information from the ontology and perform reasoning.

The system was able to correctly answered all the questions, due to the semantic-links between the information populated and the predefined human-skills rules. The questions used during the evaluation, open the door of many applications, specially for building managers. For example, they could be used to tell: the number of spaces (elements) in a storey (space); which doors should be used to go from one space to another (this could be used for firemen); to determine that maximum number of people is not exceeded (people count); a what time there is more people in a space; the flow of people; count the number of times a door is used, to schedule its maintenance; etc. All those applications can be performed while protecting the private life of the building users, by not sending any image. Furthermore, a monitor unit was developed, to agglomerate multiple services. The monitor unit is an intuitive interface designed for building managers and technicians who may not be fluent with information technology.

The second evaluation, consisted in checking the performance of the semantic-base system in tracking people. The evaluation was done using *identity-based metrics*, which measure *how long* the system was able to correctly track the identities of people. This type of metrics fits the Intelligent Video Surveillance (IVS) application where we are interested in knowing the position of people at all times. Furthermore, with this type of metrics, we can make use of the previously obtained space-time graphs.

The system presented was able to correctly track people while they moved through different spaces. In most of the video sets, the results were very good. Specifically, the results were near perfect in video sets 1, 2, 3, 4, 8 and 9, which presented scenarios that are common in indoor environments. However, there were some uncommon situations (where state-of-art computer vision algorithms struggle) that were not considered during the system design, thus, resulting in some incorrect results. Situations such as, high number of people (in some videos 14) moving together, multiple and constant occlusions due to people clutter, people going out from blind spaces. Also, there are some situations which were impossible for the system to know, for example, the presence of people in blind spaces before starting the recording. The advantage of the semantic-based system is that it allows to easily pinpoint the sources of errors and, if required, modify the system accordingly.

From the previous evaluations, it can be concluded that, the semantic combination of data coming from a smart camera network with the data of the environment, allows to a smart building to reason by itself, by using a formalism easy to understand by humans. Making it a "real" smart building. The combination of knowledge allows a dialog and interventions of building managers by executing queries. Moreover, the addition of contextual information to what the cameras observe, allows to overcome computer vision problems, specifically the occlusion problem that occurs when a persons goes outside the field of view of a camera.

The limitation of the system were also evaluated, specifically the influence of the quality of detections. Our goal was not to improve people detectors or to develop a new deep learning architecture. Our goal was to develop a framework *independent* of the people detector model. Therefore, we decided to use off-the-shelf people detectors methods and insert them into our system, kind of a plug-and-play. A comparison between the different off-the-shelf methods was presented before evaluating their influence in the system. From the tree people detectors tested, the deep-learning based models —YOLO and SSD–present much better detection performance compared to the feature based one–HOG. Furthermore, YOLO has a better capacity of performing small detections, which results in a higher detection rate compared to SSD. Also, something that was noticed during the experimentations, was that the deep-learning models presented (almost) none false detections, in contrast to the feature based models. However, the HOG is much faster compared to the other two, this is might be due to the higher complexity of deep-leaning models.

From the results obtained using the different detectors, it can be concluded that the system is able to answered some general questions concerning the building usage, for example the number of people in a space. However, the system present some problems in tracking and keeping the identity of people. This is mainly due to the re-identification task, which depends on the type of visual descriptors use. Thus, suggesting that a more robust visual descriptor should be used. Also, some tracking errors—specifically using HOG—are due to the low detection frequency and the presence of false detection. Furthermore, to determine the influence of the visual descriptors, more tests should be performed using different features and distance metrics.

Finally, it can be concluded that the quality of the people detectors has an influence on the result of the system. Even with low quality detectors (e.g., HOG) the system is able to answer correctly most of the questions related to the building usage. The choice of detectors could be guided by the computing resources available, the processing time or the required accuracy of results.

7

CONCLUSIONS AND FUTURE WORK

Currently, the most common application of a Visual Sensor Network (VSN) deployed in a smart building, is video surveillance. However, as the size of the network increases, it is almost impossible for a human operator to monitor what is happening in real-time. Furthermore, the images observed by the operator lack of contextual information, which in a built environment is an important factor to understand what is happening and to take decisions. In this context, we proposed semantic-based (symbolic AI) system, that combines the information coming from a VSN with contextual information of the environment. The proposed system enables semantic interoperability between multiple and heterogeneous sources of information by using an ontology model. Furthermore, the semantic-based system was created using human-knowledge, thus allowing a direct understanding and control of the decision process. In addition, the proposed system could be interrogated using a human understandable language, thus bridging the human-AI language gap. Consequently, new smart services are proposed, which ease the tasks of building managers. The final question concerning the combination of both types of AI—symbolic and non-symbolic—is still left open, however, we took the first steps to answer it.

In Chapter 2, we discussed about the Visual Sensor Networks (VSN), Intelligent Video Surveillance (IVS), Smart Cameras (SC) and computer vision. As concluded in this chapter, the most prominent way of achieving the IVS vision is by using Smart Cameras to capture high-level descriptions of a scene and analyze them in real-time by employing different computer vision methods. Specially, people detection and tracking are key tasks for most IVS systems. Furthermore, as presented in the state-of-the-art, the person detection problem is almost solved, specially by considering the results of deep-leaning based models. However, one important consideration is that the accuracy of the detectors depends highly on the resources available—processing resources (hardware) and data for training. Furthermore, one main drawback of current people tracking methods is the disregard of semantic contextual information which can be very useful for an IVS system. Consequently, we consider that an IVS system deployed in a smart building should be aware of its context to automatically adapt its functionality according to the contextual changes.

Chapter 3 introduces the notions of context and context-aware system in the built environment, as well as, the ontology world. As discussed in this chapter, a way of obtaining the contextual information concerning the building's structure, is by using the Industry Foundation Classes (IFC) file that represents the environment's Building Information Modeling (BIM) data. From the IFC/BIM, it could be extracted some pertinent contextual information, such as the structure of the environment, it's topology and the elements contained in it. Furthermore, a smart system deployed in a built environment requires an agent that

(1) will be able to represent the multiple and heterogeneous contextual information and that (2) will enable interoperability between different sources of information. As conclude in this chapter, an ontology agent fulfills both requirements. This is due to the inherent *semantic fusion* feature, which consists in integrating and organizing data and knowledge coming from multiple heterogeneous sources and to unify them into a consistent representation (i.e., enables interoperability). Furthermore, it was concluded that, an ontology representation of BIM is the only one to be *machine understandable* due to its formal semantic representation. Thus, it allows the inference of knowledge from existing information. This is an important feature in smart applications were machines should be able to automatically process and understand the information.

The creation of a context-aware system in the built environment is a complex task; it requires information from different domains such as environment data, sensing devices, spatio-temporal facts and details about the different events that may occur. Our research efforts to combine and integrate those heterogeneous data sources have led to the development of a semantic-based AI framework, called WiseNET. The main goals of WiseNET are to enhance what sensors "observe" by considering contextual information, and to provide a set of services to the building managers to ease their work by fusing pertinent information. In this work, we focused on visual sensors, however, the WiseNET system was designed in a generic manner, meaning that it does not depend on the type of environment, sensor or process procedure.

Chapter 4 presented the methodology followed to develop the core element of the semantic-based framework, the ontology. The WiseNET ontology is responsible for (1) describing the different kinds of information presented in the system and (2) enabling interoperability between them. The interoperability between heterogeneous sources of information is performed in a *semantic level*, i.e., a set of terms are defined which bridge the heterogeneous sources. Furthermore, one key step of the ontology development process, is to reuse external ontologies (already defined). This presents important advantages, such as saving time by not "reinventing the wheel" and using mature and proved ontological resources that have been validated by their applications and by the W3C. In addition, the ontology allows the inclusion of human-skill knowledge, in the form of semantic rules defined using a human understandable language. As concluded in this chapter, the resulting WiseNET ontology allows the description of information concerning an IVS context-aware system. This ontology model is human and machine understandable, and allows the inference of information relating the sensor network, the environment and the building usage.

Chapter 5, presented the different procedures developed to automatically populate data into the semantic model. Ontologies are normally used in static systems, were information is not changing over time. Therefore, we developed an innovative method to constantly insert data into an ontology, by using an Application Programming Interface (API) bridge between the data sources and the ontology. The API uses multiple semantic web technologies which enable the interaction with the ontology. Furthermore, the insertion procedures were divided in two types: static population and dynamic population.

The static population consisted in inserting into an ontology the knowledge that (normally) stays unchangeable, such as information of the environment and the calibration information of a sensor network. The proposed static population process automatically extracts pertinent information from an IFC/BIM file and insert it into the WiseNET ontology. The IFC/BIM was used as input due to it contains the required environment information and

they are becoming a standard in the architecture and construction communities. However, the process does not depend on the IFC/BIM, since it was designed in a modular and generic way. For example, if the IFC/BIM file is not available (which is the case for most of old buildings), the environment knowledge can still be inserted using our process by manually executing the population query.

The dynamic population consisted in inserting into the ontology the knowledge that should be inserted frequently (i.e., multiple times), such as sensor data. The dynamic population process automatically extracts structures and inserts the data observed by each camera node and insert it into the WiseNET ontology. The dynamic population process is divided into knowledge extraction and processing. The knowledge extraction consists in acquiring the image and performing a computer vision methods in it, to give some knowledge to (understand) what the camera is observing (e.g., people detection). The knowledge processing consists in representing the extracted knowledge using the vocabulary defined in the ontology, then inserting it according to the current and previous information in the system. The latter step is performed by following a domain dedicated process. Based on the IVS application, the process focus on creating people in spaces and tracking them while they move between spaces. Furthermore, the process uses the information if a detection was made around a door, to determine if a person is entering/leaving a space, and to ease the person re-identification between multiple cameras. Finally, regarding the privacy protection, is important to remark that the system does not send or save any image, just the knowledge of it is extracted. However, if required, the smart cameras used could record.

