Luca Maggiani

Heterogeneous Smart Cameras: towards the Internet of Reconfigurable Things

Sant'Anna
School of Advanced Studies – Pisa

UNIVERSITÉ
Clermont
Auvergne

Academic Year
2016/2017

Phd Course
Emerging Digital Technologies,
Embedded Systems

Heterogeneous Smart Cameras: towards the Internet of Reconfigurable Things

Author
Luca Maggiani

Tutor
Prof. Marco Di Natale

Supervisors
Dott. Paolo Pagano
Prof. François Berry

Abstract

With the nowadays improvements of the information technology, more and more smart sensors are deployed to monitor and survey our daily life. Instead of the centralized approach, sensor networks are evolving towards complete Cyber-Physical Systems, where sensing, processing and communications are spread among the network. This decentralization comes at a price though. Moving sensing, processing and communication capabilities to the network edges, pose newer and unresolved challenges for data processing, management and system coordination.

This thesis aims at presenting a novel abstraction of distributed processing network to enhance the road users safety. Along with the definition of a system description formalism, the main contributions encompass a processing architecture proposal and a coordination model for smart sensing application. These contributions directly address the previously introduced challenges, by proposing and implementing an autonomous video-enabled Cyber-Physical System for the Smart City scenario.

By following the latest technological trends, we are on the verge of a new generation of smart objects, where reconfigurable heterogeneous Cyber-Physical Systems will be massively deployed in our daily life. The *Internet of Reconfigurable Things* paradigm is ready to come.

Contents

CHAPTER 1

Introduction

> 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*

1. The Ubiquitous Computing era

Human technologies have never been so embedded in the physical world. With the nowadays improvement in information and communication technologies, smart and connected objects are almost a constant presence in our daily life. Things were significantly different in 1991 though. Going back in time, Mark Weiser – the Principal Scientist at Xerox Palo Alto Research Center (PARC) – was proposing a new, visionary concept. In his idea, technology should interact continuously within our daily life. Seamlessly integrated with our senses that we would conceive the physical world entangled with invisible and pervasive sensors, actuators, displays and computational elements [**WGB99**]. No more *computer walls*, but embedded Human-Computer Interfaces (HCIs) connected through distributed networks. Technology becomes a mean of interaction with the physical world, where computers, cameras and sensors are seamlessly embedded into our world to make it efficient, safe and calm. Weiser's visionary concept has been eventually defined as Ubiquitous Computing (UbiComp), where ubiquitous stand for everywhere and anywhere. Transparent and noninvasive technologies that are able of computation, communication and information processing by themselves or in collaboration with other objects. Three main UbiComp properties have been introduced by Weiser [**Wei91**] as follows.

Technology needs to be networked and transparently accessible. Communications make nodes interact each other and enable complex behaviours. Weiser envisioned a smart home in which heterogeneous devices exchange seamlessly information and commands. By making nodes communicate, the access to the network itself might be also transparent from the medium used. For instance, no

more gateways are needed to translate packet from a point to another, everything can be directly forwarded to peers. No more gateways mean also an interoperable communication protocols.

Human Computer interaction needs to be more hidden. Since their first appearance, Personal Computers (PCs) have been devoted to desk and to office workstation. This has been particularly important when Weiser first envisioned a "distributed" vision. When removing the hypothesis that the computing site is targeted to a specific location but rather can be deployed pervasively, also the HCIs dramatically change. From monitor and character inputs, towards noninvasive, integrated interfaces. In case of small sensors and monitoring solution, there is no point in having standard interfaces. In [**Wei91**], these were identified through smart badges and proactive control solutions (e.g., automatic door and heating control).

Computers needs to be aware of the environment context. Again, when referring to the PC scenario, the technology has no clue about the location where it operates. In order to preserve its general purpose specification, it does not require information about the physical positioning. And so does with respect to others (relative positioning). What Weiser proposes is nowadays called context-aware computing. Since the platform should operate and interact with the environment, it has to be aware of what is around it. This represents also a base condition to make the UbiComp really pervasive: make nodes be aware of the surrounding environment to proactively act with it.

FIGURE 1.1. Ubiquitous Computing.[1]

[1]source: http://www.ubiq.com/weiser/weiser.html

2. Beyond Weiser's vision

Despite the relatively old idea, the UbiComp is all but forgotten. Recent surveys [**Kin11**] and later research evolutions, e.g., ambient intelligence [**AW09**], everywhare [**Gre10**], pervasive computing [**Sat01**], make the UbiComp concept still up and running. Although the multiple definition that have appeared, the research perspectives seem converge to the Cyber-Physical System (CPS) definition. A CPS is ultimately the UbiComp concept applied with nowadays available technologies and applications. What makes the CPS paradigm still appealing for the research perspective is due to the opened challenges that still today do not have closed-ended answers [**RLSS10**].

Moreover, the CPS concept has been added by the UK Computing Research Committee (UKCRC) in the Grand Research Challenges 2004 [**HM05**,**CCC+06**] to achieve a major milestone in the successive two decades. In particular, the manifest rises challenges for both computer engineers and software designers to extend the present Internet infrastructure to support the UbiComp scenario. The major milestone indeed challenges the computing architecture and interaction model for significant advancements of knowledge and technology.

1.1. Computing architecture. At first, the computing architecture is considered as the ground of distributed processing to enable higher level data analysis. It is indeed designated to host complex and flexible processing algorithms within embedded platforms and limited computational resources. Other functional specifications might also be applied, such as processing time constraints (soft real time, hard real time), energy balance, installation and deployment requirements, end-user cost, etc. The overall computing requirements are pushing the research academia to find suitable architectures and algorithm solutions that support distributed computation sites deployments. Therefore, the target architecture has to be flexible yet optimized for seamless processing configurations.

1.2. Interaction model. Once suitable computing architectures have been selected, the challenge encompasses the communication policies. While the computing architecture locally involves the node capabilities in terms of data handling, the interaction capabilities have to be considered. The communication indeed make nodes sharing context-aware features and even processing instances.

Moreover, when considering heterogeneous network of computing nodes, communication also act as a coordination service to define optimal solutions with a

set of the available resources. This aspect also involves the support of adaptability at many different levels. Applications providing timely information need to be adapt to the users' current context in terms of location, activity and communication capability. The network environment needs to adapt to provide services, especially for those with specific quality of service requirements. System reconfiguration is also considered when newer applications are required, to automatically detect and adapt to the available resources to support them.

Another important aspect regards the context-aware reasoning. Information from multiple heterogeneous sources need to be automatically integrated to create consistent information databases. Database's entries are related to redundant measures taken by the available network nodes. Redundancy makes system more reliable in terms of measurement accuracy and node failure. Nevertheless, measurements fusion need to be carried out accordingly to the node coordination. In case of node failure, the system has to recover its operative mode by self-reconfiguring the lost connections.

3. Towards vision-enabled CPS

As already mentioned, the computing architectures are nowadays moving away from centralized and static deployment to autonomous entities distributed in the environment [**FPF17**]. This trend is supported by the latest improvements in processing technologies that permits the integration of *invisible* computing platforms in our daily life [**Sat17**]. Traffic control sensors, access authentication systems, environmental monitoring and autonomous vehicles are becoming widespread within our cities [**MSPC12**, **GBMP13**, **PLJC14**, **ZBC+14**].

At first, sensor networks have been deployed to measure scalar physical events (e.g., light, sound, temperature) but then eventually extends their computing capabilities to locally extract more complex features from the surrounding environment [**AAS+15**]. By overtaking the limits of classical centralized architectures, sensor nodes are currently addressing multimedia capabilities and in particular vision processing is becoming part of the *cyber physical* layer. Obviously, images and videos unleash new frontiers of context understanding and comprehension. This feature comes at a price though. Indeed, by considering video-enabled devices, the computing architecture and the communication models are once again under the spotlights.

These vision-enabled devices, known in literature as Smart Cameras (SCs), are nowadays challenging the processing and communication models to perform complex, computationally intensive processing operations with limited computational resources. Indeed, a smart camera is an advanced vision system which provides imaging sensing and feature extraction means. These capabilities enable SC to process the environment data and to take decisions as autonomous entities. The Smart Camera Network (SCN) is indeed nowadays an emerging research field promoting the natural evolution of centralized computer vision applications towards full distributed and pervasive systems. Differently from Wireless Sensor Networks (WSNs) which are generally supposed to perform basic sensing tasks, SCNs consist of autonomous devices, performing collaborative applications, leveraging on-board image processing algorithms optimized in respect of the limited available resources [**RWS+08**].

Such vision systems are more than electronic devices, as many people perceive it. They need advanced electronics design but also software engineering, and proper networking service design and implementation. They are built following design challenges of size, weight, cost, power consumption, and limited resources in terms of computing, memory, networking capabilities. At the same time, more and more services are required by customer specifications, so that time-to-market is setting a major challenge for architects on the design of effective and powerful solutions. From the research perspective, the European academia is also committing to accomplish a seamless integration of SCNs into the next generation CPSs to match emerging societal needs.

The decreasing cost and increasing performance of embedded smart camera systems makes it attractive to the above mentioned applications. The main research focus is directed at designing a system with sufficient computing power to execute well-known image processing algorithm in real-time. Typical scenarios for deploying traditional smart cameras are applications in well-defined environments, with static sensor setups, as traffic surveillance, intelligent transport systems, human action recognition [**QT08**] [**WR09**].

4. Application scenario

The work described in this thesis tackles the context of the SCN in the Smart City scenario. For instance, in Fig. 1.2 the roadway environment is considered. It does involve pavements, bicycle paths, and motor-vehicles lanes. These are

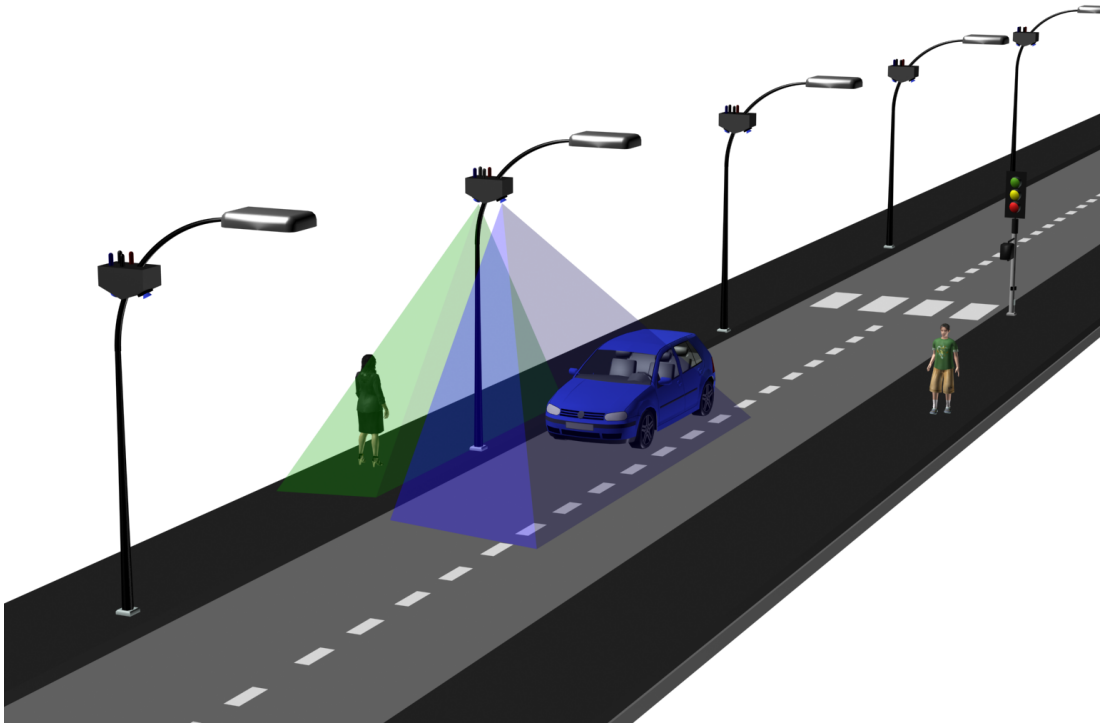


FIGURE 1.2. Application scenario

the static components defining the scenario's structure and background. Along with the structure, the *road users*, e.g. pedestrians, vehicles, cyclists, animals, represent the moving elements with dynamic and sometimes unpredictable behaviours. In this environment, it is likely that the use integrated and autonomous SCN could provide significant improvements in terms of scene understanding and preventing road accident. Although highly regulated, the road environment still present a huge variability, especially when referring to road users. Having a pervasive network of advanced sensors indeed enables better scene understanding and a context-aware reasoning, which ultimately might reduce causalities.

In considering the literature on the subject, the exact relationships between the many factors that come into play and affect road accidents are not fully known and are likely to vary unpredictably. Approximately 1.24 million people die every year on the world's roads and another 20 to 50 million sustain injuries as a result of road traffic accidents [$\mathbf{O^+13a}$]. By 2020, road accident injury is likely to be the third leading cause of disability-adjusted life years lost [$\mathbf{P^+04}$].

Moreover, half of all road traffic deaths are among vulnerable users, and more than one fifth of them are pedestrians [**O$^+$13b**].

In this manuscript, the *advanced video sensors* consist of a set of heterogeneous[2] devices capable of monitoring the environment and eventually generate events to signal others significant behaviour or dangerous situations. In this vision, we can imagine a future SCN pervasively monitoring and controlling many activities in our daily life. Small and reconfigurable smart cameras create a dynamic network, which can be used for a wide range of applications. Notably in smart cities:

Accident detection: whenever an accident occurs, the system automatically detects a threat situation and takes specific actions to provide final users by the needed information.

Smart traffic surveillance: able to regulate the amount of vehicles over a specific road and determine the best traffic light configuration on the basis of real-time traffic measurement.

Pedestrian detection: whenever a pedestrian is detected an alert is propagated to incoming vehicles.

5. Contributions and outline

This thesis aims at presenting a novel abstraction of a distributed processing network designed to enhance the road users safety. In Chapter 2, the network of system architecture is introduced. When referring to distributed and heterogeneous sensor deployments, an architecture model is required to clearly describe and assess the system behaviour. Regardless of the target architecture, a model would improve the system configuration and the management phase. The considered CPS scenario provides an ideal case study for the Smart Camera paradigm due to the large scale deployments and the need of distributed processing architectures. This model development also introduces the aspect processing and communication abstractions which are then detailed in the following sections.

Chapter 3 focuses on event generation and processing requirements in pervasive SC deployments. In envisioning scalable vision-enabled applications, only (necessary, semantic) information are exchanged among peers. These exchanges

[2]In this manuscript, heterogeneous is defined as a different nature of devices, either hardware or software oriented, seamlessly integrated towards a common purpose.

eventually involve *events*, defined as the semantic result of the scene understanding. An event can be a position of a pedestrian, the speed of a vehicle or a behaviour anomaly. These events are the result of high-level computer vision detection and might be further processed by nodes. In Chapter 4, previously considered processing and communication issues are addressed. Given the natively concurrent application, heterogeneous parallel architectures are considered. The models presented in Chapter 2 are then mapped to a flexible Field Programmable Gate Array (FPGA) architecture. By leveraging the processing and communication parallelism, the proposed architecture is finally considered for the Smart City scenario.

Finally, Chapter 5 introduces a formal interaction model for smart sensing application. Along with the architecture model described in Chapter 2 and the event processing pictured in Chapter 3, this section acts as the global understanding level between nodes. According to a stochastic model, SC nodes are capable of sharing the information they retrieve from the environment and are able to autonomously coordinating.

Chapter 6 concludes this thesis and provides open future perspectives.

CHAPTER 2

Network of systems

The Cambridge Dictionary [**Cam16**] gives system definition as follows: "*a system is a set of connected things or devices that operate together*". This definition already contains several key-points, namely *connected*, *devices*, *operate together*. A system is composed by one or several devices and offers external interfaces to be connected to other entities. These interfaces make possible information exchanges with other systems and with the surrounding environment.

1. Background

The concept of network of systems appeared in literature in the late 90es referring to a theoretical foundations of complex entities gathered together to form a mixture of several homogeneous and/or heterogeneous systems. It was first introduced in [**EMM91**], as System of Systems (SoS), to describe, analyse and simulate engineering processes. In particular, the methodology has been first applied to industrial domains, such as aeronautics [**HDW17**], transportation

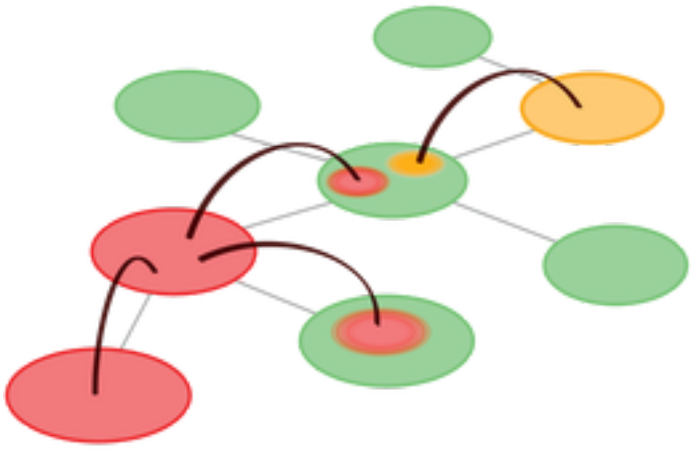


FIGURE 2.1. Network of systems connected via communication links [**PMC+13**]

[**DeL08**], defence [**DLRB08**], processing management [**KWG17**] and biology [**PMC+13**]. In Fig. 2.1, an epidemiological disease diffusion is described as a network of systems. The single entity refers to a person whom may interact with others, though communication links. The speed and the spatial diffusion are indeed results of interactions between affected entities.

Same applies to a network of systems composed by smart devices. In this manuscript, the considered network of systems is composed by autonomous processing entities, with their own objectives and procedures to follow. Each entity is not independent though, but a part of a complex environment that allows communications through communication links. According to [**BS06**], a network of systems usually shows characteristic features, presented as follows:

Autonomy. Each system entity has specific tasks. In other terms, the entities are acting autonomously with respect to their tasks, whereas their contributions might be globally visible.

Belonging. Based on their needs, individual entity independently decide on whether they want to be a part of a the system or not. This aspect is particularly important when dealing with moving entities in dynamic environments: the parental relation becomes something variable during time.

Connectivity. This property suppose the ability of each part to team up with other systems in order to enhance the overall capability. As a result, this affects the system decentralization, where connections are created based on the effective needs. For intrinsically varying environments, the connectivity is also important to make the system aware of local changes.

Diversity. The current trend in high performance and embedded computing consists in designing increasingly complex heterogeneous hardware architectures with non-uniform communication resources. Diversity permits to the system to improve its ability to adapt itself to unexpected variations.