After having defined the semantic-based framework and the process to automatically populate it, Chapter 6 presented a dataset created for evaluating the system, as well as the different evaluations procedures. As shown in this chapter, existing multi-camera multi-space dataset does not include the complete information of the environment, thus, to evaluate our system a new dataset had to be created. To our knowledge, the WiseNET dataset is the first to provide a set videos along with the complete information of the environment. Moreover, the WiseNET dataset is a contribution to the computer vision community, due to its large panel of applications, such as, people detection, re-identification, tracking, human-action recognition, office object detections, etc.

Two types of evaluations were performed, one qualitative that focus on answer some questions of interest, and the second one a quantitative which focus on the performance of our semantic-based system in a computer vision task. The first evaluation consisted in verifying if the system satisfy its intent. This was done by checking if it is able to answer some Competency Questions (CQs) concerning the built environment, the sensors deployed in it and the building usage. As concluded in this chapter, the system is able to correctly answered all the CQs. Furthermore, the proposed formalism for defining the CQs allow to easily interrogate a smart building and to monitor what is happening in real-time. Moreover, the semantic-based system enables the smart building to understand what is happening inside itself *without* any human interaction. One important example is the deduction that somebody is still present in a space even if nobody is detected by any camera (person occluded). This type of deduction shows how our semantic-based system overcomes a big limitation of computer vision systems—they only know what they observe—by using contextual information and common sense rules. Furthermore, a monitor unit interfaces is presented, which agglomerates a set of services to ease some tasks of building managers. The services are based on the CQs which results are presented in an graphical and intuitive manner. The propose services can help building managers

to: interrogate the system about information of the building structure and the elements contained in it; to ease the building monitoring; to visualizing in real-time what the flow of people; and to protecting the private life of building users by not showing any image.

The second evaluation consisted in checking the performance of the semantic-base system in tracking people. The evaluation was done using identity-based metrics, because we considered that is the most appropriate for people tracking application. In general, the results showed that the WiseNET system manage to successfully track people while they moved through different spaces, specially in standard indoor scenarios (i.e., people seated and people moving, alone or in groups of two). However, there were some uncommon scenarios that were not considered during the system design, thus, resulting in some incorrect results. Specifically, scenarios such as, high number of people moving together (14-15 people), multiple and constant occlusions due to people clutter, people going out from blind spaces. Thus, it can be concluded that, the advantage of the semantic-based AI system is that it allows to easily pinpoint the sources of errors and, if required, modify the system accordingly. However, the disadvantage is that all the situations that needs to be considered, they have to be "manually" inserted into the system, they cannot be generalized as in non-symbolic AI systems.

From the previous evaluations, it can be concluded that, the semantic combination of data coming from a smart camera network with the data of the environment, allows a smart building to reason by itself, by using a formalism easy to understand by humans. Making it a "real" smart building. Moreover, the addition of contextual information to what the cameras observe, allows to overcome computer vision problems, specifically the occlusion problem that occurs when a persons goes outside the field of view of a camera. Furthermore, the proposed semantic-based framework allows to dialog with the smart building by executing queries.

The previous evaluations validate the proof of concept of our semantic-base system. However, we decided to go further by evaluating the influence of using real people detectors, obtained from non-symbolic AI. This type of evaluation allow us to observe some limits of our system. The results of this evaluation showed that the performance of the system depends on the quality of people detections, however, even with a low quality detector the system was able, to some extend, to correctly answer some CQs and to track people. Moreover, further evaluations and test should be performed to clearly determine the influence of the detection process in the semantic-based framework, and how this influence could be decreased. However, this is a first approach of combining both types of AIs—non-symbolic and symbolic—in a distributed way. Thus, it can be considered as a distributed intelligence system, where the non-symbolic AI is used to do what it does the best, which is detection, while the symbolic AI (which is based on human knowledge) is used in a higher level, to control what is happening in the system.

Many different adaptations, test, and experiments have been left as future work. For example, experiments could be performed considering different types of sensors (e.g., temperature, motion) and actuators, in this way, the reasoning of the smart building could influence the environment. In the same line, some test could be be performed where the using the system to re-configure the smart camera process according to different situations. Furthermore, a deeper analysis could be performed to concerning the influence of people detection, specifically, different parameters could be tested, such as different visual descriptors (e.g., SIFT, SURF), number of features, different metrics (e.g., cosine distance, chebyshev distance), and the use of feature improvements (e.g., background

suppression, stripping). Also, to determine which is the most challenging camera in the dataset an statistical study should be performed, this could be interesting for understanding the problems of camera installations. Furthermore, a qualitative analysis could be done on the advantages of using the monitor unit. In addition, the system could be tested in real-time, without using recorded videos, to test the system in a real scenario. Finally, deeper analysis could be performed to in the axis of combining both non-symbolic and symbolic AI in a smart building application.

Finally, the presented work opens the doors of a plethora of applications, specially in the smart building domain. For example, the presented CQs could be used for: *maintenance scheduling*, where is important to know the usage of the different building elements (e.g., spaces, doors); *smart light control*, where the building will automatically turn on/off light according to the person's location in a space, not only if a person is moving (like most current "smart" light systems); *people counting*, where building could control if the number of people is appropriate to the space limitations; *path guidance*, the system could control if a user takes the correct path and reaches his final destination; and *fire evacuation*, in the case of a fire, the system could guide the people to different exits, as well as, the heatmap could be used to firemen to know the number of people and their location in the building. In addition, accurately knowing the occupancy of the spaces and their usage, could be very useful in optimizing building operations, such as energy saving policies.

8

AUTHOR'S PUBLICATIONS

INTERNATIONAL JOURNALS

[a] **R. Marroquin**, J. Dubois, and C. Nicolle. WiseNET: An indoor multi-camera multi-space dataset with contextual information and annotations for people detection and tracking. Data in Brief, 104654, oct 2019.

[b] **R. Marroquin**, J. Dubois, and C. Nicolle. Ontology for a Panoptes building: Exploiting contextual information and a smart camera network. Semantic Web, 9(6):803–828, sep 2018.

INTERNATIONAL CONFERENCES AND WORKSHOPS

[i] **R. Marroquin**, J. Dubois, and C. Nicolle. Know beyond seeing : combining computer vision with semantic reasoning. In Proceedings of the12th IEEE International Conference on Semantic Computing (ICSC), pages 310–311, 2018.

[ii] **R. Marroquin**, J. Dubois, and C. Nicolle. Multiple Ontology Binding in a Smart Building Environment. In Proceedings of the Workshop on Linked Data in Architecture and Construction (LDAC), 2017.

[iii] **R. Marroquin**, J. Dubois, and C. Nicolle. PhD forum: WiseNET - Smart camera network interacting with a semantic model. In Proceedings of the 10th International Conference on Distributed Smart Cameras (ICDSC), pages 224–225, 2016. *(Best paper price)*

[iv] **R. Marroquin**, J. Dubois, and C. Nicolle. WiseNET: smart camera network combined with ontological reasoning for smart building management. In 5th Workshop on Architecture of Smart Cameras (WASC), 2016.

[v] J. Dubois, A. Moinet, S. Bobbia, **R. Marroquin**, B. Heyrman, P. Bonazza, B. Darties, C. Nicolle, Y. Benezeth, J. Mitéran, D. Ginhacet. WiseEye: A Platform to Manage and Experiment on Smart Camera Networks. In 5th Workshop on Architecture of Smart Cameras (WASC), 2016.

National conferences and workshops

[I] **R. Marroquin**, J. Dubois, and C. Nicolle. Savoir au dela de voir: vision artificielle et raisonnement logique. In Proceedings of Conference sur l'Extraction et la Gestion de Connaissances (EGC), pages 385–386, 2018.

[II] **R. Marroquin**, J. Dubois, and C. Nicolle. Supervision et respects de la vie privée, l'enjeu ethique des interfaces de visualisation. In Atelier VIF: Visualisation d'informations, Interaction, et Fouille de données - In Conference sur l'Extraction et la Gestion de Connaissances (EGC), pages 385–386, 2018.

Appendices

A

WISENET ONTOLOGY SPECIFICATION

WiseNET ontology general information

Release date:
2019-03-20
This version:
http://ontology.wisenet.checksem.fr/
Previous version:
3.1
Revision:
3.2
Authors:
Roberto Marroquin (mailto:robertomarrok@gmail.com)

Abstract

The Wised-NETwork (WiseNET) ontology provides a formal model that aggregates, analysis and re-purposing the information coming from a network of smart cameras, deployed in a built environment. The WiseNET ontology incorporates a vast corpus of concepts in the domain of an intelligent video surveillance systems. The main functions of the WiseNET ontology are to enable interoperability between the heterogeneous data and to deduce implicit facts from the explicit ones. Thus, allowing to answer queries about the building usage and to perform real-time event/anomaly detection.

For more information and a better visualization of the WiseNET ontology we refer readers to the ontology **http://ontology.wisenet.checksem.fr/** .

WiseNET: Overview

This ontology has the following classes and properties.

Classes

Agent	Alarm	Bounding box	Building	Building element
Detection	Door	Event	Field of view	Image processing algorithm
Instant event	Interval event	Person	Person detection	Person in space
Procedure	Region of interest	Sensor	Smart camera	Space

Spatial thing	Storey	Temporal entity	Time duration	Time instant
Time interval	Zone			

Object Properties

adjacent zone	after	agent	agent in	appears in
before	contains element	contains person	contains zone	factor
factor of	has alarm	has beginning	has bounding box	has duration
has element	has end	has FOV	has nearby sensor	has space
has storey	has time	hosts	implemented by	implements
in region of interest	is FOV of	is hosted by	is observed by	is related to
is sub event of	made by sensor	observes	overlaps	person location
place	represents	shows	sub-event	time

Data Properties

dimension	in XSD Date-Time-Stamp	ip address	is entry violation	is event open
is intruder	is noise	is occluded	is occupied	is restricted
max capacity	number of people	numeric value of temporal duration	start recording	triggered by intruder
triggered by maxCapacity	visual descriptors	xywh		

Cross reference for WiseNET classes, properties and dataproperties

This section provides details for each class and property defined by WiseNET. Notice that the definitions of the external terms were taken directly from the original ontologies.

Legend
[c]: Classes
[op]: Object Properties
[dp]: Data Properties
[ni]: Named Individuals

Classes

Agent[c]

IRI: http://xmlns.com/foaf/0.1/Agent

The Agent class is the class of agents; things that do stuff. A well known sub-class is Person, representing people. Other kinds of agents include Organization and Group.

has sub-classes
Person [c]
is in domain of
agent in [op]
is in range of
agent [op]

Alarm[c]

IRI: http://ontology.wisenet.checksem.fr#Alarm

Signal used to aware of the presence of an undesired situation.

has super-classes
has time [op] some Temporal entity [c]
is in domain of
triggered by intruder [dp], triggered by maxCapacity [dp]
is in range of
has alarm [op]

Bounding box[c]

IRI: http://ontology.wisenet.checksem.fr#BoundingBox

Rectangular box, defined by the coordinates of the top left point, its width and its height.

It can be used to defined a region of interest, a detection, etc.

has sub-classes
Region of interest [c]
is in domain of
xywh [dp]
is in range of
has bounding box [op]

Building[c]

IRI: https://w3id.org/bot#Building

An independent unit of the built environment with a characteristic spatial structure, intended to serve at least one function or user activity [ISO 12006-2:2013].

has super-classes
Zone [c]
is disjoint with
Space [c], Storey [c]

Building element[c]

IRI: https://w3id.org/bot#Element

Constituent of a construction entity with a characteristic technical function, form or position [12006-2, 3.4.7].

has sub-classes
Door [c], Sensor [c]
is in range of
contains element [op], has element [op], represents [op]
is disjoint with
Zone [c]

Detection[c]

IRI: http://ontology.wisenet.checksem.fr#Detection

An event identifiying the presence of something in a specific point in the time/space.

has super-classes
Instant event [c]
has bounding box [op] some Bounding Box [c]
in region of interest [op] some Region of interest [c]
is in domain of
is entry violation. [dp]

Door[c]

IRI: http://ontology.wisenet.checksem.fr#Door

A hinged, sliding, or revolving barrier at the entrance to a space.

has super-classes
Building element [c]

Event[c]

IRI: http://purl.org/NET/c4dm/event.owl#Event

An arbitrary classification of a space/time region, by a cognitive agent. An event may have actively participating agents, passive factors, products, and a location in space/time.

is equivalent to
Instant event [c] or Interval event [c]
has sub-classes
Instant event [c], Interval event [c]
is in domain of
agent [op], factor [op], is noise [dp], is sub event of [op], made by sensor [op], place [op], sub-event [op], time [op]
is in range of
agent in [op], factor of [op], is sub event of [op], sub-event [op]

Field of view[c]

IRI: http://ontology.wisenet.checksem.fr#FieldOfView

Observable area of the world. A field of view may shows different elements of the world such as people, cars, doors, etc.

has super-classes
    shows [op] some Region of interest [c]
    shows [op] some Person [c]
is in domain of
    is FOV of [op], overlaps [op], shows [op]
is in range of
    appears in [op], has FOV [op], overlaps [op]

Image processing algorithm[c]

IRI: http://ontology.wisenet.checksem.fr#ImageAlgorithm

Algorithms used to process images.

has super-classes
    Procedure [c]
has sub-classes
    Person detection [c]

Instant event[c]

IRI: http://ontology.wisenet.checksem.fr#InstantEvent

An event that occurs at a precise instant in time.

has super-classes
    Event [c]
    is sub event of [op] only Interval event [c]
    time [op] only Time instant [c]
has sub-classes
    Detection [c]

Interval event[c]

IRI: http://ontology.wisenet.checksem.fr#IntervalEvent

An event that occurs in a time interval.

has super-classes
    Event [c]
    sub-event [op] only Instant event [c]
    time [op] only Time interval [c]
has sub-classes
    Person in space [c]
is in domain of
    is related to [op]
is in range of
    is related to [op]

Person[c]

IRI: http://xmlns.com/foaf/0.1/Person

The Person class represents people. Something is a Person if it is a person. We don't nitpic about whether they're alive, dead, real, or imaginary. The Person class is a sub-class of the Agent class, since all people are considered 'agents' in FOAF.

has super-classes
    Agent [c]
is in domain of
    is intruder [dp], is occluded [dp], person location [op], visual descriptors [dp]
is in range of
    contains person [op]

Person detection[c]

IRI: http://ontology.wisenet.checksem.fr#PersonDetection

Image processing algorithms that focus on localizing and classifying people in images. Person detection is a particular case of object detection algorithms.

has super-classes
    Image processing algorithm [c]
has members
    ground truth [ni], HOG_SVM [ni], SSD [ni], YOLOv3 [ni]

Person in space[c]

IRI: http://ontology.wisenet.checksem.fr#PersonInSpace

Container of detections relating a specific person with a specific space during a period of time.

has super-classes
- Interval event [c]

is in domain of
- is event open [dp]

Procedure[c]

IRI: http://www.w3.org/ns/sosa/Procedure

A workflow, protocol, plan, algorithm, or computational method specifying how to make an Observation, create a Sample, or make a change to the state of the world (via an Actuator). A Procedure is re-usable, and might be involved in many Observations, Samplings, or Actuations. It explains the steps to be carried out to arrive at reproducible results.

Example
- `The measured wind speed differs depending on the height of the sensor above the surface, e.g., due to friction. Consequently, procedures for measuring wind speed define a standard height for anemometers above ground, typically 10m for meteorological measures and 2m in Agrometeorology. This definition of height, sensor placement, and so forth are defined by the Procedure.`

has sub-classes
- Image processing algorithm [c]

is in domain of
- implemented by [op]

is in range of
- implements [op]

Region of interest[c]

IRI: http://ontology.wisenet.checksem.fr#RegionOfInterest

Abstract region of interest (ROI), defined by a bounding box. Normally, there are some ROIs in the field of view of a camera.

has super-classes
- Bounding Box [c]

is in domain of
- represents [op]

is in range of
- in region of interest [op]

Sensor[c]

IRI: http://www.w3.org/ns/sosa/Sensor

Device, agent (including humans), or software (simulation) involved in, or implementing, a Procedure. Sensors respond to a stimulus, e.g., a change in the environment, or input data composed from the results of prior Observations, and generate a Result. Sensors can be hosted by Platforms.

Example

```
Accelerometers, gyroscopes, barometers, magnetometers, and so forth are
Sensors that are typically mounted on a modern smart phone (which acts as
Platform). Other examples of sensors include the human eyes.
```

has super-classes
Building element [c]
has sub-classes
Smart camera [c]
is in domain of
has nearby sensor [op], implements [op], ip address [dp], is hosted by [op]
is in range of
has nearby sensor [op], hosts [op], implemented by [op]

Smart camera[c]

IRI: http://ontology.wisenet.checksem.fr#SmartCamera

Sensors which are capable of acquiring visual information and filtering/extracting the pertinent information of the scene by implementing image processing algorithms.

A smart camera is a self-contained vision systems.

has super-classes
Sensor [c]
implements [op] some Image processing algorithm [c]
is in domain of
has FOV [op], observes [op], start recording [dp]
is in range of
is FOV of [op], is observed by [op], made by sensor [op]

Space[c]

IRI: https://w3id.org/bot#Space

A limited three-dimensional extent defined physically or notionally [ISO 12006-2 (DIS 2013), 3.4.3].

has super-classes
Zone [c]
is in domain of
contains element [op], hosts [op]
is in range of
has space [op], is hosted by [op]
is disjoint with
Building [c], Storey [c]

Spatial thing[c]

IRI: http://www.w3.org/2003/01/geo/wgs84_pos#SpatialThing

Anything with spatial extent, i.e. size, shape, or position. e.g. people, places, bowling balls, as well as abstract areas like cubes.

has sub-classes
Zone [c]
is in range of
place [op]

Storey[c]

IRI: https://w3id.org/bot#Storey

A level part of a building.

has super-classes
Zone [c]
is in range of
has storey [op]
is disjoint with
Building [c], Space [c]

Temporal entity[c]

IRI: http://www.w3.org/2006/time#TemporalEntity

A temporal interval or instant.

is equivalent to
    Time instant [c] or Time interval [c]
has sub-classes
    Time instant [c], Time interval [c]
is in domain of
    after [op], before [op], has beginning [op], has end [op]
is in range of
    after [op], before [op], has time [op], time [op]

Time duration[c]

IRI: http://www.w3.org/2006/time#Duration

Duration of a temporal extent expressed as a number scaled by a temporal unit.

is in domain of
    numeric value of temporal duration [dp]
is in range of
    has duration [op]

Time instant[c]

IRI: http://www.w3.org/2006/time#Instant

A temporal entity with zero extent or duration.

has super-classes
    Temporal entity [c]
is in domain of
    in XSD Date-Time-Stamp [dp]
is in range of
    has beginning [op], has end [op]

Time interval[c]

IRI: http://www.w3.org/2006/time#Interval

A temporal entity with an extent or duration.