Emergence. When referring to adaptive systems, usually the local configuration is considered. In the CPS context, interactions between parts might enable novel system configurations and functionality. In this terms, through internal collaborations, the system can reshape its behaviour to meet evolving user needs, to overcome parts failures or simply to find an improved way to perform the same action. This feature is particularly important for environments that may

change very rapidly. Also to counter react to malfunctions or act of vandalism. In [**WS15**], emergence is also considered the ability of context sensitive systems to adapt itself to a newer, unpredicted event.

2. Structure model

In this section, a formalization of the distributed environment structure is proposed. This model is then used to describe and evaluate a distributed processing systems. According to Chapter 1, a distributed network is composed by concurrent entities capable to retrieve, process and produce data interacting with the network. This behaviour results of processing and connectivity management made by nodes themselves in order to achieve specific tasks.

The problem around how to model the structure of concurrent systems found its origin when computer science has started to understand the behaviour of communicating programs [**Bae05**]. As communications among computer entities became more and more complex, new ways of modeling structures and behaviours have been proposed. With these assumptions, several mathematical expressions to describe and analyze concurrent computational entities have been proposed.

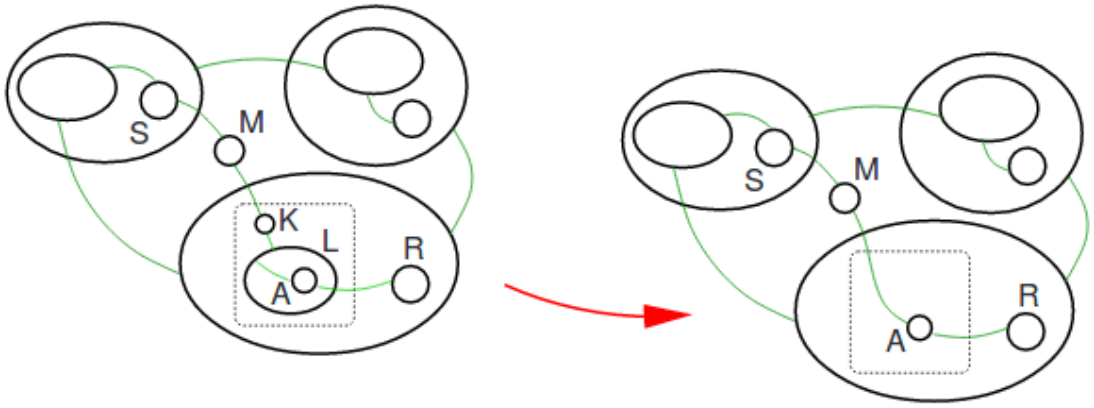


FIGURE 2.2. The Figure shows three main entities with communication links among them. In the example reported in [**Mil09**], the message M is produced by S and it is passing through the link towards L and acts as the activation token for K. The results of the iteration is then highlighted on the right. The L operator can be seen as an activation rule, that makes A processing M and forwarding the results to R.

Such formal approaches, usually known as process calculi for concurrent systems [**Bae05**], represent a step towards context-aware devices. They are based on independent systems and have recently proposed in representing interaction and communication between entities including context-awareness [**SW16**]. When referring to ubiquitous systems, the Bigraph [**Mil09**] model comes at first glance.

Bigraphs have been introduced specifically to model ubiquitous systems, where concepts of computing entity, locality, connectivity and interaction are involved. Such models not only considers processing capabilities, but also extends to interaction and communication tasks. With respect to other process calculi, e.g., π-calculus [**Mil99**], bigraphs are indeed introducing the notion of *place*. In other terms, while π-calculus is more oriented to the space of communication links, bigraphs also considers the spatial context where links are placed.

In the followings, the concept of process calculi for concurrent systems and the bigraph model are applied to analytically describe the CPS environment. Indeed, the bigraph theory already carries the elements to describe the considered application scenario. In particular, a CPS entity considers a space where communication links and processes move, as proposed by [**San93**]. This emergence behavior indeed particularly of interests when referring dynamic interactive processing entities.

Definition 1. *A computing site is an abstract formalization of the spatial and temporal context where data interactions take place.*

This definition introduces the "container" where links and processing are defined. As in service-oriented architectures, the definition detaches how the node externally behaves, e.g. interactions and data exchanges, with respect to the internal architectures involved. In other words, a computing site represents the substrate that permits link and processing being instantiated, as a cloud server makes available its resources to users that might remotely require specific operations. A computing site can be formally described as follows:

$$Comp_j : f(OP, link, scope) \tag{1}$$

The $Comp_j$ is composed by a set of Operator (OP), link and scope. An OP specifies an atomic functional behaviour and involves data exchanges with other OPs by means of links. Whenever a link is defined, an unidirectional interface exists between a couple of OPs. For instance, in Fig. 2.3 the $Comp_j$ contains four OPs instances, and a set of communication links represented as line segments.

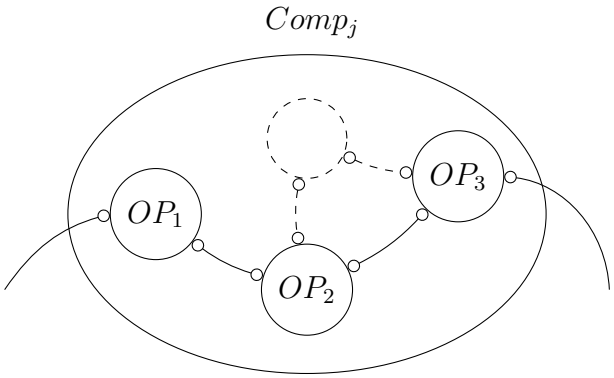


FIGURE 2.3. Computing site.

Both operators and links are locally visible within the $Comp_j$ scope, delimited by the external ellipse.

DEFINITION 2. *A link defines a logical connection between operators. It does not automatically involve the existence of a physical communication channel, but rather a relationship between two operators.*

This definition detaches the concept of *physical* channel from *logical* relations. In Chapter 4, these links will be defined as wirings between processing operators (e.g., serial communication) or indeed logical data dependencies (e.g., a processing token). In other terms, a link represents a logical relation between operators which might also involve physical communication channels.

DEFINITION 3. *An operator OP is an abstract atomic entity, describing a functional behaviour. It does not involve target-specific details.*

Each OP shows a specific behaviour according to its own configuration. A formal OP definition can be expressed as follows:

$$OP_i : f(category, ports) \tag{2}$$

where the $i-th$ operator is defined by its category, namely its own behaviour, and by the available ports. The port allows edges to be connected creating links. According to the CPS requirements, three main categories have been considered: an OP can either be a transducer T, a communication C or a processing P entity.

- *Transducer T*
- *Processing P*
- *Communication C*

The transducer T enables interfaces towards the physical world, where sensing and actuation are involved. In particular, it defines the node external interfaces thus its class and belonging, e.g., a CMOS image sensor labels the node as a video-enabled node, a displacement sensor labels the node as an accelerometer node. At the same time, when considering CPS systems also actuators might be involved. In this case, the transducer operator permits an active interface towards the physical world through an actuator, e.g., automatic lighting control, variable messages signs. With respect to other operators later introduced, the transducer represents intrinsic physical feature and might not be modified without direct intervening on the node itself.

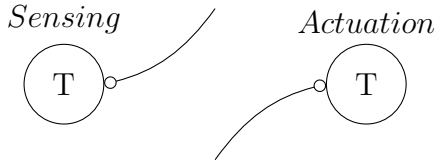


FIGURE 2.4. Transducer operator

The communication C operator instead is taking care of all that concerns the data exchanges between nodes. It might export multiple communication mediums permitting nodes communicate each other. Beneath the node architecture, the C components act as data dispatcher between transducer and processing entities by means of unidirected links.

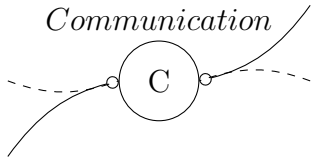


FIGURE 2.5. Communication operator

Finally, the processing P component involves data handling and processing. According to its own internal rules, data incoming through links are handled, e.g., processed, and results are made available within the computing site according to link configurations.

DEFINITION 4. *A site describes the physical space where an OP can be instantiated. It involves architecture specific features of the target architecture which may impact the operator behaviour.*

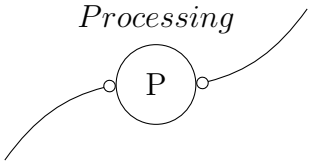


FIGURE 2.6. Processing operator

A site (as the dashed circle in Fig. 2.3) indeed defines architecture-specific constraints. Regardless of the target platform, the site abstraction provides a methodology to assess whether a new OP instance can be deployed. The site physical meaning involves hardware specifications (e.g., architecture model, computation capability, memory, interfaces), thus determines the subset of OPs that might be instantiated. For instance, in hardware-oriented platforms (e.g., FPGA) directly refers to the available hardware resource while in software-oriented platforms rather considers the remaining computing time, memory and I/Os.

In Fig.2.7, a more set of operator example is proposed. It is inspired from the smart city scenario and encompasses the sensing and actuation layers. It involves two image sensors and a variable message panel, outlined as the actuator. T_1 and T_2 represent the image sensors, thus source nodes for the sytem, while T_3 symbolize the actuator. Among the considered operators, communication and processing are provided by C_1, C_2, C_3 and P_1, P_2 respectively.

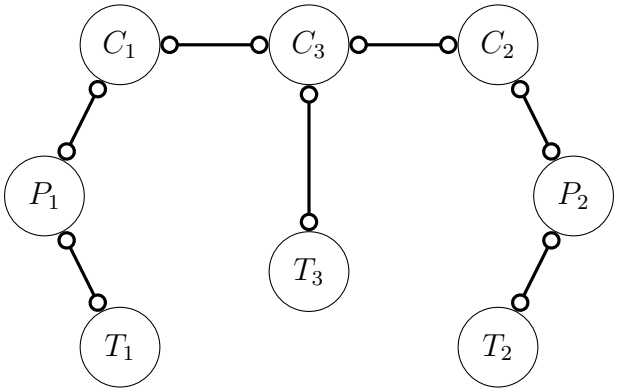


FIGURE 2.7. Computing site operators example.

Until now, the computing site is somewhat abstract, a container which hosts operators instances and their communication channels. It is a purely functional definition without constraints. However, it is clear that in practice sites and links are limited by the available resources. In this respect, the bigraph theory

comes to setup a clear and formal rules to properly bind operators, sites and links together. In the follows, two rules are being considered, the place-sorting and the link-sorting constraints.

DEFINITION 5. *The place-sorting constraint can be used to ensure that exactly one outward-binding node is connected per inward-binding port [**Jen06**].*

This rule defines the compatibility between available sites and the operator to be instantiated. The interface direction, namely outward or inward, sets the subset of the possible operator instantiation. For convenience, the forms of bindings here are taken as primitive, allowing sites to have both inward- and outward-binding ports. The place-sorting routine is provided by the *place graph*, which also describes the place-hierarchy.

The link-sorting constraint has been introduced by [**LM06**] to further clarify the link formation rule. Indeed, it is not meaningful to connect two communication operators without an intervening processing operator.

DEFINITION 6. *The link-sorting discipline defines link formation rules between operators. Indeed, it permits only specific connections to be made thus defining a "building semantic".*

Although it might sound straightforward, the link formation rule is especially of use when considering autonomous link formation. The discipline can be used as a guideline in creating the so called *link graph*, which is the ultimate goal of a self-coordinating network.

2.1. Network extension. By leveraging the previous definitions, the computing site can be further extended to a network of computing entities. The extension makes use of the place-sorting and link-sorting constraints to create *place graph* and *link graph* and ultimately to describe the distributed system. With respect to Fig. 2.3, the nodes are assembled by means of their structural appearance (e.g., operators, sites) and their interacting behaviours as well.

A bigraph treats locality and connectivity as separate graphs (as orthogonal aspects [**Jen06**]), *place graph* and *link graph* respectively.

The network is indeed a composition of operators and their communication links. It is represented by a bigraph G, defined as follows:

$$G = (OP_G, place_G, link_G) \tag{3}$$

The graph G contains a set of operators OP_G, place $place_G$ and $link_G$ graphs then describe the system structure. The configuration in Fig. 2.7 is then mapped to the link graph in Fig. 2.8. The graph G is composed by two *regions*, $\mathbb{R}_1$ and $\mathbb{R}_2$, which define the environment and the computing site roots. A set of computing sites $\mathbb{C} = \{Comp_1, Comp_2, Comp_3\}$ is defined. Each one hosts several operators $OP_1, ..., OP_8$. In the diagram below, the bigraph G is shown by means of its constituents, the place graph $place_G$ and $link_G$.

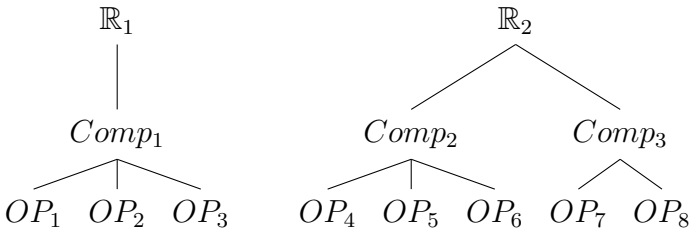


FIGURE 2.8. Place graph $place_G$.

The $place_G$ is a collection of rooted trees based on their belongings. In the example, the $place_G$ is rooted by two regions $\mathbb{R}_1, \mathbb{R}_2$. $Comp_1$ belongs to region $\mathbb{R}_1$ while $Comp_2, Comp_3$ to $\mathbb{R}_2$. Finally, operators are subordinated with respect to the container agent. The $place_G$ graph then presents an outer face of $\{\mathbb{R}_1, \mathbb{R}_2\}$ and an empty inner face, since no sites are available. This results in $place_G : 0 \to \{\mathbb{R}_1, \mathbb{R}_2\}$.

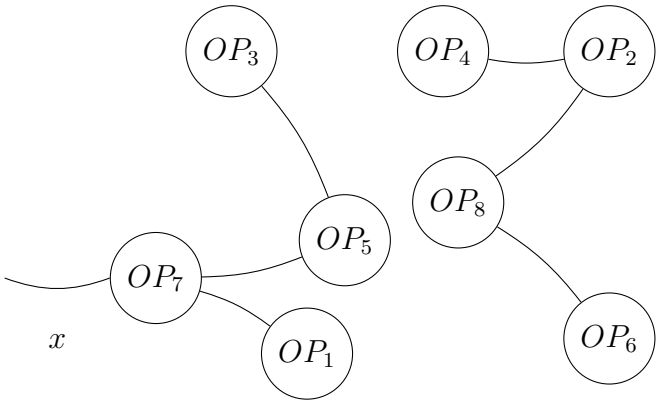


FIGURE 2.9. Link graph $link_G$.

The $link_G$ instead resumes the connections between entities, regardless of their placements. As shown in Fig. 2.9, its inner face is the subset of E_G that

connects operators within the bigraph G. The $link_G$ exhibits the x external connection thus its outer face is $\{x\}$ while inner face results empty. The graph link is then expressed as $link_G : 0 \to x$.

2.2. Network category. By leveraging the place and link constraints, the first application of the bigraph methodology concerns the network classification. Indeed, by applying different rules the operator interactions can heavily be modified. In considering the SCN scenario, the first classification might be inferred as function of how links moves between cameras. This aspect plays a major role when dealing with distributed coordination (as described in Chapter 5) and has important effect in modeling the system behaviour. Secondly, the movement can also consider the processing, as envisaged in the place-mapping routine.

Therefore, a network of computing sites can be differentiated in:

- **Static network** Once installed, the place and link graphs are not expected to change during time, except in case of a node failure. Once they reached a stable state, the place and link graphs are constant. Notable application include video surveillance and environmental monitoring.
- **Semi-static network** Whenever mobile nodes are considered, nodes iterations are required to follow newer communication channels. The link graph dynamically updates the operator connections, while place graph would conserves its structure. This is the case of a moving node within a fixed infrastructure, e.g., a moving vehicle that interacts with the road-side CPS infrastructure.
- **Dynamic network** The last case removes all the constraints over place and link graphs and foresee fully dynamic network topologies. Moving objects continuously update their connectivity models and operator locality might move according to the requirements. This flexibility comes at price of a more complex system description and modelling.

In Chapter 5, the static network deployment is considered. Further extension approaches the semi-static category, through new nodes insertions. The dynamic behaviour of the place-graph is not addressed here, but rather considered as the future perspectives.

2.3. Place-mapping methodology. According to the previous definitions, each computing site makes available, within its scope, sites for operator instantiations. Let consider the example of Fig.2.7. It describes several operator connections, while it does not specify the operators' scope. By applying place and link sorting constraints, a possible pair $place_G, link_G$ can be derived. Assume that the system is composed by the three computing sites as in Fig. 2.8, with eight available sites. One mapping strategy can regroup T_1, C_1 and P_1 in $Comp_1$ and T_2, C_2 and P_2 in $Comp_2$. Therefore $Comp_1, Comp_2$ involve acquisition, processing and communication behaviours as they act as sources within the network. On the other side, $Comp_3$ only involves the actuation while does not include processing.

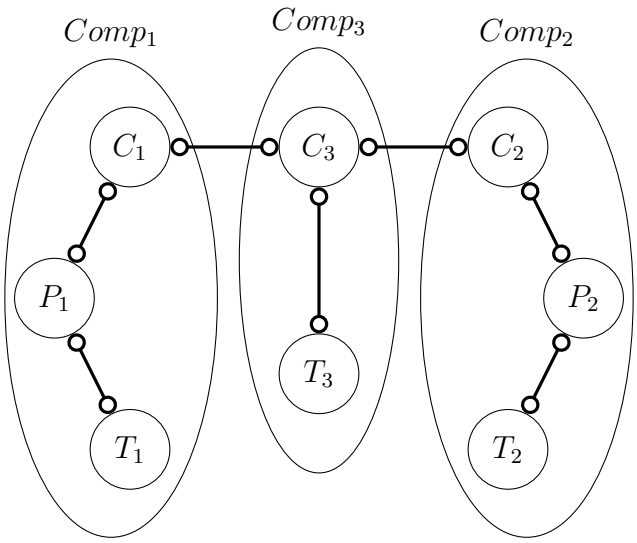


FIGURE 2.10. Example of operators mapping.

This simple example opens up several considerations. First, the operator mapping defines the node functional category. Based on the operators locality, the dominant node behaviour can be used to distinguish node categories. With this respect, a smart camera results in a device which embeds sensing, processing and communication capabilities, thus $Comp_1, Comp_2$ would probably belong to this category. A node that only performs actuation would instead be classified as a sink node. Same applies to pure processing nodes, which host P operators while retrieve data from external sources (e.g., cloud-oriented nodes) and so on.