has super-classes
Temporal entity [c]
is in domain of
has duration [op]

Zone[c]

IRI: https://w3id.org/bot#Zone

A spatial 3D division. Sub-classes of bot:Zones include bot:Site, bot:Building, bot:Storey, or bot:Space. An instance of bot:Zone can contain other bot:Zone instances, making it possible to group or subdivide zones. An instance of bot:Zone can be adjacent to other bot:Zone instances.

has super-classes
Spatial Thing [c]
has sub-classes
Building [c], Space [c], Storey [c]
is in domain of
adjacent zone [op], contains person [op], contains zone [op], has alarm [op], has element [op], has space [op], has storey [op], is occupied [dp], is restricted [dp], max capacity [dp], number of people [dp]
is in range of
adjacent zone [op], contains zone [op], person location [op]
is disjoint with
Building element [c]

Object Properties

adjacent zone[op]

IRI: https://w3id.org/bot#adjacentZone

Relationship between two zones that share a common interface, but do not intersect.

has characteristics: symmetric

has domain
Zone [c]
has range
Zone [c]

after[op]

IRI: http://www.w3.org/2006/time#after

Gives directionality to time. If a temporal entity T1 is after another temporal entity T2, then the beginning of T1 is after the end of T2.

has domain
    Temporal entity [c]
has range
    Temporal entity [c]
is inverse of
    before [op]

agent[op]

IRI: http://purl.org/NET/c4dm/event.owl#agent

Relates an event to an active agent (a person, a computer, ... :-)).

has domain
    Event [c]
has range
    Agent [c]
is inverse of
    agent in [op]

agent in[op]

IRI: http://purl.org/NET/c4dm/event.owl#agent_in

has domain
    Agent [c]
has range
    Event [c]
is inverse of
    agent [op]

appears in[op]

IRI: http://ontology.wisenet.checksem.fr#appearsIn

Property that defines that something is observed by a visual sensor.

Example
```
A door may appear in a camera's field of view.
```
has range
Field of view [c]
is inverse of
shows [op]

before[op]

IRI: http://www.w3.org/2006/time#before

Gives directionality to time. If a temporal entity T1 is before another temporal entity T2, then the end of T1 is before the beginning of T2. Thus, "before" can be considered to be basic to instants and derived for intervals.

has characteristics: transitive

has domain
Temporal entity [c]
has range
Temporal entity [c]
is inverse of
after [op]

contains element[op]

IRI: https://w3id.org/bot#containsElement

Relation to a building element contained in a zone.

has super-properties
has element [op]
has sub-properties
hosts [op]
has domain
Space [c]
has range
Building element [c]

contains person^{op}

IRI: http://ontology.wisenet.checksem.fr#containsPerson

Relation to people contained in a zone.

has domain
    Zone ^{c}
has range
    Person ^{c}
is inverse of
    person location ^{op}
has sub-property chains
    contains zone ^{op} o contains person ^{op}

contains zone^{op}

IRI: https://w3id.org/bot#containsZone

Relationship to the subzones of a major zone. A space zone could for instance be contained in a storey zone which is further contained in a building zone. bot:containsZone is a transitive property meaning that in the previous example the space zone would also be contained in the building zone.

has characteristics: transitive

has sub-properties
    has space ^{op}, has storey ^{op}
has domain
    Zone ^{c}
has range
    Zone ^{c}

factor^{op}

IRI: http://purl.org/NET/c4dm/event.owl#factor

Relates an event to a passive factor (a tool, an instrument, an abstract cause...).

has domain
    Event ^{c}
is inverse of
    factor of ^{op}

factor of[op]

IRI: http://purl.org/NET/c4dm/event.owl#factor_of

has range
    Event [c]
is inverse of
    factor [op]

has alarm[op]

IRI: http://ontology.wisenet.checksem.fr#hasAlarm

Relation to an alarm and the zone it covers.

has domain
    Zone [c]
has range
    Alarm [c]

has beginning[op]

IRI: http://www.w3.org/2006/time#hasBeginning

Beginning of a temporal entity

has domain
    Temporal entity [c]
has range
    Time instant [c]

has bounding box[op]

IRI: http://ontology.wisenet.checksem.fr#hasBoundingBox

Relation to a bounding box object.

Example
    `A detection was made in an area defined by a bounding box.`
has range
    Bounding Box [c]

has duration[op]

IRI: http://www.w3.org/2006/time#hasDuration

Duration of a temporal entity, expressed as a scaled value or nominal value.

has domain
    Time interval [c]
has range
    Time duration [c]

has element[op]

IRI: https://w3id.org/bot#hasElement

Links a Zone to an Element that is either contained in or adjacent to, the Zone. The intended use of this relationship is not to be stated explicitly, but to be inferred from its sub-properties. It will, for example, allow one to query for all the doors of a building given that they have an adjacency to spaces of the building.

has sub-properties
    contains element [op]
has domain
    Zone [c]
has range
    Building element [c]
has sub-property chains
    contains zone [op] o has element [op]

has end[op]

IRI: http://www.w3.org/2006/time#hasEnd

End of a temporal entity.

has domain
    Temporal entity [c]
has range
    Time instant [c]

has FOV[op]

IRI: http://ontology.wisenet.checksem.fr#hasFieldOfView

Relation between a visual sensor and its field of view.

has domain
    Smart camera [c]
has range
    Field of view [c]
is inverse of
    is FOV of [op]

has nearby sensor[op]

IRI: http://ontology.wisenet.checksem.fr#hasNearbySensor

Relation between two sensors. Two sensors are considered nearby if: 1) they are located at the same space, 2) if their FOV overlaps (for cameras).

has characteristics: symmetric

has domain
    Sensor [c]
has range
    Sensor [c]

has space[op]

IRI: https://w3id.org/bot#hasSpace

Relation to spaces contained in a zone. The typical domains of bot:hasSpace are instances of bot:Storey and bot:Building.

has super-properties
    contains zone [op]
has domain
    Zone [c]
has range
    Space [c]

has storey[op]

IRI: https://w3id.org/bot#hasStorey

Relation to storeys contained in a zone. The typical domains of bot:hasStorey are instances of bot:Building.

has super-properties
    contains zone [op]
has domain
    Zone [c]
has range
    Storey [c]

has time[op]

IRI: http://www.w3.org/2006/time#hasTime

Supports the association of a temporal entity (instant or interval) to anything.

has sub-properties
    time [op]
has range
    Temporal entity [c]

hosts[op]

IRI: http://www.w3.org/ns/sosa/hosts

Relation between a Platform and a Sensor, Actuator, Sampler, or Platform, hosted or mounted on it.

has super-properties
    contains element [op]
has domain
    Space [c]
has range
    Sensor [c]
is inverse of
    is hosted by [op]

implemented by[op]

IRI: http://www.w3.org/ns/ssn/implementedBy

Relation between a Procedure (an algorithm, procedure or method) and an entity that implements that Procedure in some executable way.

Example
`For example, the relationship between a scientific measuring Procedure and a sensor that senses via that Procedure.`

has domain
Procedure [c]

has range
Sensor [c]

is inverse of
implements [op]

implements[op]

IRI: http://www.w3.org/ns/ssn/implements

Relation between an entity that implements a Procedure in some executable way and the Procedure (an algorithm, procedure or method).

Example
`For example, the relationship between a sensor and the scientific measuring Procedure via which it senses.`

has domain
Sensor [c]

has range
Procedure [c]

is inverse of
implemented by [op]

in region of interest[op]

IRI: http://ontology.wisenet.checksem.fr#inRegionOfInterest

Relation between something that happened or is located around an area of interest.

has range
Region of interest [c]

is FOV of[op]

IRI: http://ontology.wisenet.checksem.fr#isFieldOfViewOf

Relation between a field of view and a visual sensor.

has domain
    Field of view [c]
has range
    Smart camera [c]
is inverse of
    has FOV [op]

is hosted by[op]

IRI: http://www.w3.org/ns/sosa/isHostedBy

Relation between a Sensor, or Actuator, Sampler, or Platform, and the Platform that it is mounted on or hosted by.

has domain
    Sensor [c]
has range
    Space [c]
is inverse of
    hosts [op]

is observed by[op]

IRI: http://www.w3.org/ns/sosa/isObservedBy

Relation between an ObservableProperty and the Sensor able to observe it.

has range
    Smart camera [c]
is inverse of
    observes [op]

is related to[op]

IRI: http://ontology.wisenet.checksem.fr#isRelatedTo

Relation between two interval events.

Example
```
Two person in space events are consider related if they involve the same agent/person.
```

has characteristics: symmetric, transitive

has domain
Interval event [c]
has range
Interval event [c]

is sub event of[op]

IRI: http://ontology.wisenet.checksem.fr#isSubEventOf

Relation between a simple event and a complex one that contains the simple one.

Example
```
A detection event is a sub event of a person in space event.
```
has domain
Event [c]
has range
Event [c]
is inverse of
sub-event [op]

made by sensor[op]

IRI: http://www.w3.org/ns/sosa/madeBySensor

Example
```
Relation between an Observation and the Sensor which made the Observations.
```
has domain
Event [c]
has range
Smart camera [c]

observes[op]

IRI: http://www.w3.org/ns/sosa/observes

Relation between a Sensor and an ObservableProperty that it is capable of sensing.

Example
`A smart camera observes a door and a person`
has domain
Smart camera [c]
is inverse of
is observed by [op]
has sub-property chains
has FOV [op] o shows [op]

overlaps[op]

IRI: http://ontology.wisenet.checksem.fr#overlaps

Relation between two field of views. They are consider overlapping if they observe the same object at the same time, e.g., a building element like a door.

has characteristics: symmetric

has domain
Field of view [c]
has range
Field of view [c]

person location[op]

IRI: http://ontology.wisenet.checksem.fr#personLocation

Location of a person in a Zone.

has domain
Person [c]
has range
Zone [c]
is inverse of
contains person [op]

place[op]

IRI: http://purl.org/NET/c4dm/event.owl#place

Relates an event to a spatial object.

has domain
    Event [c]
has range
    Spatial Thing [c]

represents[op]

IRI: http://ontology.wisenet.checksem.fr#represents

Property to state the real (physical) entity of an abstract area.

Example
```
A region of interest in an image represents a door in the real world.
```
has domain
    Region of interest [c]
has range
    Building element [c]

shows[op]

IRI: http://ontology.wisenet.checksem.fr#shows

Property to state that a visual sensor is observing something.

Example
```
A camera's filed of view shows a person.
```
has domain
    Field of view [c]
is inverse of
    appears in [op]
has sub-property chains
    shows [op] o represents [op]

sub-event[op]

IRI: http://purl.org/NET/c4dm/event.owl#sub_event

This property provides a way to split a complex event (for example, a performance involving several musicians) into simpler ones (one event per musician).

has domain
    Event [c]
has range
    Event [c]
is inverse of
    is sub event of [op]

time[op]

IRI: http://purl.org/NET/c4dm/event.owl#time

Relates an event to a time object, classifying a time region (either instantaneous or having an extent). By using the Timeline ontology here, you can define event happening on a recorded track or on any media with a temporal extent.

has super-properties
    has time [op]
has domain
    Event [c]
has range
    Temporal entity [c]

Data Properties

dimension[dp]

IRI: http://ontology.wisenet.checksem.fr#dimension

Dimension of objects in meters.

Example
```
A door has dimensions "1.5x2.075" m.
```
has range
    string

in XSD Date-Time-Stamp[dp]

IRI: http://www.w3.org/2006/time#inXSDDateTimeStamp

Position of an instant, expressed using xsd:dateTimeStamp.

has domain
    Time instant [c]
has range
    date time stamp

ip address[dp]

IRI: http://ontology.wisenet.checksem.fr#ipAddress

IP address of a sensor.

has domain
    Sensor [c]
has range
    string

is entry violation[dp]

IRI: http://ontology.wisenet.checksem.fr#isEntryViolation

Property to state if person detection is considered an entry violation.

Example
```
If a person detection is made in a restricted area.
```
has domain
    Detection [c]
has range
    boolean

is event open[dp]

IRI: http://ontology.wisenet.checksem.fr#isEventOpen

Property to define if a person in space event is open. If it is open then detections can be attached to it.

has domain
    Person in space [c]
has range
    boolean

is intruder[dp]

IRI: http://ontology.wisenet.checksem.fr#isIntruder

Property to define if a person is considered as an intruder.

Example
```
If it was detected in a restricted zone.
```
has domain
Person [c]
has range
boolean

is noise[dp]

IRI: http://ontology.wisenet.checksem.fr#isNoise

Property to state if an event is noise, i.e., if it was considered as true when it should not.

Example
```
Events generated after false detections.
```
has domain
Event [c]
has range
boolean

is occluded[dp]

IRI: http://ontology.wisenet.checksem.fr#isOccluded

Property to state if a person is occluded. A Person is occluded if it is located in a space but is not observed by any camera.

has domain
Person [c]
has range
boolean

is occupied[dp]

IRI: http://ontology.wisenet.checksem.fr#isOccupied

Property to state if there is a person in a Zone. If a Zone is not occupied, then it means is empty.

has domain
    Zone [c]
has range
    boolean

is restricted[dp]

IRI: http://ontology.wisenet.checksem.fr#isRestricted

Property to define if a zone has restricted access.

has domain
    Zone [c]
has range
    boolean

max capacity[dp]

IRI: http://ontology.wisenet.checksem.fr#maxCapacity

Property to define the maximal people capacity of a zone.

Example
```
The PhD room has a maximal capacity of 20 people.
```
has domain
    Zone [c]
has range
    int

number of people[dp]

IRI: http://ontology.wisenet.checksem.fr#numberOfPeople

Number of people located in a Zone.

has domain
    Zone [c]
has range
    int

numeric value of temporal duration[dp]

IRI: http://www.w3.org/2006/time#numericDuration

Value of a temporal extent expressed as a decimal number scaled by a temporal unit.

has domain
    Time duration [c]
has range
    decimal

start recording[dp]

IRI: http://ontology.wisenet.checksem.fr#startRecording

Property to state if a visual sensor should record.

Example
```
If there is an intruder alarm triggered, then start recording should be
true.
```
has domain
    Smart camera [c]
has range
    boolean

triggered by intruder[dp]

IRI: http://ontology.wisenet.checksem.fr#triggeredByIntruder

Property to state if an alarm has been triggered due to an intrusion.

has domain
    Alarm [c]
has range
    boolean

triggered by maxCapacity[dp]

IRI: http://ontology.wisenet.checksem.fr#triggeredByMaxCapacity

Property to state if an alarm has been triggered due to the maximal capacity of a zone has been reached.

has domain
    Alarm [c]
has range
    boolean

visual descriptors[dp]

IRI: http://ontology.wisenet.checksem.fr#visualDescriptors

Array of visual features used for describing a person.

has domain
    Person [c]
has range
    string

xywh[dp]

IRI: http://ontology.wisenet.checksem.fr#xywh

Vector that defines the coordinates of a bounding box: ' x,y ' are the coordinates of the top left point, while ' w,h ' are the width and the height of the bounding box.

has domain
    Bounding Box [c]
has range
    gstring

BIBLIOGRAPHY

[1] S. Abiteboul, I. Manolescu, P. Rigaux, M.-C. Rousset, and P. Senellart. *Web Data Management*, volume 28. Cambridge university press, 2012.

[2] H. Arasteh, V. Hosseinnezhad, V. Loia, A. Tommasetti, O. Troisi, M. Shafie-Khah, and P. Siano. Iot-based smart cities: a survey. In *Proceedings of the 16th International Conference on Environment and Electrical Engineering (EEEIC)*, pages 1–6. IEEE, 2016.

[3] S. Arora and B. Barak. *Computational complexity : a modern approach*. Cambridge University Press, 2009.

[4] R. Atkinson, R. García-Castro, J. Lieberman, and C. Stadler. Semantic Sensor Network Ontology. https://www.w3.org/TR/vocab-ssn/, 2017.

[5] AutoDesk Inc. AutoCAD DXF file format documentation. https://www.autodesk.com/techpubs/autocad/acad2000/dxf/index.htm, 2000.

[6] AutoDesk Inc. Revit | BIM Software. https://www.autodesk.com/products/revit/overview, 2019.

[7] F. Baader, I. Horrocks, C. Lutz, and U. Sattler. *An Introduction to Description Logic*. Cambridge University Press, Cambridge, 2017.

[8] A. Bar-Hillel, D. Levi, E. Krupka, and C. Goldberg. Part-based feature synthesis for human detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 127–142, 2010.

[9] J. Beetz, J. Van Leeuwen, and B. De Vries. IfcOWL: A case of transforming EXPRESS schemas into ontologies. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM)*, 23(1):89–101, 2009.

[10] R. Benenson, M. Omran, J. Hosang, and B. Schiele. Ten years of pedestrian detection, what have we learned? In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 613–627. Springer, 2014.

[11] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: The CLEAR MOT metrics. *Eurasip Journal on Image and Video Processing*, 2008:1–10, 2008.

[12] T. Berners-lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001.

[13] A. Bialkowski, S. Denman, S. Sridharan, C. Fookes, and P. Lucey. A database for person re-identification in multi-camera surveillance networks. In *2012 International Conference on Digital Image Computing Techniques and Applications, DICTA 2012*, 2012.

[14] P. Bonazza, J. Mitéran, D. Ginhac, and J. Dubois. Machine Learning VS Transfer Learning Smart Camera Implementation for Face Authentication. In *Proceedings of the 12th International Conference on Distributed Smart Cameras (ICDSC)*, pages 1–2. ACM Press, 2018.

[15] A. Borgida. On the Relative Expressiveness of Description Logics and Predicate Logics. *Artificial Intelligence*, 82(1-2):353–367, apr 1996.

[16] A. Borgida. On Importing Knowledge from Ontologies. In *Modular Ontologies*, chapter 4, pages 91–112. Springer, 2009.

[17] G. R. Bradski and A. Kaehler. *Learning OpenCV - computer vision with the OpenCV library: software that sees.* O'Reilly, 2008.

[18] M. Bramberger, A. Doblander, A. Maier, B. Rinner, and H. Schwabach. Distributed embedded smart cameras for surveillance applications. *Computer*, 39(2):68–75, feb 2006.

[19] D. Brickley and R. V. Guha. RDF Schema 1.1. https://www.w3.org/TR/rdf-schema/, 2014.

[20] D. Brickley and L. Miller. FOAF Vocabulary Specification. http://xmlns.com/foaf/spec, 2014.

[21] BuildingSMART. IFC Releases. http://www.buildingsmart-tech.org/specifications/ifc-releases/summary.

[22] A. Burbano, S. Bouaziz, and M. Vasiliu. 3d-sensing distributed embedded system for people tracking and counting. In *Proceedings of the International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 470–475. IEEE, 2015.

[23] C. J. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[24] A. A. Butt and R. T. Collins. Multi-target Tracking by Lagrangian Relaxation to Min-Cost Network Flow. In *Proceedings of the 26th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1846–1853, 2013.

[25] S. M. Cahn. *Classics of Western philosophy*. Hackett Pub. Co, 2012.

[26] Y. Cai and G. Medioni. Exploring context information for inter-camera multiple target tracking. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 761–768. IEEE, mar 2014.