Secondly, the mapping strategy heavily affects the node behaviour and the system architecture as a consequence. The mapping function might indeed consider

cost, power efficiency, processing time and robustness metrics. Thus it requires a mathematically-formulated, reproducible ways to evaluate how the application would perform in a target architecture. In the following Section, a model of architecture is introduced to address this aspect.

3. Model of Architecture

In the previous section, the system structure has been defined as composed by the place-graph, which involves architecture-specific constraints, and the link-graph, which involves data exchanges between operators. A distributed application can be then described as a concurrent system, where data dependencies and interactions are mapped to cost functions. In this section, the work proposed in [**PDM$^+$15**], and later extensions [**PDM$^+$16**], comes to define a methodology to evaluate performance indexes when an application is mapped to a specific platform. In [**PDM$^+$15**] a Model of Architecture (MoA) is developed as a design space exploration tool to assess the efficiency of different Model of Computations (MoCs) mapped onto specific architectures. The MoA is "is an abstract efficiency model of a system architecture that provides a unique, reproducible cost computation, unequivocally assessing a hardware efficiency cost when processing an application described with a specified MoC." Indeed, The MoA provides reproducible computation cost when an application is mapped to a target architecture. In particular, it aims at providing quantitative yet abstract performance index for a defined application deployment.

With respect to the referred works, in this manuscript the MoA provides an evaluation of the performance of a distributed application mapped over a network of heterogeneous nodes. The distributed application drives the OP requirements while nodes coordinate the OP instances according to their capabilities, resources and costs.

The following notions are required prior to the model definition: application activity, quanta and tokens.

Definition 7. *The application activity $\mathbb{A}$ corresponds to the amount of processing and communication necessary for accomplishing the requirements of the application. The application activity is composed of processing and communication tokens.*

Definition 8. *A quantum q is the smallest unit of application activity.*

There are two types of quanta: processing quantum q_P and communication quantum q_C. These quantum are formally equivalent, thus represent the same amount of activity. However, since they represent different physical behaviours, they do not share the same unit of measurement. For instance, in a system with an unique clock and byte addressable memory, 1 cycle of processing quantum and 1 Byte as the communication quantum. Since q_P and q_C represent the smaller unit of the application activity, they are straightforward converted to energy, time and latency quantities for a physical architecture.

DEFINITION 9. *A token τ is a non-divisible unit of application activity, composed by a number of quanta.*

As for the quantum, a token can be either a processing processing tokens τ_P or a communication token τ_C and belongs to the token sets T_P or T_C respectively. The $T_P = \{\tau_P^1, \tau_P^2, ..., \tau_P^N\}$ is the set of processing tokens composing the application processing and $T_C = \{\tau_C^1, \tau_C^2, ..., \tau_C^M\}$ is the set of communication tokens composing the application communication. While quantum defines a single application step, the token represents the ensemble of operations required to produce a result. A token describes how many elementary q_P are required to perform a run-to-completion task or how many q_C to send a packet over the network.

Once an operator gets fired, it takes τ_P, τ_C to perform the required action. The overall application activity $\mathbb{A}$ can be defined as follows:

$$\mathbb{A} = \{T_P, T_C\} \tag{4}$$

By leveraging the two levels of activity quantization, the application activity reflects the cost in terms of processing and communication overhead and management.

The previously bigraph abstractions, place and link graphs, can be then associated with cost functions, thus providing effective and reproducible ways of evaluating the system architecture performance. As previously mentioned, the T operator is tightly related to the platform configuration, thus it is considered differently from other OPs. With respect to Eq. 3, the model of architecture Σ associates a cost function to the $place_G$ and $link_G$ graphs.

$$G = (OP_G, place_G, link_G)$$
$$\Omega = (\mathbb{A}, cost(T_P), cost(T_C)) \tag{5}$$

where OP_G is mapped the application activity $\mathbb{A}$, the place graph $place_G$ as the cost related to T_P executions and the link graph $link_G$ as the cost related to T_C execution. Once G is created, the resulting Ω provides cost computation when the activity $\mathbb{A}$ is mapped into the target architecture. The cost unit is specific to the application being modeled. It can be nJ for energy evaluation rather than delay cycles for timing performance evaluation. In general, for a platform U the cost functions $cost^U(T_P)$, $cost^U(T_C)$ are defined as:

$$cost^U(T_P) : T_P, P_n \rightarrow \alpha_n^U \cdot dim(T_P) + \beta_n^U$$
$$cost^U(T_C) : T_C, C_n \rightarrow \alpha_m^U \cdot dim(T_C) + \beta_m^U \tag{6}$$

where α_n^U is the fixed cost when τ_P is executed over P_n and α_m^U is the fixed cost when τ_C is executed over C_m [**PDM$^+$16**]. Clearly, the cost functions rely on the target platform U. Since tokens τ are homogeneous units of the application activity, α costs are equally distributed over the T_P or T_C elements. The β_n and β_m addends instead resume the cost overheads for P_n and C_n respectively. This overhead can be explained as a fixed cost when the operator is fired. This might involve instance costs, reconfiguration overheads and pipeline stalls.

Finally, the application cost in the platform U is as follows:

$$cost^U(\mathbb{A}) = cost^U(T_P) + \lambda \cdot cost^U(T_C) \quad \lambda \in \mathbb{R} \tag{7}$$

where T_P and T_C costs are linearly combined to evaluate the overall application footprint. Since T_P and T_C do not share the same measurement units, a conversion factor λ is applied. This conversion factor makes processing and communication costs comparable and heavily depends on the target platform regardless of the architecture mapping operation. Same applies for the α and β parameters, which are retrieved from the architecture specifications.

The above presented MoA gives a way of comparing different architecture platforms with a fixed application activity. According to Eq. 5, the architecture model indeed provides the mapping costs as function of the link-graph and place-graph specifications.

The goal is twofold. At first, the MoA-bigraph methodology can optimize the mapping strategy through place and link graphs reformulations. It obviously requires unaltered application behaviour while permits the reduction of the activity cost in a target platform, identified by the set $\{\alpha, \beta, \lambda\}$. Secondly, Eq. 5 can be iteratively applied to a set $\{\alpha^U, \beta^U, \lambda^U\}$ describing different platforms U. Such a methodology is particularly of interest when dealing with heterogeneous platforms, where specific strengths can be exploited.

4. Conclusion

This chapter presents the author's contributions related to the formalization of a distributed network of autonomous nodes. Starting from the general definition of a distributed context, the structure model has been presented. Such a formalism, inspired from the Milner's bigraph theory, is used to describe and analyze a concurrent distributed system. It also introduces the concept of Operator, which is an abstract entity describing a functional behaviour. The operators concur to the system formation as data handler and data processing entities. According to the structure model, the system is then formed as the combination of the place graph with the link graph. The former describes the operators instance placement and boundaries while the latter considers how the operators are connected according to the required behaviour. The proposed model closely describes the behaviour of a real distributed system, thus serving as the starting point for the proposed Smart Camera architecture described in the following chapters.

CHAPTER 3

Event generation for distributed systems

Since in the environment a distributed sensing is autonomously performed by each sensor, a cloud of local semantic results – called *events* – is then generated. According to the relevant literature [**KJBB09**, **WDWS10**, **WDA+12**, **NIJP14**, **KVH16**], an event is "an important phenomenon that occurs or might have occurred". In a more formal sense, an event is the semantic result of a combination of physical measures associated to specific behaviours. It can stand for a particular occurrence or rather for a particular not expected situation (e.g., anomaly). The event identification is therefore the procedure through which a physical phenomenon is reliably revealed, detected and classified. From the CPS device point of view, event identification opens the way of context-aware, perhaps getting closer to what Weiser stated as "Computers needs to be aware of the environment context".

Notable applications nowadays encompass the Smart City scenario, with traffic control sensors, access authentication and environmental monitoring [**MSPC12**, **PLJC14**]. More is yet to come, especially for video-enabled deployments in health-care [**CFK+14**, **PEK+15**, **Hos16**], vehicles tracking [**ZLY14**], video surveillance [**FKSK15**], people tracking [**KDM+14**] and distributed computing [**CCT+15**]. These latest works, also show that event identification is far more complex than the simplistic acceptation of deviation from a set-point threshold. Although effective in well-constrained context, such an approach is inadequate for complex events, whose require a deeper understanding of the environment [**KVH16**]. Since then, newer event detection techniques have been developed to provide *context-aware* CPSs.

In this Chapter, an architecture of smart camera is described as part of its main processing components. Then an overview of the event identification algorithms for image processing is presented. These techniques challenges the computing the computing architecture to provide novel and more complex event

detection algorithms which show characteristic advantages or disadvantages for specific target applications.

1. Event detection approaches

The CPS distributed architecture that is being defined is shown in Fig.3.1. The green box highlights the macro-blocks that have been described up to here. The environment *Env* is observed through image sensing, performed by the nodes themselves. Such measures are then processed to detect events thus performing local scene understandings.

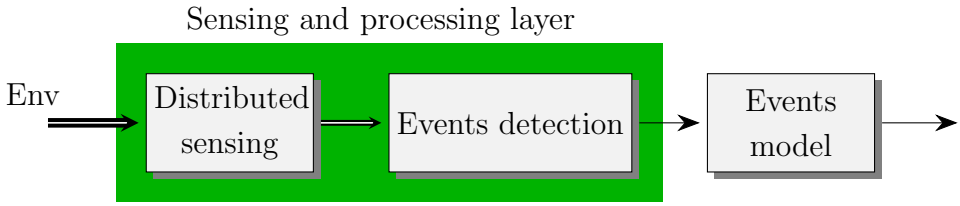


FIGURE 3.1. Distributed sensing and processing architecture.

Through machine vision algorithms, the sparse and noisy image data are sampled to canonical projections [**Koe84**]. A projection describes a particular set of the image features that are of interest, e.g., shape, luminance, boundary, movement. In this context, an event becomes the occurrence of a target projection (or a set of them) within the image space. It is up to the vision algorithm to reliably reveal such projections through image processing operations. An extensive literature is available for event detection algorithms in image processing according to each tailored application [**TMM$^+$16**].

Among them, three main approaches are encountered:

- Change detection,
- Pattern-matching,
- Machine learning.

3.1. Change detection. The first intuitive image event detection technique is obviously related to change detection. The goal is to identify set of pixel that are significantly different between the reference image I_{ref} and the current one I_i, with $i \in \mathbb{N}^+$. The differences are described in the *change mask*, which results as the mathematical combination of $\{I_{ref}, I_i\}$.

The image I_i contains $N \times M$ pixels and it is defined as an euclidean space $\mathbb{R}^{N \times M}$. The difference between $\{I_{ref}, I_i\}$ is therefore the distance in the image space. Such differences result as topological structures that are considered to perform the event identification.

In Fig. 3.2 a common change detection pipeline is shown. The pre-processing stage suppresses or filter out unimportant yet disturbing features from the image, e.g., nonuniform light distribution, affine transformation, camera motion, CMOS sensor noise. The change mask is then computed according to the image processing algorithm. Finally, the change mask is processed to reveal *blobs*, literally close regions of the image that differ from the surrounding pixels. This segmentation step ultimately reveals whether an event has been detected.

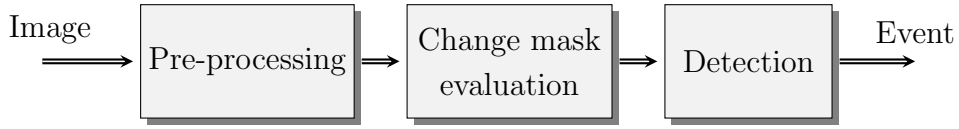


FIGURE 3.2. Event detection: Change detection pipeline.

In [**RAAKR05**, **CB96**] surveys, a comprehensive literature of change detection algorithms is considered. Important applications of change detection algorithms include video surveillance [**KS94**], parking detection [**MMN$^+$11**], medical diagnosis [**BHA$^+$03**,**PE07**], civil infrastructure [**NYSF12**] and agriculture [**TKH$^+$08**]. Although the considered applications assume steady camera deployments and rigid camera registration, they all consider quiescent drift of the image structure. Indeed, the scene model is only virtually static during the application, i.e., camera vibration, changing lighting conditions, gradually affect the structure model. In order to propose effective algorithms, the state-of-the-art methods always involve policies to update the reference image I_{ref} during time.

With this respect, the updating strategies can be distinguished in single-shot or continuous modes. The former refers to those environments where the structures are suddenly changing after a stable state. For instance, in medical diagnosis [**PE07**], a single-shot update allows an objective definition of disease progression, e.g., cancer evolution. Same applies to civil engineering [**NYSF12**], where comparisons are periodically performed with respect to the initial setup point. These application obviously do not consider significant scene changes between updates, thus are meant to well-constrained environments where geometry, reflectance and illumination do not change over time.

Continuous mode updating is instead required in case the scene variability period is comparable with the image sampling period. In such cases – e.g., video surveillance footage, leak detection and, in general, external video acquisitions – the I_{ref} image is continuously updated by using previously captured frames. The procedure still complies with Fig. 3.2, while involves statistical methodologies to update the I_{ref} frame. These methods, in literature also referred as background subtraction models [**Pic04**], aim to react to permanent changes in the scene (e.g., luminance distribution) while filtering spurious transitions.

The background $I_{ref}$is continuously updated as the average value of each pixel for a longer period compared with the frame acquisition [**TLL07**]. Averaging methods differ according to the background subtraction algorithm involved. The easiest provides the long-term average method:

$$I'_{ref} = \frac{1}{i} \sum_{i} I_i \tag{8}$$

where newer images incrementally decreases their contributes. However, this eventually leads to the algorithm failure due to sudden scene variations, which are not absorbed by the background. Improvements can be provided by on-line learning methods, such as the Q-learning [**Wat89**], which equally weights newer contributes over the older ones to compute the newer I'_{ref} image.

$$I'_{ref} = (1 - \alpha) I_{ref} + \alpha I_i \quad \alpha \in \mathbb{R}(0, 1) \tag{9}$$

where I'_{ref} is the new background model and α the learning rate.

Further improvements involve *multi-modal approaches*, where foreground and background are statistically modeled as stochastic distributions. In this respect, Gaussian Mixture Model (GMM) [**SG99**] provides a rigorous statistical model to describe pixels with a set of adaptive Gaussian distributions. The temporal variation of each pixel is modeled by the weighted mixture Gaussians, each one of them described by three parameters: *mean value* (μ), *variance* (σ^2) and *weight* (w).

3.2. Pattern-matching. Differently from change detections, the pattern-matching algorithms provide event identification through specific data sequence occurrences [**LV94**]. Instead of temporally modelling the environment, pattern-matching relies on the spatial and geometrical properties within images compared to a reference template. In particular, belongs to this category the area-based algorithms, where areas or regions of the original image data are matched with

minimal pre-processing or with pre-processing that preserves most of the image [**ENLM11**]. In this case, areas are basically compared through linear correlation without explicit correspondence between points.

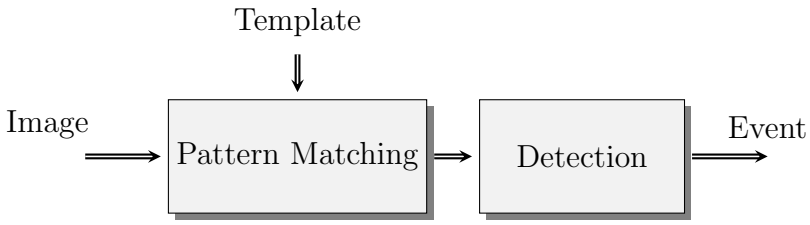


FIGURE 3.3. Event detection: Pattern-matching algorithms.

The pattern-matching similarity image correlation can be expressed as follows:

$$D(I,T) = \sum_{i,j \in T} \big| I\big(x+i, y+j\big) - T\big(i,j\big) \big| \tag{10}$$

where I and T denote, the search image and the template region, respectively. The sum is done over all i, j coordinates and the similarity is expressed as the $L1$-norm as the sum of absolute differences. The correlation results in a coefficient that is higher as the pattern is similar to the processed region. The operation needs to be repeated iteratively through-out the search image to reveal all pattern occurrences. According to Fig. 3.3, each matching result is then evaluated by a detector stage, which fixes thresholds and filter spurious detections.

Correlation-based methods share the following common drawbacks: (i) they operate pixel-wise comparisons of pixel regions, (ii) they are affected by luminance variation and noise, (iii) they might involve affine transformation to optimize the similarity between the template to the target pixel region.

At first, the pattern-matching approaches are indeed computationally intensive. With $N \times M$ image pixels, they require $o(N^2 \cdot M^2)$ operations with data dependencies that decrease the efficiency. Although other methods might speed up the elaboration (correlations in the frequency domain), the computing requirement becomes prohibitive when dealing with full-scale image resolution and strict timing constraints. Other optimizations have been proposed, such as [**ZF03**], to reduce the pattern iteration over selected regions that shows higher correlation after the first operations [**Ros16**].

The second issue concerns the luminance variation effect. Since pattern-matching uses local pixel intensities as template references, it obviously relies

on luminance distribution. In literature, this drawback has been mitigated by using normalized intensity pixel regions or by exploiting more complex correlation schemes, such as in the Census Transform [**ZW94**]. These alternatives significantly improves the pattern-matching descriptor yet increases the computational cost.

Finally, whether the template is not aligned with the image region, translation will not be enough. A pre-processing step is therefore needed to perform scale and rotation transformations to match the referred areas. These pre-processing operations shows the rotation have a worst case time complexity $o(N^2 \cdot M^3)$, while recent work [**Ros16**] reduces it to $o(N^2 \cdot M^2)$.

3.3. Machine learning. Current trends in decision-making algorithms increasingly support Machine Learning (ML) techniques as the core foundation of event identification. Machine learning algorithms are indeed promising technologies due to their ability in understanding complex context behaviours [**BMP+10**] thus improving the scene understanding. This aspect is even more of relevance when addressing vision-enabled CPS, where sparse image features have to be revealed and classified.

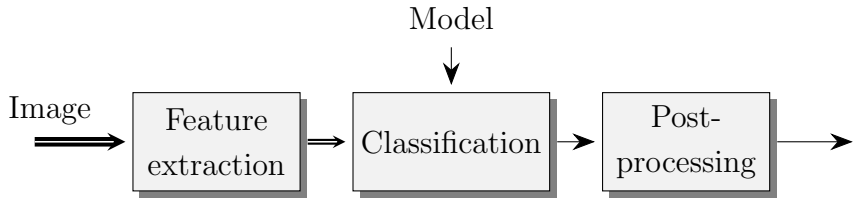


FIGURE 3.4. Event detection: Machine learning

A common machine learning processing flow is depicted in Fig. 3.4. With respect to the pattern-matching algorithm, feature-extraction pre-processing is usually preferred to template due to its robustness to light distribution, orientation and noise. Such image features are peculiar, highly distinctive set of points that match local image properties. The goal of the feature extraction is to provide a robust descriptor, regardless of the geometric distortions, rotations, points-of-view and nonuniform illumination, that might affect the image. In feature extraction, the following requisites are crucial:

- **Robustness** An information lost during the feature extraction step directly impacts the classification phase [**JGDE08**]. The classification

error is therefore the results of the feature weaknesses and misclassified classification descriptors.