[27] S. Castano, A. Ferrara, and S. Montanelli. Ontology-based Interoperability Services for Semantic Collaboration in Open Networked Systems. In *Interoperability of Enterprise Software and Applications*, pages 135–146. Springer-Verlag, London, 2006.

[28] S.-H. Cha. Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, 1(4):300–307, 2007.

[29] W. Chaochaisit, M. Bessho, N. Koshizuka, and K. Sakamura. Human Localization Sensor Ontology: Enabling OWL 2 DL-Based Search for User's Location-Aware Sensors in the IoT. In *Proceedings of the 10th IEEE International Conference on Semantic Computing (ICSC)*, pages 107–111, 2016.

[30] C. H. Chen. *Handbook of Pattern Recognition and Computer Vision*. World scientific, feb 2016.

[31] W. Chen, L. Cao, X. Chen, and K. Huang. An equalised global graphical model-based approach for multi-camera object tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(1):2367–2381, 2017.

[32] X. Chen and B. Bhanu. Integrating Social Grouping for Multitarget Tracking Across Cameras in a CRF Model. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(11):2382–2394, nov 2017.

[33] M. Chouchene, F. E. Sayadi, J. Miteran, H. Bahri, M. Atri, and J. Dubois. Optimized parallel implementation of face detection based on GPU component. *Microprocessors and Microsystems*, 39(6):393–404, aug 2015.

[34] C. T. Chu and J. N. Hwang. Fully Unsupervised learning of camera link models for tracking humans across Nonoverlapping cameras. *IEEE Transactions on Circuits and Systems for Video Technology*, 24(6):979–994, jun 2014.

[35] C. T. Chu, J. N. Hwang, H. I. Pai, and K. M. Lan. Tracking human under occlusion based on adaptive multiple kernels with projected gradients. *IEEE Transactions on Multimedia*, 15(7):1602–1615, nov 2013.

[36] R. T. Collins, A. J. Lipton, and T. Kanade. Introduction to the special section on video surveillance. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):745–746, 2000.

[37] R. T. Collins, A. J. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, O. Hasegawa, P. Burt, et al. A system for video surveillance and monitoring. *VSAM final report*, pages 1–68, 2000.

[38] M. H. Construction. The Business Value of BIM for Construction in Major Global Markets. Technical report, Construction, McGraw Hill, 2014.

[39] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, sep 1995.

[40] R. Cyganiak. Rooms ontology - DERI Vocabularies. http://vocab.deri.ie/rooms.html, 2012.

[41] R. Cyganiak, D. Wood, and M. Lanthaler. RDF 1.1 Concepts and Abstract Syntax. https://www.w3.org/TR/rdf11-concepts/, 2014.

[42] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 886–893. IEEE, 2005.

[43] L. Daniele, F. den Hartog, and J. Roes. Created in Close Interaction with the Industry: The Smart Appliances REFerence (SAREF) Ontology. In *Proceedings of the International Workshop Formal Ontologies Meet Industries*, pages 100–112. Springer, Cham, 2015.

[44] T. M. de Farias, A. Roxin, and C. Nicolle. A Rule Based System for Semantical Enrichment of Building Information Exchange. In *Proceedings of the RuleML*, aug 2014.

[45] M. Dibley, H. Li, Y. Rezgui, and J. Miles. An ontology framework for intelligent sensor-based building monitoring. *Automation in Construction*, 28:1–14, 2012.

[46] P. Dollar, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 36(8):1532–1545, aug 2014.

[47] P. Dollar, Z. Tu, P. Perona, and S. Belongie. Integral Channel Features. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 91.1–91.11. British Machine Vision Association, 2009.

[48] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 34(4):743–761, 2012.

[49] B. East. Construction-Operations Building Information Exchange (COBie) | WBDG - Whole Building Design Guide. http://www.wbdg.org/resources/construction-operations-building-information-exchange-cobie, 2016.

[50] C. M. Eastman, C. Eastman, P. Teicholz, R. Sacks, and K. Liston. *BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors*. John Wiley & Sons, 2011.

[51] V. L. Erickson, M. Á. Carreira-Perpiñán, and A. E. Cerpa. Observe: Occupancy-based system for efficient reduction of hvac energy. In *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 258–269. IEEE, 2011.

[52] M. Everingham, S. A. Eslami, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge: A Retrospective. *International journal of computer vision (IJCV)*, 111(1):98–136, 2015.

[53] R. Falco, A. Gangemi, S. Peroni, D. Shotton, and F. Vitali. Modelling OWL ontologies with Graffoo. In *Proceedings of the European Semantic Web Conference (ESWC)*, pages 320–325, 2014.

[54] M. Farenzena, L. Bazzani, A. Perina, V. Murino, and M. Cristani. Person re-identification by symmetry-driven accumulation of local features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2360–2367, 2010.

[55] T. M. D. Farias, A. Roxin, and C. Nicolle. IfcWoD, Semantically Adapting IFC Model Relations into OWL Properties. In *Proceedings of the 32nd CIB W78 Conference on Information Technology in Construction*, pages 175–185, 2015.

[56] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object Detection with Discriminative Trained Part Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(9):1627–1645, 2010.

[57] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. Technical report, Network working group, jun 1999.

[58] R. Fisher, J. Santos-Victor, and J. Crowley. CAVIAR: Context Aware Vision using Image-based Active Recognition. http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/, 2004.

[59] Y. Freund and R. E. Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1):119–139, aug 1997.

[60] D. Gerónimo, A. M. López, A. D. Sappa, and T. Graf. Survey of pedestrian detection for advanced driver assistance systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(7):1239–1258, jul 2010.

[61] T. Gevers and A. W. Smeulders. Color-based object recognition. *Pattern Recognition*, 32(3):453–464, mar 1999.

[62] H. Ghayvat, S. Mukhopadhyay, X. Gui, and N. Suryadevara. Wsn-and iot-based smart homes and their extension to smart buildings. *Sensors*, 15(5):10350–10379, 2015.

[63] R. Girshick. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[64] R. Girshick, D. Jeff, D. Trevor, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014.

[65] D. Gray and H. Tao. Viewpoint invariant pedestrian recognition with an ensemble of localized features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 262–275, 2008.

[66] M. Grüninger and M. S. Fox. The Role of Competency Questions in Enterprise Engineering. In *Benchmarking — Theory and Practice*, pages 22–31. Springer, Boston, MA, 1995.

[67] A. Hampapur, L. Brown, J. Connell, A. Ekin, N. Haas, M. Lu, H. Merkl, and S. Pankanti. Smart video surveillance: exploring the concept of multiscale spatiotemporal tracking. *IEEE Signal Processing Magazine*, 22(2):38–51, 2005.

[68] J. Han and C. Moraga. The influence of the sigmoid function parameters on the speed of backpropagation learning. In *Proceedings of the International Workshop on Artificial Neural Network*, pages 195–201. Springer, Berlin, Heidelberg, 1995.

[69] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[70] L. Hernandez, C. Baladron, J. M. Aguiar, B. Carro, A. J. Sanchez-Esguevillas, J. Lloret, and J. Massana. A survey on electric power demand forecasting: future trends in smart grids, microgrids and smart buildings. *IEEE Communications Surveys & Tutorials*, 16(3):1460–1495, 2014.

[71] P. Hitzler, M. Krotzsch, S. Rudolph, M. Krotzsch, and S. Rudolph. *Foundations of Semantic Web Technologies*. Chapman and Hall/CRC, 2009.

[72] J. R. Hobbs and F. Pan. Time ontology in OWL. https://www.w3.org/TR/owl-time/, 2017.

[73] J. Y. Hong, E. H. Suh, and S. J. Kim. Context-aware systems: A literature review and classification. *Expert Systems with Applications*, 36(4):8509–8522, 2009.

[74] I. Horrocks. Ontologies and the semantic web. *Communications of the ACM*, 51(11):58–67, dec 2008.

[75] I. Horrocks, O. Kutz, and U. Sattler. The irresistible SRIQ. In *Proceedings of the Workshop on OWL: Experiences and Directions (OWLED)*, page 11, 2005.

[76] I. Horrocks, O. Kutz, and U. Sattler. The Even More Irresistible SROIQ. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, volume 6, pages 57–67, 2006.

[77] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A semantic web rule language combining OWL and RuleML. https://www.w3.org/Submission/SWRL/, 2004.

[78] I. Horrocks, P. F. Patel-Schneider, and F. Van Harmelen. From SHIQ and RDF to OWL: The making of a Web Ontology Language. *Web Semantics*, 1(1):7–26, dec 2003.

[79] L. Hou, W. Wan, J. N. Hwang, R. Muhammad, M. Yang, and K. Han. Human tracking over camera networks: a review. *Eurasip Journal on Advances in Signal Processing*, 2017(1):43, dec 2017.

[80] L. Hou, W. Wan, K. H. Lee, J. N. Hwang, G. Okopal, and J. Pitton. Robust Human Tracking Based on DPM Constrained Multiple-Kernel from a Moving Camera. *Journal of Signal Processing Systems*, 86(1):27–39, jan 2017.

[81] International Organization for Standardization. ISO 16739:2013 Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries. https://www.iso.org/standard/51622.html.

[82] R. Iqbal, M. Azrifah Azmi Murad, A. Mustapha, and N. Mohd Sharef. An Analysis of Ontology Engineering Methodologies: A Literature Review. *Research Journal of Applied Sciences, Engineering and Technology*, 6(16):2993–3000, 2013.

[83] Javed, Rasheed, Shafique, and Shah. Tracking across multiple cameras with disjoint views. In *Proceedings of the 9th IEEE International Conference on Computer Vision (ICCV)*, pages 952–957. IEEE, 2003.