- **Reliability** Similar image features in different images should provide similar descriptors. In other words, the feature dimensionality should carries enough information to reveal distinctive set of points in the image. This is aspect obviously is mandatory when real-world applications are envisaged.
- **Complexity** The computational requirements are also at the utmost importance. They pose stringent constraints to the processing architecture, especially when time critical application are addressed.
- **Specificity** Feature extractions are used for a wide range of application. The selected method is required to reveal distinctive properties of the target application. For instance, feature extraction for pedestrian detection enhances human silhouettes rather than road signs.

A large number of feature extraction methods are available in the literature, especially for pedestrian detection. A complete survey is out of the scope here, the reader might refer to the following overviews [**KKN14**], [**DWSP12**], [**KKWS15**], [**YSK$^+$15**].

Once the features have been extracted, the classifier evaluates the descriptor according to a decision function. The nature of the decision function depends on the machine learning methodologies involved. Since their first proposal in the 50ies, several ML concepts have been proposed, such as Artificial Neural Networks (ANNs) [**Wer74**], linear regression Generalized linear model (GLM) [**NB72**] or probabilistic reasoning Support Vector Machine (SVM) [**Vap82**]. Machine learning methods are separated between two macro classes: unsupervised machine learning and supervised machine learning algorithms.

Unsupervised machine learning. The unsupervised machine learning is a automatic learning methodology able to classify the input stimuli by inferring the classification function among the available data set. With respect to the supervised methodology, it only relies on unannotated data and a-priori unspecified object classes. The machine indeed compares the data and seeks for similarities or pattern correlations and create class maps where data are sorted. Clearly, the unsupervised method is particularly efficient in classifying those classes that

present a limited feature spaces, e.g., search words in a textbook, and with numerical representations. However, when referring to a specific event pattern, the unsupervised learning is not able to classify according to other classes. By its nature, this machine does not provide binary distinction between the feature space and specific input patterns. This condition also applies to the classification accuracy, that is meaningless without the comparison with a reference ground truth. Notable examples include Fuzzy Logic [**KY95**], Hidden Markov Model (HMM) [**BP66**], k-means clustering [**Jai10**].

Supervised machine learning. Supervised machine learning aims at classifying specific samples pattern by exploiting a set of reference examples, which are part of a training phase. Such reference examples are initially provided (e.g., the supervised phase) and labelled according to a specific feature. Once the model has been learnt, the machine is able to discriminate whether a new input belongs to the trained class (e.g., classification phase). This supervised method can be applied to a wide range of applications. For instance, it has been applied in healthcare to reveal whether the patient would present crisis by exploiting previously acquired samples or to detect handwritten notes by learning the writing style through past document. A training sample consists in a data structure with the corresponding object label. The learning algorithm then processes the training samples and generates a data model capable at *classifying* object labels among the trained ones. Notable examples include SVM [**Vap82**], k-Nearest Neighbors (k-NN) [**Alt92**] and Convolutional Neural Network (CNN) [**LBBH98**]. Closer to the SC domain, supervised machine learning algorithms have been successfully used for vehicles detection [**SBM06**], pedestrian detection [**DT05**] and facial recognition [**GLC00**].

2. Events in Smart Camera Networks

The complexity of event identification poses critical challenges especially in the Smart Camera Network (SCN) scenario. Among the different fields of application, common shared challenges might be summarized. In the following, a non-exhaustive list of challenges is presented, with a particular interest to computational intensive event generations [**KVH16**].

Timing critical application. The system is subject to timing deadline which should not missed otherwise the task might fail. Such systems are expected to guarantee response time within the specification, regardless of the processing operation involved. In case a deadline miss occurs, the failure might degrade the performance (soft deadline) or might endanger the human lives (hard deadline). Notable examples are in video-enabled Advanced Driver Assistance Systems (ADAS), where pedestrian presence needs to be revealed on-time to prevent collision.

Data reliability. The system operates in an environment which changes very rapidly and measures are also affected by an intrinsic noise. Nevertheless, it should be robust enough to provide *reliable* event detections. The robustness of a system is here defined as the capability of sorting among the available measures which are those to rely on. This aspect usually involves higher level data analysis, i.e. events post-processing tasks.

Complex event. As previously introduced, the exceeding threshold event might not be sufficient when referring to more complex scenario. In order to augment the environment understanding, the system should extract higher semantic events that represent more complex behaviour. These more complex events need to be efficiently executed in computationally limited embedded platforms.

Heterogeneity. CPSs are generally composed by heterogeneous entities. Diversity indeed improves the ability to adapt themselves to newer external conditions, unpredictable node mobility and dynamic network variations. However, heterogeneity requires formal architecture models to be efficiently deployed.

Adjustability. Due to the variable environment conditions, even event detections must also support a degree of run-time adaptation to newer application requirements.

Two main challenges arise from the application point of view. First, vision-enabled CPS should provide *reliable* detection information. This aspect obviously is the essential condition to rely on computer vision deployment and already represents a challenging task so far. But, because they operate in an environment which intrinsically changes very rapidly, they should also provide this information in real-time. Most of the time, given the complexity of the involved computer vision algorithms, this requires a computing power which is still not available from Commercial-Of-The-Shelf (COTS) embedded processing platforms.

In this manuscript, a feature-based supervised machine learning algorithm is considered. With respect to Fig. 3.4, we selected the Histogram of Oriented Gradients (HOG) as feature descriptor due to its robustness in real world implementation [**DWSP12**]. As the classifier stage, the SVM algorithm has been selected. Although newer methods have recently been proposed, according to several surveys [**DWSP12**,**KKWS15**,**YSK+15**], the SVM classifier still shows comparable detection performance while requiring less computational resources. Moreover, SVM training phase is significantly shorter than deep-learning methodologies, that ease the SC development. To further improve the detection reliability, as the post-processing step we also propose a spatio-temporal filtering stage. In the followings, the implemented event detection system is detailed by its composing steps as in Fig. 3.4. The implementation details will be then detailed in Chapter 4.

3.1. Feature extraction. In the context of pedestrian detection, descriptors relying on spatial orientation patterns, such as Local Binary Patternss (LBPs) [**OPH96**] and HOG [**DT05**] have been specially studied. With LBP, neighbor pixel comparisons are encoded with binary strings to describe the local pattern distribution. Since the result is based on relative luminance, this method limits the dependence to global luminance variation. HOG-based methods, on the other hand, aim at improving the descriptor robustness for structured image patterns. For this, the local spatial characteristics are described with a distribution of local intensity gradients within uniformly spaced cells. The resulting gradient histogram within each cell, possibly normalised, represents an element of the final HOG descriptor. Due to its invariance to luminance variation and good detection performance [**BOHS14**], the HOG descriptor is well suited to mobile vision applications.

The extracted features are HOG, computed on a dense grid of uniformly spaced cells and normalized. Practically, the input image is divided into small connected regions, called *cells*, and within each cell an histogram of gradient directions is computed. The resulting descriptors are sent to a pre-trained SVM-based classification machine.

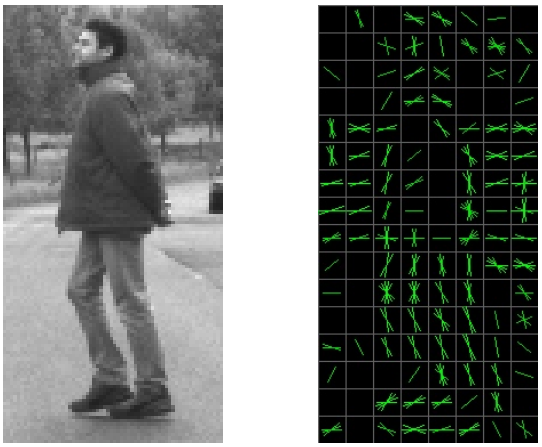
Following Dalal's formulation in [**DT05**], the 1-D spatial gradient components are first computed as follows:

$$G_x = \frac{\partial I}{\partial x} = I\big(x+1, y\big) - I\big(x-1, y\big) \tag{11}$$
$$G_y = \frac{\partial I}{\partial y} = I\big(x, y+1\big) - I\big(x, y-1\big)$$

Then the gradient magnitude $\|\nabla I(x,y)\|$ and orientation angle θ are evaluated as in Eq. 12.

$$\|\nabla I(x,y)\| = \sqrt{G_x^2 + G_y^2} \tag{12}$$
$$\theta = \arctan \frac{G_y}{G_x}$$

Within each cell, gradient orientations are then discretized and accumulated in a small number of histogram bins. Each histogram is finally normalized using a normalization factor derived from the mean intensity computed across a larger region of the image (block). This normalization results in better invariance to changes in illumination or shadowing [**GT07**]. The final HOG descriptor of an image is obtained by aggregating the normalized histograms.



(A) (B)

FIGURE 3.5. (a) a pedestrian image from the Daimler dataset. (b) the associated HOG Descriptor.

Fig. 3.5b shows an example image and its associated HOG descriptor. It is worth to note that the foreground silhouette is clearly enhanced with respect to

the background details. These features are completely meaningless without a supervised learning machine.

3.2. Classification. Machine-learning based systems are generally used to find correlations between the computed descriptors and a reference model. These systems typically emit detection results with their associated confidence interval. The higher is the confidence in the result, the higher is the probability of a correct classification. In the context of pedestrian detection, SVM are commonly used for performing the classification step, as a widespread and efficient supervised learning technique, based on strong mathematical foundations. The goal of this technique is to separate sets of descriptors in two classes. Once the reference model has been generated (with an off-line training phase), SVM produces detection classifications by labelling the input feature descriptors. Although algorithmically simple, application of SVM to image classification raises challenging implementations issues, especially if real-time performance level is required, as in pedestrian detection applications.

The classification stage compares the produced HOG descriptor with a pretrained reference model. This results in a binary decision, that would mark the current descriptor for belonging to one of two categories. The detection reliability is then expressed as a confidence interval, that is usually exploited to assess the classifier accuracy.

The SVM classifier compares the input vector descriptor with a reference model, which is produced by a supervised learning phase. The training step builds the reference model by assigning a great number of labelled examples to a specific class among a set of *classes*. The resulting model is then used online to map each descriptor to one of the predefined classes (non-probabilistic binary classification).

More formally, given a set of l data elements $\mathbf{x_1}, ..., \mathbf{x_l}$ and their corresponding classes $y_1 = y(\mathbf{x_1}), ..., y_l = (\mathbf{x_l})$ where $\mathbf{x_i} \in \mathbb{R}^d$ and $y_i = \pm 1$, the classification function can be expressed as follows:

$$y(\mathbf{x}) = sgn\left(\sum_{i=1}^{N_{SV}} y_i \alpha_i K(\mathbf{x_i}, \mathbf{x}) + b\right), 0 \leq \alpha_i \leq C \tag{13}$$

where $K(\mathbf{x_i}, \mathbf{x})$ is a kernel function, C is a regularization constant, α_i and b are parameters given by the learning phase. N_{SV} is the number of reference features, called Support Vectors (SVs). The examples $\mathbf{x_1}, ..., \mathbf{x_l}$ are HOG feature vectors

evaluated from database images and y corresponds to the associated class. In the case of a binary classification with a linear kernel, the general Eq. 13 can be simplified as follows:

$$y(\mathbf{x}) = \mathbf{w}^T \cdot \mathbf{x} + b \tag{14}$$

where $\mathbf{w}$ and b are model parameters, produced by the training phase. Note that in Eq. 14 the *sgn* operator has disappeared because what is actually used is the *distance* of the considered input with respect to the decision hyperplane.

In most published work, evaluation of object presence is performed in a limited number of regions of the image, called Regions of Interest (ROI). We propose, by contrast, to perform this evaluation for all possible positions of a detection window, therefore called *sliding window*, in the camera field of view. This approach has the advantage of being able to automatically detect new objects in images, whereas ROI-based approach can only focus on predefined targets and need some kind of tracking system to adjust the position of the ROIs.

The equation 13 is then generalized to take into account the fact that the response of the SVM step is now a matrix in which each element gives the detection result for a given detection window in the original image:

$$Y = \begin{bmatrix} y_{1,1} & \cdots & y_{1,M} \\ \vdots & \ddots & \vdots \\ y_{N,1} & \cdots & y_{N,M} \end{bmatrix} = \begin{bmatrix} y_{n,m} \end{bmatrix} \tag{15}$$

where $n \in \{1, \cdots, N\}$ and $m \in \{1, \cdots, M\}$. The values N and M respectively corresponds to the number of windows in the vertical and horizontal direction.

3.3. Post-processing. Because of their potentially pervasive applications, pedestrian detection systems are a very active research area. Although SVM-based system have already shown good detection performance with real-world images [**HPD09**, **DWSP12**], work is still on-going in order to improve the detection reliability. For this, two main directions have been suggested. The first is to extract and exploit several sets of feature descriptors. The second is to improve the robustness of the extracted feature descriptors.

Work described in [**ZHYT11**, **YPW+15**, **TTY+15**, **HKHK15**] belong to the first category. In particular, in [**ZHYT11**] and [**YPW+15**] HOG-LBP and HOG-Local Self-Similarity (LSS) algorithms are respectively deployed to increase the detection accuracy with multiple algorithm predictions. A further improvement

has been recently proposed in [**TTY+15**], where a HOG-LBP detection process is applied simultaneously and separately on the three channels of color images. These implementations rely on weighted inferences among the available detections to improve the final decision result. It can be noted, however, that requiring simultaneous computation of several feature sets places severe constraints on the implementation, especially if real-time behavior is targeted. In [**HKHK15**], the computation of a combined HOG-Discrete wavelet transform (DWT) is restricted to selected ROIs in image in order to reduce the global computation latency. A similar approach is presented in [**LYD+15**], where ROI-based pedestrian multi-view pose evaluations are processed with a HOG-LBP pipeline followed by a non linear SVM. Nonetheless, ROI-based approaches inherently reduce the "active" image part. In an environment which changes very rapidly, finding an acceptable trade-off between global computation time and the number of active ROIs is a complex problem, especially for time critical applications.

The second investigated approach for improving detection reliability concerns the descriptor robustness. Basically, the idea is to extend the feature descriptor dimension in order to improve its robustness in real-world deployment. In [**WLP15**], for instance, the HOG descriptor is improved by taking into account similarity on the local representation of contours. Since the pedestrian appearance usually presents symmetry properties, a geometric feature description is added. A two-level cascade of SVM classifiers is then applied to the extended descriptor. Even though this method improves the performance with respect to a classic HOG descriptor, the processing performance overhead once again reduces the applicability of the method in embedded platforms. Another approach is to augment HOG descriptors with temporal information obtained from video sequences. In [**HMY+15**], for example, the authors aggregate several descriptors obtained by different techniques to extract temporal information from images. The 3DHOG descriptor proposed in [**KMS08**] for characterizing motion features with a co-occurence spatio-temporal vector also belongs to this category. The HOGHOF descriptor using histogram of optical flow and the STHOG (Spatio-temporal histogram of oriented gradient) proposed by the author both increase the discriminative nature of the usual HOG descriptor space, at the price, here again, of a highest computation cost.

3. Detection reliability

As previously introduced, the classification reliability is one of the most critical point for an event detection system. Classically, reliability is assessed by means of two indicators: the true positive detection rate (TPR) and a false positive detection rate (FPR). The former corresponds to the "sensitivity" of the detection process and the latter to its "specificity". Best methods are supposed to achieve the highest TPR while keeping the lowest FPR.

Reliability numbers obtained with pure SVM-based algorithms are generally not sufficient for an effective deployment. In response, several post-processing techniques have been proposed to improve this reliability, essentially by reducing the FPR.

Basically, the idea is to extend the feature descriptor dimension in order to improve its robustness in real-world deployment. In [**WLP15**], for instance, the HOG descriptor is improved by taking into account similarity on the local representation of contours. Since the pedestrian appearance usually presents symetry properties, a geometric feature description is added. A two-level cascade of SVM classifiers is then applied to the extended descriptor. Even though this method improves the performance with respect to a classic HOG descriptor, the processing performance overhead once again reduces the applicability of the method in embedded platforms. Another approach is to augment HOG descriptors with temporal information obtained from video sequences. In [**HMY$^+$15**], for example, the authors aggregate several descriptors obtained by different techniques to extract temporal information from images. The 3DHOG descriptor proposed in [**KMS08**] for characterizing motion features with a co-occurence spatio-temporal vector also belongs to this category. The HOGHOF descriptor using histogram of optical flow and the STHOG (Spatio-temporal histogram of oriented gradient) proposed by the author both increase the discriminative nature of the usual HOG descriptor space, at the price, here again, of a highest computation cost.

Since the SVM classifier here generates a full detection map, with a degree of confidence associated to each detection point in the image, we can directly apply a Dynamic Neural Field (DNF) model on the HOG-SVM classification results. DNFs can can be viewed as a neuro-inspired and massively parallel ways of performing probabilistic recursive estimation at the image level (mesoscopic

scale in the visual cortex), instead of relying for instance on the Bayesian filtering and tracking of a few areas of interest (Kalman or particle filter). DNFs exploit the spatio-temporal coherence of the stimuli, relying on population coding (i.e. distributed representations relying on overlapping visual receptive fields) to perform robust inference. We can indeed expect SVM outputs to display coherence between subsequent frames (movement continuity) and between nearby locations on the image (SVM input being generated from overlapping groups of HOG cells). Incorporating an evolution model of the target dynamics, DNFs can even further improve detection reliability by reducing the FPR. The following paragraphs rely on the classical stationary equation for explanatory purpose, but the reader can refer to [**QG11**] for details on the predictive version using a linear model of movement.

The classical DNF equation models the dynamics of the mean potential of the neural population in cortical columns, forming 2D fields in the visual cortex. The potential at position $\vec{x} \in X$ and time t on such a field is defined by $u(\vec{x}, t)$, while the input stimulation of the field is set to $y(\vec{x}, t)$, corresponding to SVM outputs over the entire image for the frame at time t. A simplified version of the one-layer DNF equation gives:

$$\tau \frac{\partial u(\vec{x}, t)}{\partial t} = -u(\vec{x}, t) + c(\vec{x}, t) + y(\vec{x}, t) \tag{16}$$

where c is a lateral competition term leading to the selection/detection of targets, and defined by:

$$c(\vec{x}, t) = \int_{x' \in X} w(\vec{x}, \vec{x'}) \sigma(u(\vec{x'}, t)) d\vec{x'} \tag{17}$$

where σ is a non linear activation function (classically a sigmoid function), and $w(\vec{x}, \vec{x'})$ is the lateral connection weight function satisfying Eq. 18. Excitatory standard deviation a controls the expected size of the target, thus allowing for multi-scale tracking, while the inhibitory standard deviation b determines the minimal distance between acquired targets. A and B are respectively the amplitudes of the excitatory and inhibitory components of the kernel. They thus control/weight the influence of the lateral competition relatively to the noisy input data (i.e. here the outputs from the SVM). In probabilistic terms, they thus weight the prior distribution on target location with observations from the current frame.

$$w(\vec{x}, \vec{x'}) = Ae^{-\frac{|\vec{x} - \vec{x'}|^2}{a^2}} - Be^{-\frac{|\vec{x} - \vec{x'}|^2}{b^2}} \tag{18}$$

4. Distributed sensing

As previously stated in Chapter 1, CPS devices internally handle processing capabilities as long as the communication interfaces. In this Chapter, a survey of the event detection techniques have been presented in a local processing node. The straightforward extension considers the distributed environment, where the CPS is distinguished by the distributed sensing logic and the local processing and control logic. The term "distributed" assumes a deeper and more appropriate meaning, referring more to collaborative nature of the overall application logic, moving from the sensing layer to the processing layer in so far.

A distributed sensing system can be typically distinct as follows:

(1) *Centralized* Data are acquired by the node and conveyed to a central entity that executes the processing. Any processing is involved on the sensing layer nor collaboration between nodes. Although this solution makes available all the information in a single point, the deployment results not feasible for large scale, video-enabled installations.

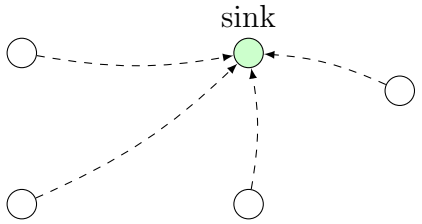


FIGURE 3.6. Centralized: node measurements are sent to a central processing entity (Sink).

(2) *Clustered* Instead of a central point, the aggregation is moved to cluster of nodes as closer as possible to the measurement points. This approach permits to balance the processing between "cluster-heads" yet considers a fixed hierarchical network configuration.
(3) *Distributed* The aggregation takes place directly on the sensing level, where data are captured. Although it reduces the communication requirements, the node itself has to be able to process its own measures and generate events to the network.

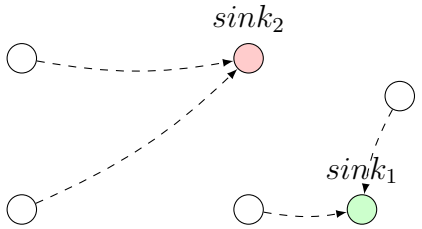


FIGURE 3.7. Decentralized: cluster heads are receiving data from external nodes. Each cluster head acts as a centralized sink for node that belongs to it.

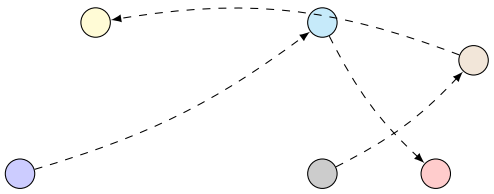


FIGURE 3.8. Distributed: Elaboration is performed by the nodes themselves. Coordination is required to aggregate consistent information.

In this manuscript, the distributed sensing paradigm is considered to address the application requirements as in Chapter 1. Indeed, such an approach conserves, at the same time, the high reconfigurability, scalability and robustness for a wide range of applications.

5. Conclusion

In this Chapter, the event detection methodology has been described. Instead of relying on raw data acquisitions, the proposed visual CPS architecture considers aggregated semantic features to understand the environment and its behaviour. However, the event detection paradigm requires dedicated computational resources and complex computer vision algorithms. A survey, yet not exhaustive, of the available state of the art event detection methodologies have been then introduced. Among them, we selected a feature-based machine learning approach with a spatio-temporal post-processing stage. The proposed approach, to the best of our knowledge, realizes the higher trade off between detection reliability and required computational resources.

CHAPTER 4

Heterogeneous SC architecture

In previous Chapters, formal approaches have been discussed. At first, an abstract model to describe a generic computing site has been proposed by applying the bigraph formalization. Such a model obviously relies on several architectural constraints that need to be kept under consideration while designing real-world implementations. The MoA approaches this issue by providing effective and reproducible metrics to evaluate the system performance, thus reducing the gap between the abstract representation and the physical realization. With the nowadays architecture complexities, the MoA aspect is more than necessary to address newer and yet not resolved issues.

Later, in Chapter 5, the event-based paradigm poses newer challenges to the processing architectures, requiring advanced processing capabilities within small and cost-contained SCN nodes. Event detection introduces timing and reliability constraints that need to be addressed by the node itself.

Smart cameras are indeed becoming an effective part of pervasive applications such as health-care [**CFK+14**, **PEK+15**, **Hos16**], vehicles tracking [**ZLY14**], people tracking [**KDM+14**] and pervasive computing [**CCT+15**]. Such applications require dynamic and reconfigurable architectures to rapidly adapt to changing conditions, e.g., scene illumination, object movement, unexpected event, where coordination with other nodes is also essential to manage resources and information [**SMS+14**]. Along with the previous mentioned open issues, these advanced processing capabilities require a newer, more flexible architecture paradigm.

1. Background

Despite the nowadays advancements in the computing technologies, the newer application requirements are growing faster than the increase of the computing and communication improvements so far. In recent surveys [**TPD15**,**TW17**], the

authors predicted the "End of Moore Law", where traditional computing architecture reaches their maximum performance per energy cost. The "Von Neumann" architecture was originally thought to be linear and sequentially executed, hence not addressed to distributed computing architectures.

Newer architectures concepts go beyond the Von Neumann paradigm towards natively concurrent and distributed computing platforms. Such architectures, commonly defined as "Heterogeneous architectures" promote performance efficiency and modularity by merging hardware and software in cooperative environments. Indeed, these hybrid infrastructures create improved processing designs that express the benefits and capabilities of programmable computing elements, working together seamlessly [**Geo12**]. In the latest years, several standards *de-facto* have been proposed by the biggest industry players to provide unified programming models that harmoniously integrate different computing technologies, e.g., Central Processing Unit (CPU), Graphic Processing Unit (GPU), FPGA, Application Specific Integrated Circuit (ASIC).

Several are the main goals:

- Removing the programmability barriers
- Reducing the communication latency
- Optimize the energy per processing cost
- Opening to wider application set

In [**Wm15**], the current trends for heterogeneous architecture are presented. The authors provide a deep perspective of several issues that going heterogeneous might uncover: parallel processing, memory model, concurrency and tool chains. In order to address them, the Heterogeneous System Architecture Foundation (HSAF) [**Fou**] is currently grouping the industry leaders to a greater integration of computing systems coming from different vendors.

In particular, such architectures are gaining interest in the broader Internet of Things (IoT) ecosystem, where processing capabilities and reconfigurability are becoming more and more important. With this respect, in [**RES+15**, **KVH16**, **SBH16**], the IoT ecosystem considers heterogeneity at the utmost importance for effective deployments. The authors survey the state-of-the-art IoT technologies and the perspective beyond them. Currently, the most promising technologies are based on CPU, ASIC and FPGA designs due to their target to power-aware, low-end devices.

When referring to the Smart Camera (SC) domain, many solutions have been proposed to tackle the computing and communication challenges. Regarding the CPU domain, Symmetric Multi Processings (SMPs) and Asymmetic Multi Processing (AMP) methods are commonly used to approach the heterogeneous design concept [**SHTE08**, **HAPP12**, **HLP13**]. However, ASIC and FPGA are currently shaping the edge of research to provide energy-efficient accelerators executing specific operations and algorithms. Indeed, they result in a better integration of communicating and processing elements mapped to a set of dedicated physical resources. In [**AKC+07**, **AKC+08**, **CCT+15**, **SSR+16**], ASICs deploy custom hardware designs to enable smart camera applications with VLSI technologies. Such custom implementations tailor specific video processing applications through dedicated hardware circuitry.

On the other side, FPGAs are becoming prevalent in several domains, such as signal processing, networking and cryptography [**TPD15**]. Such programmable architectures are more than electronic glue logic, as they were perceived in the past. In the last available evolutions, FPGAs host dedicated general control units, communication means and power management besides the "electronic sea" of reprogrammable arrays.

Nowadays, the programmable logic ("glue logic") is only but one part of what FPGA architectures are providing. Indeed, three programmable layers are involved: (i) I/O programming, (ii) hardware programming and (iii) software programming. At first, I/O programming defines the external connectivity capabilities and data input/output infrastructure. This involve dedicated hardware circuitry for DDR RAM, high speed I/O ports, communication transceivers, etc. Along with interface configuration, the I/O creates the subsystem to support the other architecture layers. The hardware programming then defines the modular architectures, where FPGA facilities are exploited to create hardware accelerators, tailored to specific tasks. Such modules, can be run-time configured to match newer application requirements.

Finally, the software programming manages the overall system configuration through Application Programming Interface (API) made available by the other layers. When referring to the latest FPGA designs, this upper level is handled with embedded CPU instances, either hard-coded or rather programmed within the FPGA.

The new era of FPGA is pushing the concept of Reconfigurable Computing (RC) [**TPD15**], where "spatially programmable architectures opened an unique design space" to integrate knowledge from different disciplines. Hardware and software reconfigurable devices are already deployed in WSN, such as in [**HSS+14**], where hardware and software routines are working side by side to optimize the processing while reducing the energy footprint. Further improvements have been proposed by [**LMA+12**] with a heterogeneous many-core architecture. The authors integrate DSP, FPGA and CPU technologies to provide the best performance with run-time remote requests for vision applications.

Heterogeneous solutions have been also proposed by [**KPL+10**, **KPdlTR11**, **HAPP12**, **KBN12**, **KBNH14**] as the next generation nodes for future Wireless Sensor Network (WSN). These solutions leverage hybrid CPU/FPGA platforms to enable transparent communication between hybrid networking functions, regardless of their implementation. Such proposals found support in the latest survey on the Smart Camera (SC) [**SMS+14**, **PEK+15**, **TW17**], where heterogeneous architecture are paving the way for the next computing revolution.

2. Our architecture

In Chapter 2, we first introduced a formalism able to describe a distributed processing system. Such a model involves several functional entities, called Operator (OP), and semantic constraints that specify the OP instantiations. In this section, we specify the relation between the OP semantic and its implementation by integrating the work presented in [**BMS+13**, **MSP+13**] and in [**MBQ+16**].

The operators categories, previously proposed in Chapter 2, are one-to-one mapped into the computing infrastructure as follows:

Processing operator. A processing operator is defined as a self-consistent processing operation implemented either on hardware or software. Hence, it can be implemented in any software language (e.g., C, C++) or any Hardware Description Language (HDL) (e.g., Verilog, VHDL). It hosts at least two ports, one input and one output, that define the data accesses. The operator is considered active when data are passing through its input port. The results are then available on the output port according to the relative computing latency.

Communication operator. The communication operator handles data structures and their proper redirection. It follows the concept of the Network on

Chip (NoC) [**HJK+00**] and manages data adapters and protocol handshakes according to the link-graph connections. According to the application specifications, it can be implemented by software routines or hardware modules. The resulting cost parameter considers the maximum throughput achievable and the operator capability to redirect the data flows.

Transducer operator. The transducer operator acts the external interface, converting external stimuli to data structures and viceversa. According its role, it might host only one output port or one input port, for sensor or actuator respectively. When referring to SCs, the transducer would probably handle the CMOS video sensor and provides the video stream through its output port. Since the physical video interface requires strict timing constraints and high throughput capabilities, this operator would probably be implemented as an hardware module. Whether needed, this operator also handles data buffering and data adapters.

In Fig. 4.1, the architecture proposed in [**MSP+13**] is shown. This architecture concept relies on hardware-oriented modules implemented in an embedded FPGA. Transducers are at the rightmost and leftmost sides of the architecture, while communication operators are placed vertically as crossbar switches for the processing operators. Data flows from the left side, where video sinks are placed, to the right through processing and communication blocks. The application manages the data connections (through the red accesses below) and the processing operator configuration (through the violet access on the top).

Such a linear architecture can be described as in Fig. 4.2 according to the link-graph methodology. The mapping to the link-graph is straightforward for processing and transducer, while the crossbar presents a double stage communication operators, where data adapters take places. Each processing site presents only one input port and our output port while C^A and C^B operators show one-to-many and many-to-one ports.

In Fig. 4.1, we indeed assumed that all operators reside within the same computing site. However, the model can be directly extended to heterogeneous distributed platform, where operators can be seamlessly integrated. This concept extends the architecture to a distributed network of computing sites, able

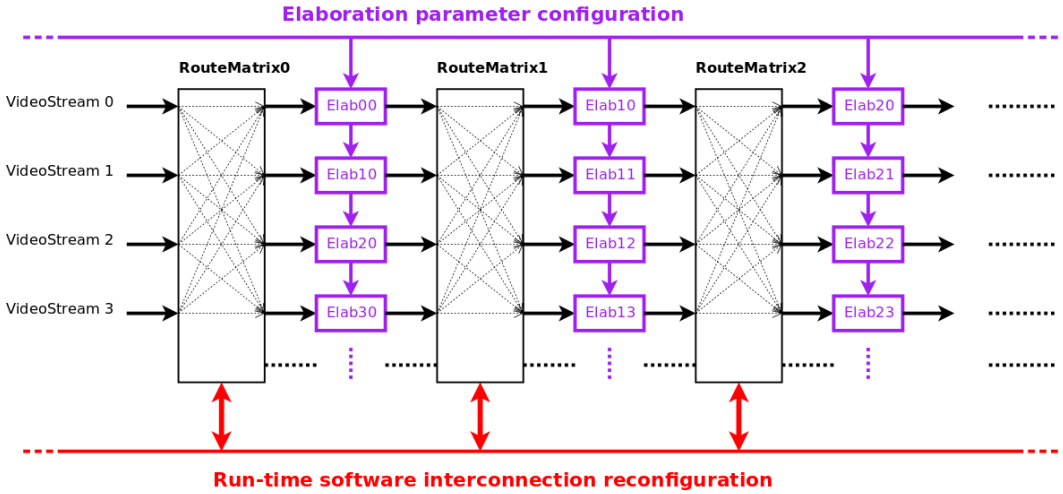


FIGURE 4.1. Configurable hardware architecture.

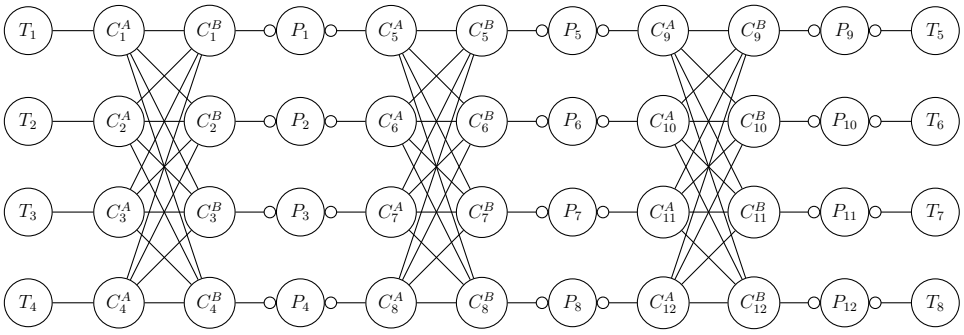


FIGURE 4.2. Link graph from Fig. 4.1.

to exchange either processed data or events following the application requirements. The interactions between nodes are regulated through the previously introduced distributed coordination model (Chapter 5) and operators are spread among the network nodes by leveraging heterogeneous implementations. As a result, operators in Fig. 4.2 are distributed over heterogeneous place graphs, where communication and processing operators are mapped according to the available platform sites.

Such an architecture benefits of the recent hybrid FPGA technologies, which embed general purpose CPU within the FPGA fabric. Available sites can both implement tightly coupled software and hardware routines to improve the system flexibility and the processing capabilities at the same pace. Different vendors, e.g., Intel Cyclone V SoC, Xilinx Zynq SoC, are nowadays integrating in the

same chip FPGA and ARM CPUs, interconnected through a sea of logic arrays. By leveraging the flexibility given by the CPU and the processing capabilities by the FPGA, these hybrid architectures are indeed opening newer possibilities, especially for distributed processing applications.

In envisioning heterogeneous SCs, the architecture in Fig. 4.1 is then implemented, where transducer, communication and processing operators are mapped according to an heterogeneous place graph.

Three main components are referred: the configuration manager, the communication interfaces and the hardware subsystem. The configuration component manages the place graph reconfiguration according to the application requirements, while the communication manager handles the link graph connections and the distributed interactions. The hardware subsystem implements the processing infrastructure and the physical communication mediums, realizing the flexible subsystem to support operators instances. These system components comply with the middle-ware presented in [**PSA$^+$15**], where software aims at uniforming the heterogeneous systems by hiding hardware-related features with unified application accesses. The development follows the hardware-software co-design technique [**Gup01**], where typically design proceeds by moving computationally intensive parts of the application from software to hardware accelerators, while keeping in software the parts having the lowest computational cost and/or the more dynamic behaviour.

3. Hardware platform

The proposed implementation has been validated on the Arrow SOC development board. This board embeds an Altera Cyclone V FPGA with up to 45k Arithmetic Logic Module (ALM) and 336 9x9 DSP multipliers. Within the FPGA fabric, a dual core ARM Cortex-A9 is deployed as a Hard Processor System (HPS) with dedicated Gigabit Ethernet and 1GB DDR3 RAM. The HPS and the FPGA are tightly coupled through the AMBA bus, which allows seamless communications between hardware and software routines. The processed images are both coming from an external 1.3 Megapixels Aptina MT9D112 CMOS sensor connected through the high speed HSMC connector, or rather redirected from the Gigabit port for dataset evaluation and bench marking. Other external interfaces include USB and external general purpose IO for expansions.

4. Application use-case

The considered application involves a network of heterogeneous devices capable of generating events and signaling others significant behaviour or dangerous situations. In this vision, we focused our attention in developing a reliable and flexible pedestrian detection system capable of interacting with other nodes. In here, the cameras are either intended to be placed on the roadside or either placed outside the vehicles, observing the road environment.

The proposed detection system, previously introduced in Chapter 3, is developed following the the hardware-software co-design technique [**Gup01**], where typically design proceeds by moving computationally intensive parts of the application from software to hardware accelerators, while keeping in software the parts having the lowest computational cost and/or the more dynamic behaviour.