[84] O. Javed, K. Shafique, and M. Shah. Appearance modeling for tracking in multiple non-overlapping cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 26–33. IEEE, 2005.

[85] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg. Multiple Hypothesis Tracking Revisited. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4696–4704, 2015.

[86] I. S. Kim, H. S. Choi, K. M. Yi, J. Y. Choi, and S. G. Kong. Intelligent visual surveillance - A survey. *International Journal of Control, Automation and Systems*, 8(5):926–939, oct 2010.

[87] H. Kleine Buning and T. Lettman. *Propositional logic : deduction and algorithms*. Cambridge University Press, 1999.

[88] M. Kostinger, M. Hirzer, P. Wohlhart, P. M. Roth, and H. Bischof. Large scale metric learning from equivalence constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2288–2295. IEEE, jun 2012.

[89] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

[90] C. H. Kuo, C. Huang, and R. Nevatia. Inter-camera association of multi-target tracks by on-line learned appearance affinity models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 383–396, 2010.

[91] C. H. Kuo and R. Nevatia. How does person identity recognition help multi-person tracking? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1217–1224, 2011.

[92] O. Lassila. Serendipitous Interoperability. In *Proceedings of the Semantic Web Kick-Off Seminar*, pages 243–256, 2002.

[93] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[94] X. Li, K. Wang, W. Wang, and Y. Li. A multiple object tracking method using Kalman filter. In *Proceedings of the IEEE International Conference on Information and Automation (ICIA)*, pages 1862–1866. IEEE, jun 2010.

[95] S. Liao, Y. Hu, X. Zhu, and S. Z. Li. Person re-identification by Local Maximal Occurrence representation and metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2197–2206, 2015.

[96] K. Lin, M. Chen, J. Deng, M. M. Hassan, and G. Fortino. Enhanced fingerprinting and trajectory prediction for iot localization in smart buildings. *IEEE Transactions on Automation Science and Engineering*, 13(3):1294–1307, 2016.

[97] M. Lin, Q. Chen, and S. Yan. Network In Network. In *Proceedings of the International Conference on Learning Representation (ICLR)*, page 10, 2014.

[98] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature Pyramid Networks for Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2117–2125, 2017.

[99] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. Focal Loss for Dense Object Detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, 2017.

[100] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dolí. Microsoft COCO: Common Objects in Context. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 740–755. Springer, 2014.

[101] G. Lisanti, I. Masi, A. D. Bagdanov, and A. D. Bimbo. Person Re-identification by Iterative Re-weighted Sparse Ranking. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 37(8):1629–1642, 2015.

[102] C. Liu, S. Gong, C. C. Loy, and X. Lin. Person Re-identification: What Features Are Important? In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 391–401, 2012.

[103] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 21–37, 2016.

[104] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, pages 3431–3440, 2015.

[105] D. G. Lowe. Distinctive image features from scale invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004.

[106] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, X. Zhao, and T.-K. Kim. Multiple Object Tracking: A Literature Review. *arXiv preprint arXiv:1409.7618*, 2014.

[107] B. Ma, Y. Su, and F. Jurie. Discriminative Image Descriptors for Person Re-identification. In *Person Re-Identification*, pages 23–42. Springer London, London, 2014.

[108] D. Makris, T. Ellis, and J. Black. Bridging the gaps between cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 205–210, 2004.

[109] R. Marroquin, J. Dubois, and C. Nicolle. PhD forum: WiseNET - Smart camera network interacting with a semantic model. In *Proceedings of the 10th International Conference on Distributed Smart Cameras (ICDSC)*, pages 224–225, 2016.

[110] R. Marroquin, J. Dubois, and C. Nicolle. Multiple Ontology Binding in a Smart Building Environment. In *Proceedings of the Workshop on Linked Data in Architecture and Construction (LDAC)*, 2017.

[111] R. Marroquin, J. Dubois, and C. Nicolle. Know beyond seeing : combining computer vision with semantic reasoning. In *Proceedings of the12th IEEE International Conference on Semantic Computing (ICSC)*, pages 310–311, Laguna Hills, USA, 2018. IEEE.

[112] R. Marroquin, J. Dubois, and C. Nicolle. Ontology for a Panoptes building: Exploiting contextual information and a smart camera network. *Semantic Web*, 9(6):803–828, sep 2018.

[113] R. Marroquin, J. Dubois, and C. Nicolle. Wisenet: An indoor multi-camera multi-space dataset with contextual information and annotations for people detection and tracking. *Data in Brief*, page 104654, 2019.

[114] T. Matsukawa, T. Okabe, E. Suzuki, and Y. Sato. Hierarchical Gaussian Descriptor for Person Re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1363–1372, 2016.

[115] K. Meshgi, S. Ishii, H. Skibbe, S. Oba, S.-i. Maeda, and Y.-z. Li. An occlusion-aware particle filter tracker to handle complex and persistent occlusions. *Computer Vision and Image Understanding*, 150:81–94, sep 2016.

[116] C. Mignard and C. Nicolle. Merging BIM and GIS using ontologies application to Urban facility management in ACTIVe3D. *Computers in Industry*, 65(9):1276–1290, dec 2014.

[117] D. Mills, J. Martin, J. Burbank, and W. Kasch. Network time protocol version 4: Protocol and algorithms specification. Technical report, Internet Engineering Task Force, 2010.

[118] D. Minoli, K. Sohraby, and B. Occhiogrosso. Iot considerations, requirements, and architectures for smart buildings—energy optimization and next-generation building management systems. *IEEE Internet of Things Journal*, 4(1):269–283, 2017.

[119] B. Morvaj, L. Lugaric, and S. Krajcar. Demonstrating smart buildings and smart grid features in a smart energy city. In *Proceedings of the 3rd International Youth Conference on Energetics (IYCE)*, pages 1–8. IEEE, 2011.

[120] R. Mosqueron, J. Dubois, M. Mattavelli, and D. Mauvilet. Smart camera based on embedded HW/SW coprocessor. *Eurasip Journal on Embedded Systems*, 2008(1):597872, 2008.

[121] B. Motik, P. F. Patel-Schneider, and P. Bijan. OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition). https://www.w3.org/TR/owl2-syntax/, 2012.

[122] B. Motik, U. Sattler, and R. Studer. Query answering for OWL-DL with rules. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(1):41–60, 2005.

[123] A. Nambiar, M. Taiana, D. Figueira, J. C. Nascimento, and A. Bernardino. A multicamera video dataset for research on high-definition surveillance. *International Journal of Machine Intelligence and Sensory Signal Processing*, 1(3):267–286, 2014.

[124] S. M. Naqvi, L. Mihaylovay, and J. A. Chambers. Clustering and a joint probabilistic data association filter for dealing with occlusions in multi-target tracking. In *Proceedings of the 16th International Conference on Information Fusion*, pages 1730–1735. IEEE, 2013.

[125] N. F. Noy and D. L. McGuinness. Ontology Development 101: A Guide to Creating Your First Ontology. Technical report, Standford University, Stanford, CA, 2001.

[126] T. Ojala, M. Pietikäinen, and D. Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51–59, jan 1996.

[127] M. Ortega, Y. Rui, K. Chakrabarti, S. Mehrotra, and T. S. Huang. Supporting ranked boolean similarity queries in MARS. *IEEE Transactions on Knowledge and Data Engineering*, 10(6):905–925, 1998.

[128] C. Papageorgiou and T. Poggio. Trainable system for object detection. *International Journal of Computer Vision (IJCV)*, 38(1):15–33, 2000.

[129] C. Park, T. J. Woehl, J. E. Evans, and N. D. Browning. Minimum cost multi-way data association for optimizing multitarget tracking of interacting objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 37(3):611–624, mar 2015.

[130] G. Pass, R. Zabih, and J. Miller. Comparing images using color coherence vectors. In *Proceedings of the 4th ACM international conference on Multimedia*, pages 65–73, New York, New York, USA, 1996. ACM Press.

[131] P. Pauwels, T. M. de Farias, C. Zhang, A. Roxin, J. Beetz, J. De Roo, and C. Nicolle. A performance benchmark over semantic rule checking approaches in construction industry. *Advanced Engineering Informatics*, 33(C):68–88, aug 2017.

[132] P. Pauwels, T. Krijnen, W. Terkaj, and J. Beetz. Enhancing the ifcOWL ontology with an alternative representation for geometric data. *Automation in Construction*, 80:77–94, 2017.

[133] P. Pauwels and J. Oraskari. IFC-to-RDF-converter. https://github.com/mmlab/IFC-to-RDF-converter, 2016.

[134] P. Pauwels and A. Roxin. SimpleBIM: From full ifcOWL graphs to simplified building graphs. In *Proceedings of the 11th European Conference on Product and Process Modelling (ECPPM)*, pages 11–18. CRC Press, 2016.

[135] P. Pauwels and W. Terkaj. EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. *Automation in Construction*, 63:100–133, 2016.

[136] J. Perš, V. S. Kenk, R. Mandeljc, M. Kristan, and S. Kovačič. Dana36: A multi-camera image dataset for object identification in surveillance scenarios. In *Proceedings of the 9th IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*, pages 64–69, 2012.

[137] B. Prosser, S. Gong, and T. Xiang. Multi-camera Matching using Bi-Directional Cumulative Brightness Transfer Functions. In *Proceedings of the British Machine Vision Conference (BMVC)*, page 10, 2008.

[138] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, mar 1986.

[139] Y. Raimond and S. Abdallah. The Event Ontology. http://motools.sourceforge.net/event/event.122.html, 2007.