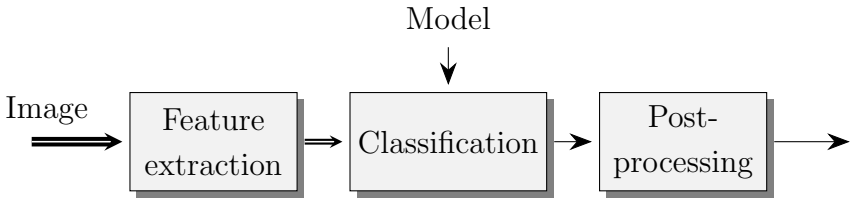
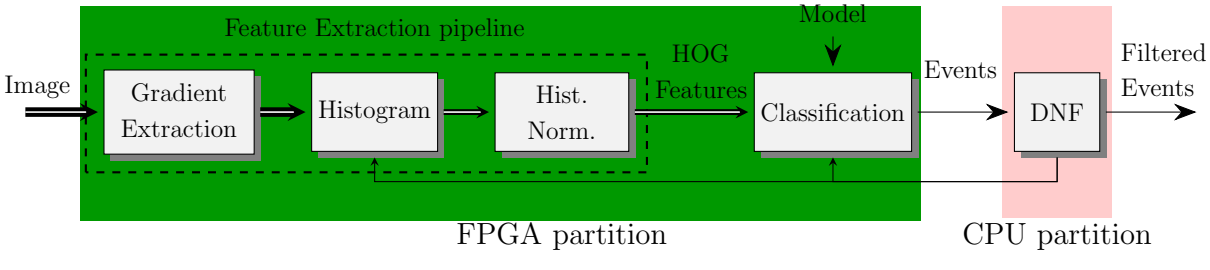


FIGURE 4.3. Processing pipeline for pedestrian detection.

Since the goal is the meet the real-time constraints, this partition is based on the computational cost of each algorithmic component. For this, we evaluated the required throughput (in MB/s) for each algorithmic component. The results are reported in Tab. 4.1, for a full-HD resolution (720p) at 30 fps. Not surprisingly, the highest numbers are associated to the first two steps, which respectively computes the gradient and accumulates the histograms of their directions. The related computations involve massive, regular data-parallelism. They are therefore well suited to FPGA implementation. The last step – Dynamic Neural Fields (DNF) filtering – exhibits a very moderate requirement in terms of data throughput but involves computations for which the accuracy is a key requirement. This step is therefore suited for a purely software CPU implementation. The two intermediate steps (normalization and SVM computation) have intermediate requirements both in terms of data throughput and accuracy. Because they require an tight coupling with the previous steps, with still significant throughput, they will benefit from an implementation on the FPGA.

Function	Required throughput	Optimal target
Gradient	27.6 MB/s	FPGA
Histogram	27.6 MB/s	FPGA
Normalization	3.45 MB/s	FPGA
SVM	3.45 MB/s	FPGA
DNF	0.34 MB/s	CPU

TABLE 4.1. Required data throughput (720p resolution, 30 fps, 8x8 cells, 8 bins) and proposed target architecture



;

FIGURE 4.4. Overview of the proposed hardware-software architecture.

The resulting proposal for the hardware-software heterogeneous architecture is sketched in Fig. 4.4. Mapping the three first steps on the same FPGA minimizes the risk of communication bottleneck between the hardware and software parts. On the other hand, having the post-filtering stage realized in software will make it easier to validate and export the pedestrian event detections.

4.1. Gradient computation. Since the highest throughput appears right after pixel acquisition, the performance of this first stage significantly affects the overall results. Here, computation of the gradient is performed using the *HOG-Dot* algorithm proposed in [**MBP+15**]. With this algorithm, each gradient is discretized by projecting it onto a set of predefined directions θ_k ($k = 0 \dots N_b - 1$). The k^{th} projection is given by :

$$\frac{\partial I}{\partial \hat{i_k}}(x,y) = \cos\theta_k \left[I(x+1,y) - I(x-1,y)\right] + \sin\theta_k \left[I(x,y+1) - I(x,y-1)\right] \quad (19)$$

where $\hat{i_k}$ is the unitary vector in direction θ_k.

The greatest projection then directly gives the closest gradient approximation in the discrete domain. The process is summarized in Fig. 4.7 (with $N_b = 8$), where ∇I_{Dot} here gives both to the discretized gradient orientation and the corresponding magnitude for each input pixel.

From an implementation point of view, this approach has two advantages. First, the N_b projections can be computed in parallel, thus significantly improving the overall throughput. Second, it involves only linear operations – and no square root neither arc tangent – and is therefore particularly suitable for hardware implementations. In fact, we have shown that, in this context, the use of linear approximations – compared to non-linear, floating point realizations – leads to an average error of less than 2% [**MBB+15**].

The hardware design is obtained following Eq. 19. The matrix convolution between the kernel coefficients and the 3×3 mask pixel is computed by leveraging on pipelined hardware module. Since these coefficients are constant for a given angle θ_k, the k^{th} gradient component can be evaluated by deploying two hardware multipliers and three adder/subtractor modules. In Fig. 4.5 the hardware circuitry for a single gradient projection is shown. More in detail, the input pixel is represented with 8bit while each coefficient value has been represented with 9bit two's complement.

Due to the internal pipeline structure, the module shown in Fig. 4.5 is capable of processing new input data every clock cycle to achieve the maximum throughput. In order to extend the gradient computation to all the available projections, N convolution modules are instantiated in parallel. As suggested by Fig. 4.7, the resulting hardware structure generates N parallel gradient projections. Each clock cycle, those resulting projections are compared each other to extract the maximum magnitude value. In Fig. 4.6 the hardware $argmax$ implementation is shown. This is implemented as $\log_2(N)$ comparative stages within a pipelined architecture.

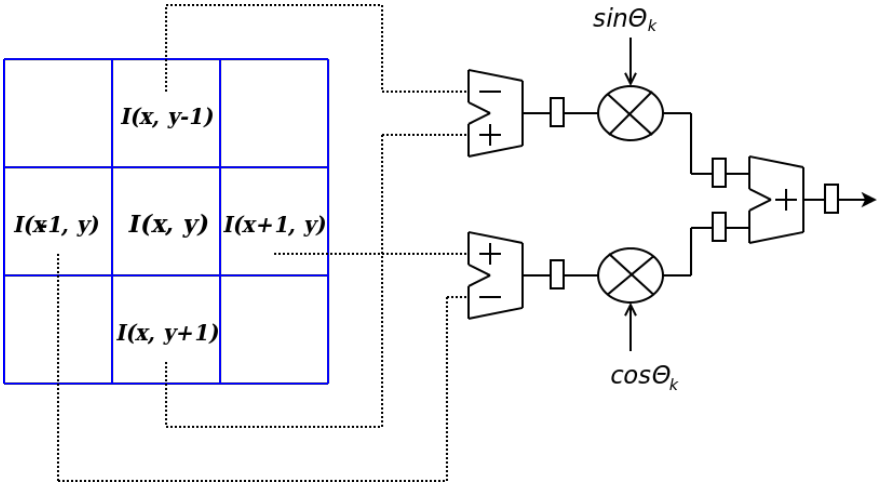


FIGURE 4.5. The hardware implementation for the i_k gradient component.

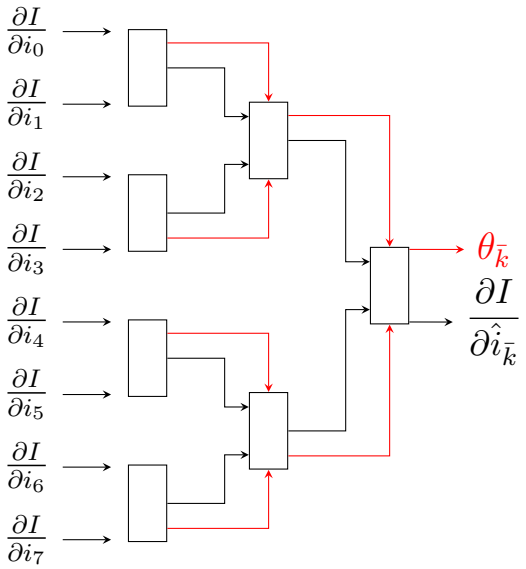


FIGURE 4.6. Pipelined *argmax* extraction ($N = 8$).

With the assumption of $N = 8$, the first stage is composed by four parallel comparators, which are fed by the computed gradient projections. Each comparator propagates to the next stage the temporary maximum projection and its orientation. In the presented example, the comparison ends at the third stage, where the closest gradient approximation is generated. The resulting couple $\frac{\partial I}{\partial \hat{i}_{\bar{k}}}$

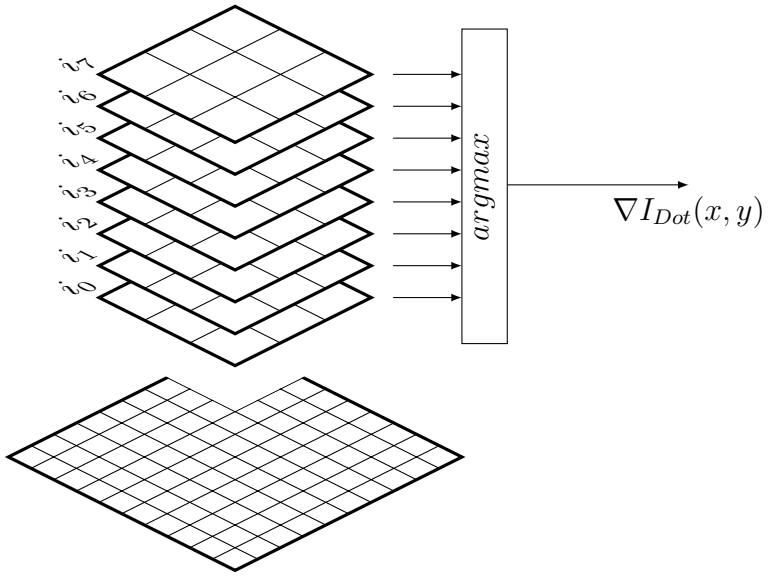


FIGURE 4.7. The HOG-Dot parallel implementation.

and $\theta_{\bar{k}}$ represents the closest spatial gradient approximation retrieved from the HOG-Dot method [**MBP+15**].

Histogram computation. Algorithmically speaking, this step is straightforward: it only consists in accumulating, in an array, each gradient orientation (the number N_b of discretized orientations therefore corresponds to the number of histogram bins). Nonetheless it raises challenging issues at the implementation level when applied to the data flow produced by the previous stage. Classically, histogram computation is decomposed in three successive steps: (i) address computation, (ii) memory read and addition (iii) memory write. With a synchronous implementation, the read-modify-write procedure therefore requires two clock cycles to handle each input data. The gradient computation stage, instead, produces one gradient direction per clock cycle. The architecture we used to solve this problem is depicted in Fig. 4.8.

It is a modified version of one initially presented in [**MSP+14**] and uses two distinct histogram modules (SubCell0 and SubCell1 in Fig. 4.8). Input data is directed alternately to one or the other of these modules, so that none of them requires more than one access to the memory at each clock cycle. At the end of each cell, the final histogram is obtained by simply adding the two sub-histograms.

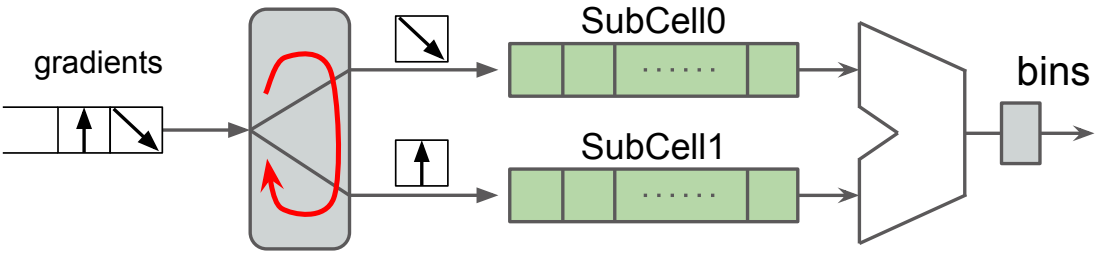


FIGURE 4.8. Internal histogram cell generation.

Histogram Normalization. As previously underlined, a normalization step is required to limit the sensitivity of the algorithm to the variations in global luminance. We have evaluated well-known normalization schemes for HOG in terms of detection performance. Results are given in [**MBB+15**], using the Receiver Operating Characteristics (ROC) metric – *i.e.* by plotting the True Positive Rate (TPR) with respect to the False Positive Rate (FPR). With no surprise, a lack of normalization leads to the poorest performance, with a maximum of 83% of TPR at 10% FPR. The three other assessed normalization schemes (L1-norm, L1-srqt and L2) give similar performances. We have chosen to implement the L1-norm because it offers the best trade-off between precision and implementation complexity. The L1-norm shows only a limited loss in precision with respect to L2-norm – from 95% to 93% of TPR both evaluated at 10% FPR. Moreover, L1-norm does not require the square root operation with respect to L2-norm but only a division.

Our implementation of the L1-norm computation step is sketched in Fig. 4.9. The normalization factor is here computed as the sum of the histogram. A pre normalization factor is applied to each value to limit the effect of quantification due to fixed-point encoding. This factor can be adjusted according to the required accuracy on the one hand and the output data range on the other hand.

In Fig. 4.9 a shift register (upper left) is used to buffer the histogram values belonging to the same cell so that each value can eventually be divided by the sum of values within his cell. This allows the normalization step to be carried out in a fully pipelined fashion. The division operation (right) is itself implemented using a pipelined architecture in order to maximize the global throughput.

$$\tilde{H}_i^b = \frac{H_i^b \cdot M}{\sum\limits_{i \in B} H_i^b} \quad , \forall i \in B, \forall b \in image \tag{20}$$

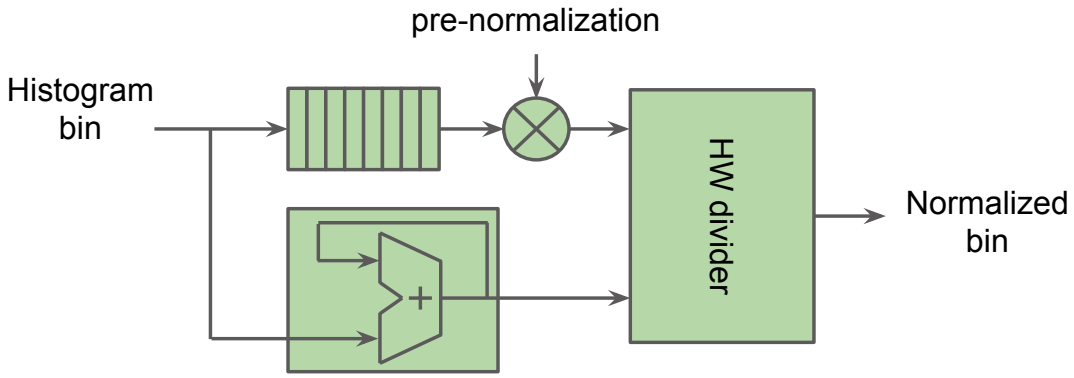
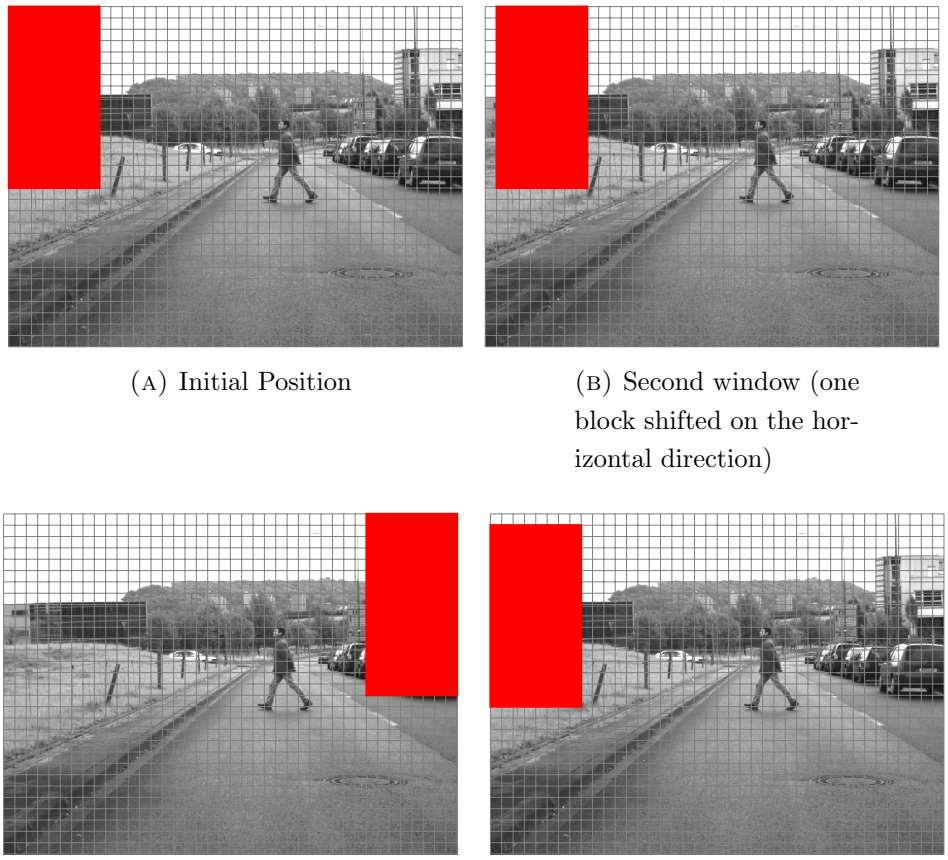


FIGURE 4.9. L1 norm module.

Support Vector Machine. As introduced in Eq. 15, the SVM classification is computed over image subsets, called *detection windows* (or simply *windows* in the sequel). Fig. 4.10, for example, shows a situation in which a detection window, here composed of 8 × 16 cells, is slided over the entire image in order to detect pedestrians.

In most implementations, this sliding mechanism is carried out sequentially, i.e., by processing one window after the other. This purely sequential approach is highly time consuming and, in practice practically precludes real-time performance to be obtained on general purpose CPUs. Our implementation circumvent this problem by leveraging the massive parallelism offered by FPGAs. The idea is to process the (overlapping) windows of a single row in parallel (e.g., Fig. 4.10 those located between positions a and c). For this, we introduce a dedicated accumulation unit called a *slice*. Each slice computes, in parallel, the dot product between the descriptors obtained on a set of neighboring windows and the model obtained by the offline training phase.

The dot product itself is obtained from the Eq. 13 with a linear kernel ($K(\mathbf{x_i}, \mathbf{x})$ is linear) and a binary classification problem. For each window:



(A) Initial Position

(B) Second window (one block shifted on the horizontal direction)

(C) Last window in the first row

(D) First window of the second row (one block shifted in the vertical direction)

FIGURE 4.10. Illustration of the sliding detection window

$$y_{n,m}(\mathbf{x}) = \sum_{\substack{1\leq i\leq 16\\ 1\leq j\leq 8}} \underline{w_{B}}_{i,j} \cdot \underline{x_{B}}_{n+i,m+j} + b \tag{21}$$

$$\text{with } n \in \{1, \cdots, N\} \text{ and } m \in \{1, \cdots, M\}$$

where $\underline{x_{B}}_{n,m}$ is the HOG descriptor value at coordinates (n, m) and $y_{n,m}$ is the corresponding SVM detection result. As introduced in Eq. 15, N and M respectively gives number of windows in the vertical and horizontal direction.

Since weight coefficients have constant position within the detection window, each slice has a closed-loop mechanism for hardware reuse. Indeed, the first weight coefficients $\underline{w_B}_{1,j}$ for a block $\underline{x_B}_{n,m}$ are also the first coefficients for the next block $\underline{x_B}_{n+1,m}$ and so on. This temporal relationship allows the transfer of the weights between slices, thus reducing the memory footprint and wiring complexity of the design.

Slices are first structured into horizontal machines managing the horizontal overlap and the model weights distribution. For a specific line of HOG descriptors at the n vertical coordinate, the hardware instance H_z (as in Fig. 4.11) computes all the horizontal stride windows as follows:

$$\underline{y}_{n,1..M} = \left\{ \sum_{i=n}^{n+15} \sum_{k=j \bmod 8}^{j \bmod 8+7} \underline{w_B}_{\alpha,k} \cdot \underline{x_B}_{i,j} \,\middle|\, \begin{array}{l} j \in \{1,...,M\} \\ \alpha = i - n + 1 \end{array} \right\} \tag{22}$$

The horizontal machine computes the SVM projections by addressing the $w_{B\alpha,k}$ and $x_{Bi,j}$ values. The indexes (i, j) identifies the cell coordinates within the image. The horizontal iterations, relative to windows and to cells within the window are referenced with j and k respectively. The H_z machine finishes when the last cell in the $n + 15^th$ cell row has been processed. In Fig. 4.11, the hardware architecture of the horizontal system is given for the generic line z. Here the SVM weights $\underline{w_B}_{\alpha,k}$ coming from the previous H_{z-1} flow as a carry chain among slices and then to the next H_{z+1} instance. In order to further minimize the number of concurrent read accesses to the memory containing the model descriptor, a circular chain of coefficients is deployed, allowing reuse of these coefficients between active H machines. Once the weight coefficients have been sent once, the machines internally exploits the temporal correlations without requiring any further memory access.

The same approach is applied to a vertical overlap machine, as illustrated in Fig. 4.12, which shows how 16 horizontal H modules – as defined in the previous paragraph – are instantiated to process in parallel 16 lines of a set of windows, with the corresponding SVM weights distributed in a loop. The role of the output multiplexer is to select the SVM result corresponding to the corresponding window line, following equation Eq. 23. In this equation, $y_{n,m}$ is the result associated to the window at coordinates (n, m) and 16 is the number of lines per window.

$$\underline{y}_{n,1\cdots M} = \{H_z\}, \text{ with } z = n \bmod 16 \tag{23}$$

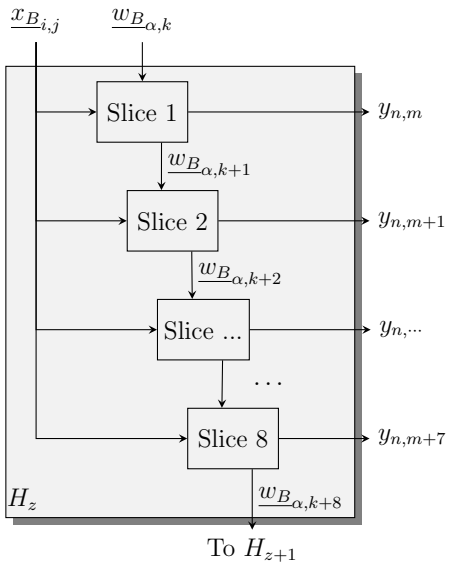


FIGURE 4.11. The horizontal module H_z

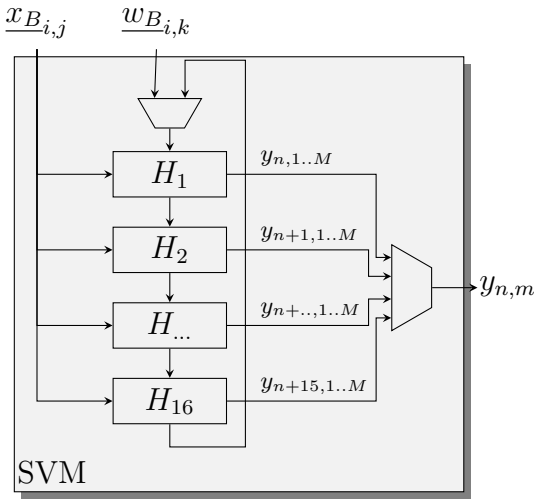


FIGURE 4.12. The complete SVM architecture

The final result is a parallel architecture capable of carrying out the SVM detections in a parallel manner.

Dynamic Neural Field. To make the DNF model simulation possible, discretization is performed in the spatial (grid lattice approximation) and temporal

domains (Euler integration scheme). Using a matrix formulation, Eq. 16 becomes:

$$U(t+dt) = \left(1 - \frac{dt}{\tau}\right) U(t) + \frac{dt}{\tau}\left(C + \overline{Y}(t)\right) \tag{24}$$

where $\overline{Y}$ is obtained by first thresholding Y (from Eq. 15), and then normalizing it in $[0, 1]$. This operation has the effect of setting all values below $-\mu$ to 0, where μ corresponds to a tolerance on the negative side of the classification separator. Increasing μ (especially above the SVM margin value) leads to retaining a larger amount of false positive results. Yet since DNF models are designed to deal with low signal-to-noise ratios and effectively filter false positives out (i.e. limited FPR), setting $\mu > 0$ actually leads to higher TPR. Nevertheless, and in order to make a fair comparison with the raw data in the result section, μ was set to 0. The spatial competition term C is then defined by:

$$C = W^{+} * \sigma(U(t)) - B \times \Sigma(U(t)) \tag{25}$$

where W^{+} is the matrix version of the excitatory part of the kernel function w (see Eq. 18), and $\Sigma(U(t))$ is the grand sum of the matrix $\sigma(U(t))$. This approximation is made possible by setting $b = +\infty$, focusing on a single pedestrian at a time, but also limiting the convolution to a reduced excitatory kernel of 5 $\times$ 9. Other parameters take the following values: $A = 1$, $B = 0.3$, $a = 0.25$, assuming an affine transform is applied to the input space (matrix $\overline{Y}$ of 72 $\times$ 44) so that the pedestrian detection window (8 $\times$ 16 HOG cells) corresponds to a unit square.

In order to evaluate the heterogeneous architecture proposed in the previous section, a prototype implementation has been developed. This prototype is built upon an Arrow SOC development board. This board embeds an Altera Cyclone V FPGA with up to 45k Arithmetic Logic Unit (ALM) and 336 9x9 DSP multipliers. Within the FPGA fabric, a dual core ARM Cortex-A9 is deployed as a Hard Processor System (HPS). The HPS and the FPGA part are tightly coupled through the AMBA bus, which allows interoperable communications between hardware and software routines.

Two issues are addressed by the evaluation process. Performance on the one hand and reliability on the other hand. Performance is assessed in terms of processing time and hardware resource usage (Sec. 4.1). Reliability is assessed in terms of detection accuracy with respect to state of art existing realizations with different evaluation conditions (Sec. 4.1).

Performance evaluation. Tab. 4.2 gives the details of the FPGA implementation as obtained using the Altera Quartus II 13.1 toolchain using the Arrow SOC boad as described in 3. The reported values give, for each algorithmic step the resource usage in terms of ALM, RAM and DSP, the latency (L) and the estimated maximum clock frequency.

The complete processing pipeline takes only 18% of the available hardware resources. The required amount of memory (for line buffering and internal data storage), in particular, is less than 200 Kbit and largely fits in the available on-chip FPGA RAM.

Concerning latency, it is defined as the time between the consumption of the first input data and the production of the first output result. This time directly depends on the depth of the processing pipeline. In our case, in does not depend on the value of the input data and is therefore constant.

	ALMs	RAM	DSPs	L	f_{MAX}
Gradient	464	16 Kb	8	10	107
Histogram	244	131 Kb	0	5120	145
Norm	400	8 K	0	16	83.9
SVM	7300	42 Kb	128	10240	130

TABLE 4.2. FPGA hardware results for VGA resolution. The f_{MAX} and Latency (L) are expressed as MHz and clock cycles respectively.

In order to demonstrate the potential benefits of our heterogeneous architecture, the results given above have been compared to those obtained with two other architectures: a commercial Intel i7-870 workstation and an embedded ARM HPS subsystem. For the latter, a state of the art OpenCV implementation of the HOG and SVM algorithms was used. Results are given in Tab. 4.3. Because the notion of clock frequency is not directly applicable to software implementations, comparison is performed in terms of processing time for a single frame. It must be noticed, however, that this notion of processing time has an interpretation which ultimately depends on the platform. For a purely software implementation, it is generally defined as the interval separating the end of the input frame acquisition

and the end of the processing of this frame by the CPU ((b)-(d) in Fig. 4.13). For a FPGA-based implementation, processing actually occurs in parallel with acquisition, in a fully pipelined fashion. Processing time, in this case, can be evaluated by measuring the latency of the output result (interval (b)-(c) or (a)-0 in Fig. 4.13).

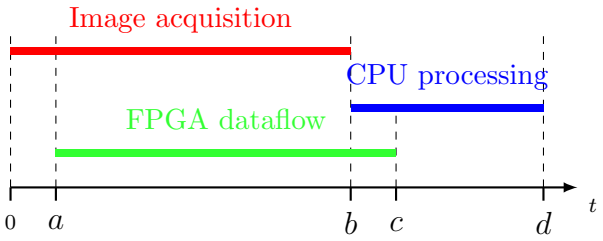


FIGURE 4.13. Performance comparison.

In our case, the maximum clock frequency and latencies reported in Tab. 4.2 lead to a global latency of 184 μs (sum of each module latency). Compared to the I7-870 and ARM implementations this corresponds to a reduction of 400x and 4630x respectively. In terms of FPS, the multiplication factor, with respect to these same implementations, are of 20x and 218x respectively. However, these numbers should be interpreted keeping in mind that the reported processing times for the CPU implementations ((b)-(d) in Fig. 4.13) do not take into account the image acquisition time. This might lead to a significant overestimation of the performance results. For the FPGA implementation instead, the fps value is experimentally measured and only depends to the input pixel rate. The maximum clock frequency, which in our case, gives the maximum pixel rate, is 83.9 MHz. For VGA resolution, this gives a maximum throughput of 273 fps.

We also compared our implementation to other FPGA-based implementations of the HOG-SVM application. In [**KSH+09**] a solution with 10 parallel processing elements is described. The reported maximum clock frequency is 127.49 MHz but with a maximum frame rate of 30 fps at VGA resolution. Another heterogeneous HOG architecture has been proposed in [**BRH13**]. But, due to the processing bottleneck caused by the PCIe communication interface between the FPGA and the rest of the system, the histogram computation is 10x time slower than in our solution (657 μs instead of 64.4 μs). In [**HSH+13**] the PCIe limitation is significantly reduced to 150 μs with Full-HD images sent through a

	i7 870	ARM	FPGA
Function	time (μs)	time (μs)	time (μs)
Gradient	38200	425756	0.12
Histogram			61.4
Normalization			0.2
SVM	36000	376566	122
Overall HOG	74200	802322	184
fps	13.5	1.25	273

TABLE 4.3. Comparison of processing times at VGA resolution.

GigaE connection but still represents a significant bottleneck, hindering global system performance for the CPU-FPGA proposed solution. In [**BKDB10**] an hybrid FPGA-CPU-GPU solution is described. Using FPGA modules leads to a processing time of 311 μs only for gradient and histogram computations. The normalization step and the Gaussian kernel SVM are then handled by an external CPU-GPU device though multiple DMA instances.

Compared to those realizations, our architecture minimizes bottlenecks by privileging on-chip communications and limiting inter-chip communications to low bandwidth data. Implementing all the most throughput demanding modules in the FPGA, in particular, allows these throughputs to be limited only by the critical path in the associated circuitry and not by the bandwith of the hardware-software interface. The proposed hybrid approach minimizes also the external interfaces requirements thus reducing the global system complexity.

Detection accuracy. In this section we evaluate the final accuracy of our detection application by comparing it to other state of the art similar applications. The selected alternatives are those described in [**HSH$^+$13**], [**KSH$^+$09**], [**BKDB10**].

Evaluation is carried out in two steps : first with the well-known INRIA dataset, second with real-world VGA image sequences taken from the Daimler Pedestrian Benchmark Dataset [**EG09**]. In both cases, the classifiers were

trained with positive and negative image samples from the INRIA pedestrian dataset [**DT05**], with an offline training phase performed with the SVMLight framework [**Joa99**].

Evaluation using the INRIA dataset. Results for this first evaluation step are given in Tab. 4.4. For fair comparisons, all the considered techniques are trained with the INRIA train dataset [**DT05**] and verified with the INRIA test subset (1126 positives and 453 negatives 64×128 windows) and no post-SVM filtering was used. For [**BKDB10**] only the gradient and histogram parts have been ported inside the FPGA.

All implementations exhibit comparable results in terms of TPR (the most critical aspect for pedestrian detection applications). Our implementation exhibit a slightly higher FPR (4%). This effect has been tracked down to the use of the L1-norm approximation and can be eliminated by adding a post-SVM filtering stage (as shown in the next section).

	[**HSH$^+$13**]	[**KSH$^+$09**]	[**BKDB10**]	Our
TPR	93 %	95 %	95.4 %	94.9 %
FPR	1.0 %	1.0 %	0.1 %	5.5 %

TABLE 4.4. Comparisons with state of the art FPGA HOG implementations on the INRIA train/test dataset.

.

Evaluation using the real world video sequences. Obviously, and as recalled in [**BOHS14**], this step is mandatory for any pedestrian detection system aiming at realistic applications, such as those integrated in ADAS. We therefore have evaluated our application using a dataset consisting in sequences of VGA images obtained with a camera mounted on an outside moving vehicle (Daimler pedestrian video sequence [**EG09**]). This kind of data is indeed the ideal benchmark for an ADAS prototype because experimental evaluation here nearly perfectly matches the actual exploitation conditions (with different intrinsic camera parameters, extreme luminance variation and rapid scene changes). Moreover since most of the false positive detections are not temporally coherent, they can be removed using a DNF-based post-processing step (as showed below). Training

	CPU	CPU T_p=2.5s		CPU T_p=250ms		Our FPGA+ARM	
	[**BOHS14**]	[**EG09**]		[**EG09**]			
	Raw	Raw	Tracking	Raw	Tracking	Raw	DNF
TPR	41.5	64.3	68.7	67.4	79.1	91.0	80.3
FPR	-	1.17	0.14	14.3	1.3	17.6	1.0
AUC	-	-	-	-	-	0.785	0.891

TABLE 4.5. Detection performance for cross-dataset implementations. Area Under ROC Curve (AUC) $\in [0, 1]$, True Positive Rate (TPR) and False Positive Rate (FPR) in percents.

of the models, however, is still be carried out using the INRIA dataset. This so-called *cross-dataset* approach has been proposed in [**EG09**] and [**BOHS14**].

Results are given in Tab. 4.5 using three metrics : the TPR/FPR metrics on the one hand and the AUC metric on the other hand. The former is the same as previously defined. The latter has been proposed in [**DWSP12**] in order to evaluate the overlap between the detected and the "ground truth" bounding boxes (BB_{dt} and BB_{gt} in the sequel) when multiple detection windows are placed around a pedestrian silhouette. For this, a overlaping factor is computed as

$$a = \frac{Area(BB_{dt} \cap BB_{gt})}{Area(BB_{dt} \cup BB_{gt})} \tag{26}$$

and compared to given threshold a_0. In the PASCAL challenge [**EVGW$^+$10**], a_0 is set to 0.5 and we adopt it for comparisons as well. Each detected bounding box is evaluated with respect to the manually labelled ground truth. The higher the overlap, the higher the detection confidence results. If the overlap does not exceed the threshold value, the bounding box is labeled as false detection. The final detection performance is finally expressed as the Area Under Curve (AUC) and the TPR/FPR with respect to the "ground truth" bounding box BB_{gt}. Receiver Operating characteristic (ROC) curves are generated by varying the threshold value and then used to compute the AUC metric. The AUC metrics is 1 is for the ground truth, 0 for a fully wrong result and 0.5 for chance level. Because it does not depend on the detection threshold (b parameter in Eq. 21), this metric is

particularly useful to estimate the detection gain with respect to brute FPR/TPR values.

In Tab. 4.5, the *Raw* columns correspond to the zero-crossing output of the SVM with respect to b (e.g., positive values as pedestrians), while Tracking and DNF results are relative to the post-processing method applied. Note that [**BOHS14**] reports only TPR number, without any additional implementation detail. Results in [**EG09**] are given for two implementations based on different constraints on a 2.66 GHz Intel processor. Note also that these two implementations differ in the size of the detection grid applied to the image and thus exhibit distinct processing times (with respect to trajectory-based detections [**GM07**]).

First of all, Tab. 4.5 clearly shows that overall performances, compared to those given in Tab. 4.4, are significantly reduced. This is an unavoidable consequence of using a *cross-dataset*-based approach, in which the detector is confronted to unexpected features. Moreover, raw SVM detections are considerably less accurate than filtered ones. The work reported in [**EG09**] shows that temporal tracking allows a reduction of the FPR from 1.17% to 0.14% and from 14.3%0 to 1.3% with 2.5 s and 250 ms implementations respectively and an increase of the TPR from 64.3% to 68.7% and from 67.4% to 79.1% respectively. With respect to the TPR, our solution outperforms other implementations regardless of the post processing stage, and leads to a reduced miss rate for pedestrians in a real deployment scenario.

As already shown in Tab. 4.4, our architecture presents an higher raw FPR with respect to other methods. Indeed, with cross-dataset validation the raw FPR reaches 17.6% with respect to the 5.5% achieved with INRIA dataset. This rate is also higher than the other considered cross-dataset validations, which limits the FPR to 1.17% and 14.3% respectively. Nevertheless, our solution performs significantly better for raw TPR values, reaching 91% of accuracy with respect to 41.5%, 64.3% and 67.4% of the considered comparison in Tab. 4.5.