[140] M. H. Rasmussen, P. Pauwels, M. Lefrançois, G. F. Schneider, C. A. Hviid, and J. Karshøj. Recent changes in the Building Topology Ontology. In *Proceedings of the 5th Linked Data in Architecture and Construction Workshop (LDAC)*, page 7, 2017.

[141] J. Redmon. Darknet: Open Source Neural Networks in C. https://pjreddie.com/darknet/.

[142] J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement. *Computing Research Repository (CoRR)*, abs/1804.0, 2018.

[143] S. Ren, K. He, and R. Girshick. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks Shaoqing. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99, 2015.

[144] S. H. Rezatofighi, A. Milan, Z. Zhang, Q. Shi, A. Dick, and I. Reid. Joint probabilistic data association revisited. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 3047–3055. IEEE, dec 2015.

[145] B. Rinner and W. Wolf. An introduction to distributed smart cameras. *Proceedings of the IEEE*, 96(10):1565–1575, 2008.

[146] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi. Performance Measures and a Data Set forMulti-Target, Multi-Camera Tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 17–35, 2016.

[147] A. Rosebrock. Softmax Classifiers Explained - PyImageSearch. https://www.pyimagesearch.com/2016/09/12/softmax-classifiers-explained/, 2016.

[148] M. Rouse. What is open API (public API)? https://searchmicroservices.techtarget.com/definition/open-API.

[149] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the fisher vector: Theory and practice. *International Journal of Computer Vision (IJCV)*, 105(3):222–245, 2013.

[150] J. C. SanMiguel, J. M. Martinez, and A. Garcia. An Ontology for Event Detection and its Application in Surveillance Video. In *Proceedings of the 6th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 220–225, 2009.

[151] B. Senouci, I. Charfi, B. Heyrman, J. Dubois, and J. Miteran. Fast prototyping of a SoC-based smart-camera: a real-time fall detection case study. *Journal of Real-Time Image Processing*, 12(4):649–662, dec 2016.

[152] O. Sidla, Y. Lypetskyy, N. Brändle, and S. Seer. Pedestrian detection and tracking for counting applications in crowded situations. In *Proceedings of the IEEE International Conference on Advance Video and Signal Based Surveillance (AVSS)*, pages 70–76. IEEE, nov 2006.

[153] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proceedings of the International Conference on Learning Representations*, 2015.

[154] E. Simperl. Reusing ontologies on the Semantic Web: A feasibility study. *Data and Knowledge Engineering*, 68(10):905–925, oct 2009.

[155] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *Web Semantics*, 5(2):51–53, jun 2007.

[156] F. Sivrikaya and B. Yener. Time synchronization in sensor networks: a survey. *IEEE network*, 18(4):45–50, 2004.

[157] A. W. Smeulders, D. M. Chu, R. Cucchiara, A. Calderara, Simone Dehghan, and M. Shah. Visual Tracking: An Experimental Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 36(7):1442–1468, jul 2014.

[158] D. Snoonian. Smart buildings. *IEEE spectrum*, 40(8):18–23, 2003.

[159] S. Soro and W. Heinzelman. A survey of visual sensor networks. *Advances in multimedia*, 2009:21, 2009.

[160] K. Srikrishna, M. Gou, Z. Wu, A. Rates-Borras, O. Camps, and R. J. Radke. A Systematic Evaluation and Benchmark for Person Re-Identification: Features, Metrics, and Datasets. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 41(3):523–536, 2018.

[161] A. Stadlhofer, Bernd and Salhofer, Peter and Durlacher. An overview of ontology engineering methodologies in the context of public administration. In *Proceedings of the 7th International Conference on Advances in Semantic Processing*, pages 36–42, 2013.

[162] Standford University. Protégé. https://protege.stanford.edu/.

[163] R. Studer, V. R. Benjamins, and D. Fensel. Knowledge engineering: principles and methods. *Data & knowledge engineering*, 25(1-2):161–197, 1998.

[164] J. Suchan and M. Bhatt. Deep Semantic Abstractions of Everyday Human Activities. In *Proceedings of the 13th Iberian Robotics conference*, pages 477–488. Springer, 2017.

[165] J. Sweller. Cognitive load during problem solving: Effects on learning. *Cognitive Science*, 12(2):257–285, 1988.

[166] S. Tang, M. Andriluka, B. Andres, and B. Schiele. Multiple People Tracking by Lifted Multicut and Person Re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3539–3548, 2017.

[167] R. Thavot, R. Mosqueron, M. Alisafaee, C. Lucarz, M. Mattavelli, J. Dubois, and V. Noel. Dataflow design of a co-processor architecture for image processing. In *Proceedings of the Conference on Design and Architectures for Signal and Image Processing (DASIP)*, pages 1–8, 2008.

[168] D. Toczek, T.Ginhac, F. Hamdi, B. Heyrman, J. Dubois, J. Miteran, and D. Ginhac. Scene-based non-uniformity correction: From algorithm to implementation on a smart camera. *Journal of Systems Architecture*, 59(10):833–846, nov 2013.

[169] C. Town. Ontological inference for image and video analysis. *Machine Vision and Applications*, 17(2):94–115, 2006.

[170] J. R. Uijlings, T. Van De Sande, Koen EA Gevers, and A. W. Smeulders. Selective Search for Object Recognition. *International Journal of Computer Vision (IJCV)*, 104(2):154–171, 2013.

[171] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 511–518. IEEE, 2001.

[172] P. Viola, M. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *Proceedings of the 9th IEEE International Conference on Computer Vision (ICCV)*, pages 734–741. IEEE, 2003.

[173] S. Wang, M. Lewandowski, J. Annesley, and J. Orwell. Re-identification of pedestrians with variable occlusion and scale. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1876–1882. IEEE, nov 2011.

[174] X. Wang. Intelligent multi-camera video surveillance: A review. *Pattern Recognition Letters*, 34(1):3–19, 2013.

[175] X. Wang, T. X. Han, and S. Yan. An HOG-LBP human detector with partial occlusion handling. In *Proceedings of the 12th IEEE International Conference on Computer Vision (ICCV)*, pages 32–39. IEEE, sep 2009.

[176] Weiming Hu, Nianhua Xie, Ruiguang Hu, Haibin Ling, Qiang Chen, Shuicheng Yan, and S. Maybank. Bin Ratio-Based Histogram Distances and Their Application to Image Classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 36(12):2338–2352, 2014.

[177] S. K. Weng, C. M. Kuo, and S. K. Tu. Video object tracking using adaptive Kalman filter. *Journal of Visual Communication and Image Representation*, 17(6):1190–1208, dec 2006.

[178] T. Weng and Y. Agarwal. From buildings to smart buildings—sensing and actuation to improve energy efficiency. *IEEE Design & Test of Computers*, 29(4):36–44, 2012.

[179] Wikipedia. Color histograms. https://en.wikipedia.org/wiki/Color_histogram.

[180] Wikipedia. Ontology in philosophy. https://en.wikipedia.org/wiki/Ontology.

[181] T. Winkler and B. Rinner. Security and privacy protection in visual sensor networks: A survey. *ACM Computing Surveys (CSUR)*, 47(1):2, 2014.

[182] World Wide Web Consortium OWL Working Group. OWL 2 Web Ontology Language Document Overview (Second Edition). https://www.w3.org/TR/owl2-overview/, 2012.

[183] World Wide Web Consortium SPARQL Working Group. SPARQL 1.1 Overview. https://www.w3.org/TR/sparql11-overview/, 2013.

[184] F. Xiong, M. Gou, O. Camps, and M. Sznaier. Person re-identification using kernel-based metric learning methods. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 1–16, 2014.

[185] B. Yang and R. Yang. Interactive particle filter with occlusion handling for multi-target tracking. In *Proceedings of the 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, pages 1945–1949. IEEE, aug 2015.

[186] Y. Yang, J. Yang, and J. Yan. Salient Color Names for Person Re-identification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 536–551. Springer, 2014.

[187] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS)*, pages 3320–3328, nov 2014.

[188] L. Zhang, T. Xiang, and S. Gong. Learning a Discriminative Null Space for Person Re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1239–1248, 2016.

[189] S. Zhang, R. Benenson, M. Omran, J. Hosang, and B. Schiele. Towards Reaching Human Performance in Pedestrian Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 40(4):973–986, apr 2018.

[190] S. Zhang, R. Benenson, and B. Schiele. Filtered channel features for pedestrian detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1751–1760. IEEE, jun 2015.

[191] S. Zhang, E. Staudt, T. Faltemier, and A. K. Roy-Chowdhury. A camera network tracking (CamNeT) dataset and performance baseline. *Proceedings - 2015 IEEE Winter Conference on Applications of Computer Vision, WACV 2015*, pages 365–372, 2015.

[192] R. Zhao, W. Ouyang, and X. Wang. Unsupervised salience learning for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3586–3593. IEEE, jun 2013.

[193] W. Zhao and J. K. Liu. OWL/SWRL representation methodology for EXPRESS-driven product information model Part I. Implementation methodology. *Computers in industry*, 59(6):580–589, 2008.

[194] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian. Scalable person re-identification: A benchmark. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1116–1124, 2015.

[195] L. Zheng, Y. Yang, and A. G. Hauptmann. Person Re-identification: Past, Present and Future. *Computing Research Repository (CoRR)*, abs/1610.0, 2016.

[196] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 391–405. Springer, Cham, 2014.

[197] M. A. Zulkifley and B. Moran. Robust hierarchical multiple hypothesis tracker for multiple object tracking. *Expert Systems with Applications*, 39(16):12319–12331, nov 2012.

List of Figures

List of Tables

Document generated with LaTeX and:
the LaTeX style for PhD Thesis created by S. Galland — http://www.multiagent.fr/ThesisStyle
the tex-upmethodology package suite — http://www.arakhne.org/tex-upmethodology/