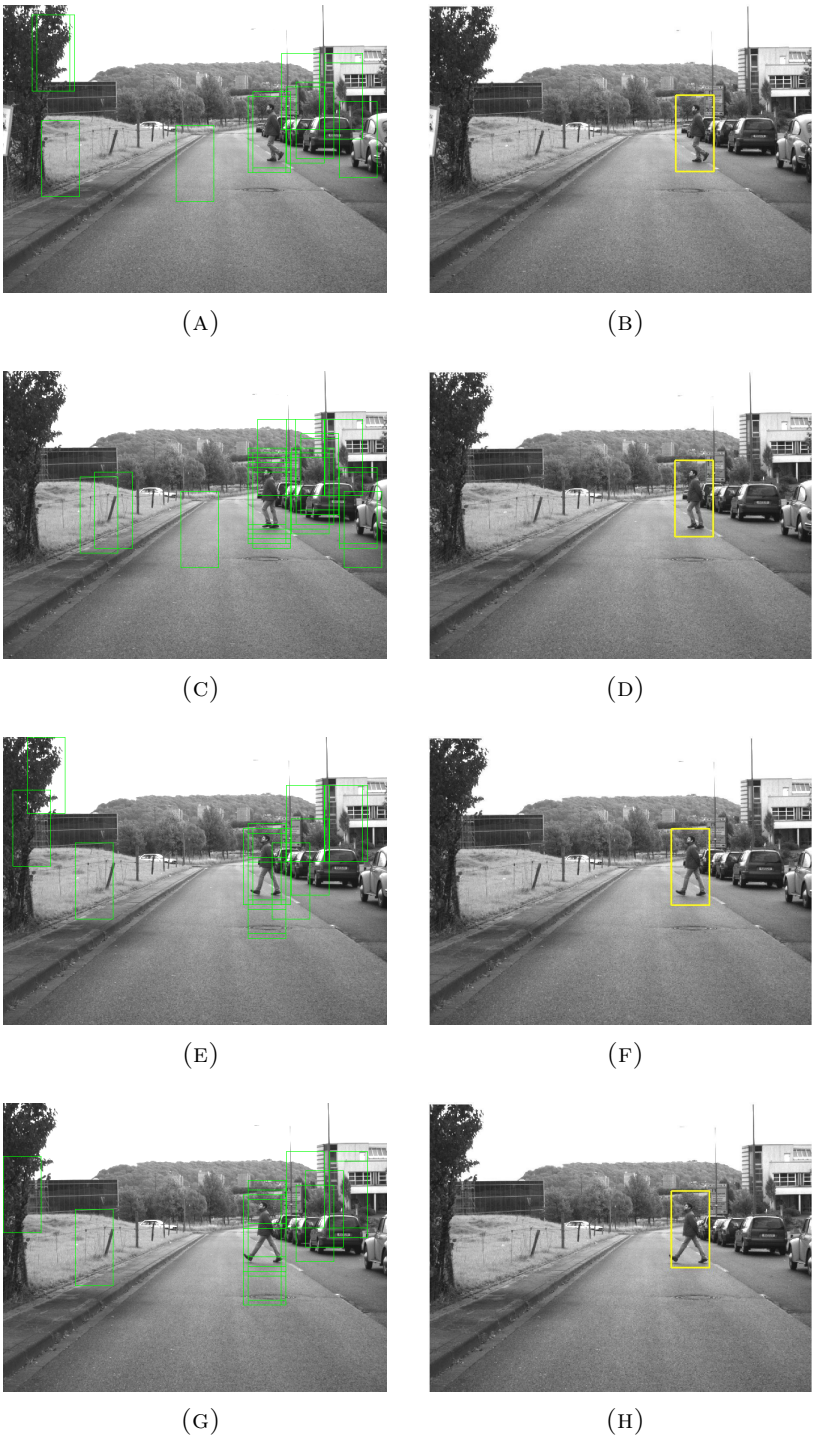
Tab. 4.5 also shows the positive effect of the post-processing stages, either based on tracking or on DNF. For our implementation, in particular, the DNF step drastically reduces the number of false detections, reducing the FPR from 17.6% to 1.0% (82% improvement). This drastic reduction in FPR comes at a price though. As shown in Tab. 4.4, the TPR value results are indeed slightly

reduced by the DNF post-processing (-10%). This effect is due to some misled false detection suppression, which are particularly important in case of not completely overlapping detection windows. Despite the TPR degradation, the AUC metric shows that DNF has improved the system response. Indeed, the FPR improvements largely compensates for the TPR degradation, and the resulting SVM+DNF combination globally improves the detection reliability. Fig. 4.14 illustrates visually the impact of the DNF-based post-processing stage on the detection results. The left column (Fig.4.14a, 4.14c, 4.14e, 4.14g) shows the raw SVM detections provided by the hardware system on four consecutive frames (from Daimler pedestrian dataset). The right column (Fig. 4.14b, 4.14d, 4.14f, 4.14h) gives to the detections after the DNF filtering stage has been applied. As suggested by results in Tab. 4.5, the FPR value has been considerably reduced with a stable and consistent target detection.

The experimental results, obtained in realistic conditions, show that this architecture achieves satisfactory event detection performance for Smart City deloyments. In particular, they show the benefit of a co-design partitioning methodology, in which a tight coupling of hardware and software processing modules leads both to a gain in performance and an improvement in detection reliability. They also demonstrate the benefits of a spatio-temporal post processing algorithm in terms of detection reliability, by reducing the amount of false positive detections.

5. Conclusion

In this chapter, a bio-inspired heterogeneous architecture for pedestrian detection has been presented. This architecture is suitable for smart camera deployment with local processing capabilities. Such FPGA-based smart cameras are indeed close to the concept of visual CPS introduced in Chap. 1. Our proposed methodology leverages the FPGA parallel processing capabilities coupled with high level software routines. Experimental results, obtained in realistic conditions, show that this architecture achieves satisfactory detection performance for the target applications. The evaluations show in particular the benefit of a co-design partitioning methodology, in which a tight coupling of hardware and software processing modules leads both to a gain in performance and an improvement in detection reliability. They also demonstrate the benefits of a spatio-temporal DNF-based post-filtering step in terms of detection reliability, by reducing the amount of false positive detections.



(A) (B) (C) (D) (E) (F) (G) (H)

FIGURE 4.14. Raw SVM detections (left) and DNF filtering results (right) computed by the proposed architecture. Subsequent frames were chosen to display the spatiotemporal coherence of the SVM results.

CHAPTER 5

Coordination model

Events are locally generated then globally analyzed and classified to recover higher level inferences according to an aggregation model. This model can be seen as a correlation map of the environment, based on observations. Once the model has been defined, each local event can be classified by inferring correlations with its neighbor events, thus enabling event prediction properties.

As long as the network remains relatively small, the event model can be statically defined as a function of the node positioning. Consider a Smart City scenario where a set of SCs triggers events in response to vehicles movements, and witness a road accident. Based on cameras placement, the event semantics can be empirically estimated due to the well-constrained scenario (e.g., vehicles directions, speed). However, along with the network growth or in case of variability (e.g., roundabout, pedestrian movements), retrieving the semantics and relationships between nodes will rapidly become more complex. Moreover, due to the often noisy input and detection glitches (e.g., false alarms), a single sensor measure alone might not be reliable enough to guarantee correct higher level inferences.

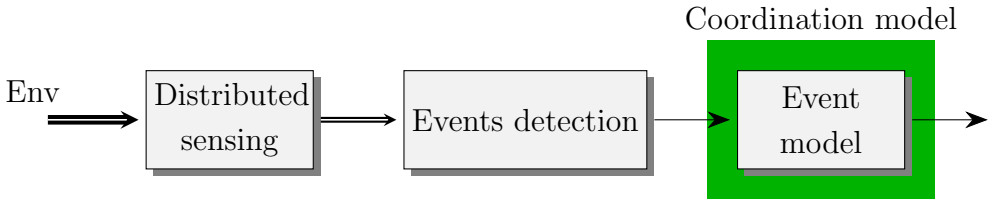


FIGURE 5.1. A distributed sensing environment formalization.

Indeed, in designing distributed systems – and in system of systems applications as well [**WS15**] – a multi-layer formalization is needed. In this respect, we refer to the three layers shown in Fig. 3.1. Since in the environment a distributed sensing is performed by each sensor autonomously, a cloud of local events is then generated. These local events are eventually analyzed and classified to

recover higher level events and eventually performing actuation. The nodes become themselves part of the control loop, by modifying the actively interacting with the environment [**GEN$^+$16**].

Although intuitive and straightforward, such a collaborative schema adds complexity to the system description and formalization. Since each node conserves autonomy, the system is intrinsically composed by concurrent entities, which are communicating each other by means of neighborhood relationship graphs (see Chapter 2). These relation-based connections are triggered upon environmental conditions (e.g., shared events between nodes, unexpected conditions happened) thus evolving during the system up-time.

This Chapter addresses the following questions: How autonomously coordinated networks can be analytically described? Which are the interaction models behind communications? In the following, the relative literature is considered and then a novel coordination model [**MBKQ$^+$16**] is proposed and evaluated through numerical simulations.

1. Related work

Autonomous coordination founds its origin when researches started to study the "subtle interaction between knowledge, action and communication in distributed system" [**HF85**]. The notion is indeed deeply found of the concept of UbiComp that have been initially presented in Chapter 1 and nowadays still does not have an unique answer. In literature, coordination models pose three main objectives: (i) derive the network topology, (ii) enforce the event aggregation among nodes and (iii) exploits correlated detection measures to improve the system reliability.

5.1. Topology discovery. In Chapter 2, the *computing site* concept describes an autonomous entity that involves the temporal and spatial processing space. Such an entity performs measures to understand the surrounding environment with respect to its own point of view that obviously differs from others according to the physical positioning. Therefore, each node heavily relies of the ability to establish its relative position with respect to other nodes involved.

Retrieving the relative positions is particularly of interest among nodes which show correlation between measures, namely among whose are observing the same

scene from different point of views. In literature, many solutions have been proposed to address this issue. In particular, they might be divided in two category: communication-based discovery and semantic-based discovery.

In [**SRB01**], a general framework to retrieve relative nodes positions belongs to the first category. Relative locations are computed from sparsely deployed anchor nodes, whose coordinates are a-priori fixed. The algorithm indeed relies on radio range measurements, which are iteratively executed to sequentially refine the localization. Although it proposes an effective localization approach, it relies on fixed and centralized radio anchor, which are supposed to be visible by the node population.

Similar method has been proposed in [**GYJW10**] for distributed vehicles mobile ad-hoc networks. On top of radio measurements, the authors include also a prediction step to make routing forward looking. Indeed, when mobile nodes are involved, the topology control becomes essential to provide *cognition capability* to evaluate how long the link will be stable over time. This is particularly of interest when dealing with dynamic network topologies (according to Chapter 2 definitions), where network estimation should be carried out periodically.

In the following papers, the authors tackle the *cognition capability* with more formal approaches. In [**Gri03**] and [**GC14**], the topology discovery is described as a way of revealing *spatial auto-correlation*. The authors give the following definition; "data values that are not independent but rather are tied together in overlapping subsets within a given geographic landscape. Relative location, commonly expressed in terms of geographic closeness, partly determines the spatial autocorrelation". Although the definition is applied to satellite imagery, it describes how spatial correlation between measures can directly imply relative location estimation.

In this sense, correlated spatial measurements are also exploited in WSNs. For instance, in [**NPSM14**] the authors applied the spatial auto-correlation analysis to an arbitrarily deployed sensor nodes to recover the physical topology only by local measurements. The concept is further extended to heterogeneous measure sources (e.g., high-quality sources, binary threshold values), which makes the approach more general.

In [**VBDO$^+$14**] the semantic-based category is introduced. Instead of relying on radio or protocol-related positioning, the authors introduces the concept

of *event correlation*, referred as *spatial event correlation*. This concept reformulates the *cognition capability* concept and introduces also metric to optimize the communication routing to reduce power consumption. The authors proposed dYnamic and scalablE tree Aware of Spatial correlaTion (YEAST), a spatial-aware network protocol optimized to retrieve and exploit spatial correlations between measures.

The concept of spatial event correlations are particularly of interest when approaching video-enabled devices, where visual features can be enforced to provide localization through image understanding.

In [**ST12**], an algorithm to automatically recover cameras position through visible/infrared light emission is presented. It involves a direct localization method, by adding to each camera node a blinking LED with a specific and unique blinking code. By using the orientation data coming from an internal accelerometer, the node computes its own relative position with respect to blinking-IDs within its field-of-view. The geometric approximations are locally computed and bring to a relative network topology. Even though it only addresses indoor deployments, the spatial correlation mechanism is somewhat interesting for outdoor environment.

In [**GJNC$^+$14**] a more complex event source is considered. Although a simple and lightweight change-detection algorithm has been applied, the authors propose an indoor multi-camera based system which is capable an a-priori unknown number of people. The system relies on a centralized cooperation, while processing is completely devoted to the sensor nodes. The central sink acts as a fusion center, where the tracking phase takes place. Besides the spatial correlations, cameras are also providing special key-features, e.g., speed, height, that reinforce the event identifications between cameras (re-identification). Although the system has been designed for completely observable environment (e.g., indoor location with multiple camera views), the event source characterization would definitely be of use in outdoor deployments.

More formal approach has been proposed in [**ES12**], where a distributed tracking application is envisaged. The network is modeled with unidirectional links and connection weighted as a finite graph. The network evolution follows node attitude to communicate with a set of neighbours according to a mathematical formulation through adjacency matrices. An adjacency matrix is a square matrix where the elements indicate whether pairs of nodes are connected or not in the connection graph. This formal methodology is also suited to integrate other

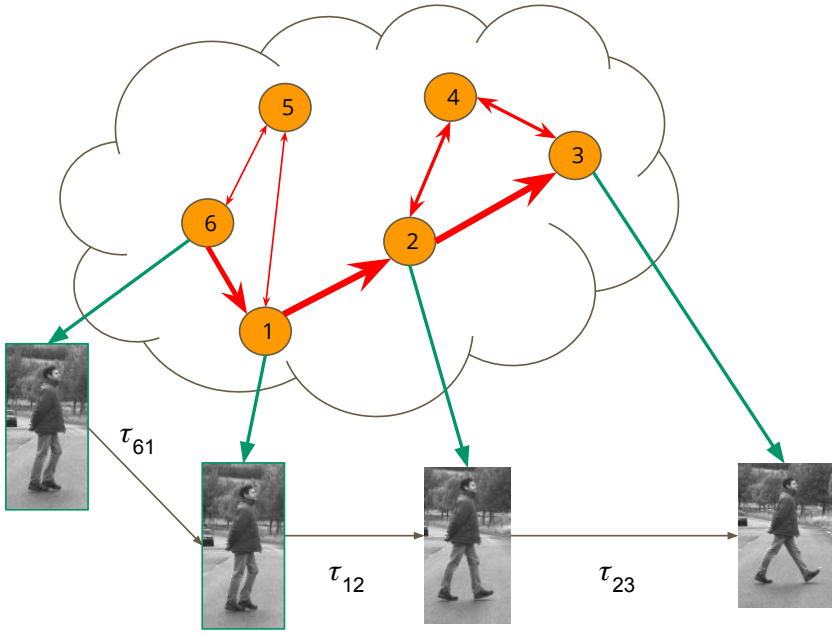


FIGURE 5.2. Local event fusion.

environmental details, e.g., furniture, relative position, to improve the tracking performance. Such formulation eases the camera installation since it avoids intrinsic and extrinsic camera calibrations and a-priori motion models for indoor tracking applications.

5.2. Event aggregation. Once the spatial signature of the environment has been retrieved, nodes are made aware of their positioning with respect to others. However, the model structure still lacks a way to globally understand the events. It is not practical (and feasible, though) to assume that events are continuously and fully observable by the network, hence the need of an event correlation description. As in Fig. 5.2, cameras C_6, C_1, C_2, C_3 show temporal event relations that might infer semantic environment properties. The event aggregation methodologies aim at providing higher level semantic to pattern of specific events. In the illustration, the higher event might refer to "a pedestrian is just crossed the road". At the same time, event aggregation strategies might also trigger anomalies, where expected event patterns are abruptly changing with respect to the expected evolution.

In following works, the authors focus their efforts on event aggregation methods for sensing applications. In literature, the event aggregation model has manifold definitions [**FPF17**]. Consensus formation, distributed estimation, data fusion, data association and collaborative acquisitions are, among all, the commonly used. While retrieving spatial information might be seen as static or aperiodic operation, such aggregations are run-time performed and require synchronization, interaction and knowledge.

A clear *consensus* definition can be found in [**OSFM07**], where is the achievement of "an agreement regarding certain interest that depends on the state of all agents". This definition obviously refers to a network of dynamic agents, where the communication rules between agents are defined by a *consensus algorithm*.

The consensus agreement indeed refers to many different problems, apparently not related, which show similar approaches. In the distributed sensing applications, event aggregations are indeed presented in biology [**ARABL17**], control of mobile nodes [**OSJ12**], distributed tracking [**WL14**], camera handoffs [**SQ14**], multi-camera tracking [**WWW14**], data mining [**FPF17**] and semantic ontology [**SMG09**, **MDN16**].

Most of the previous application use distributed sensing systems, e.g., WSN, SCN, deployed to understand and analyze the environment. In [**ARABL17**], the authors proposed a novel environmental monitoring solution for water supply supervision. Given the potential network size, a central data aggregation would not be practical. Moreover, given the link unreliability, the system needs to safely assure data correctness, through multiple observations. The algorithm has been further described in [**ARBL16**], where the stochastic nature of real-world interferences and fading has been considered. The authors propose an optimal state estimation based on the Kalman filter approach. Such methodology, also considered in [**OSFM07**, **OSJ12**], predicts the next state evolution as a linear system as follows:

$$x_{k+1} = Fx_k + w_k \quad \text{with } x_k \in \mathbb{R}^M \tag{27}$$

where x_k is the M-dimension state vector, x_{k+1} represents the next system state, F the $M \times M$ control matrix, and w_k the expected measure noise, assumed zero-mean Gaussian $\mathbb{N}(0, \sigma_w^2)$. The environment is measured through N observations, thus generating the z_k N-dimension measure vector. The relation between x_k

and z_k is:

$$z_k = Hx_k + v_k \quad \text{with } z \in \mathbb{R}^N \tag{28}$$

The H represents the observation space, where x_k projections provide the z_k measures. The v_k adds a Gaussian zero-mean observation noise $\mathbb{N}(0, \sigma_v^2)$, intrinsically part of the measure. With the assumptions of independent, white and Gaussian v_k and w_k noises, According to the Kalman filter theory [**WB95**] the a-posteriori state prediction $\hat{x}_{k+1}$ can be evaluated as:

$$\hat{x}_{k+1} = \hat{x}_k + G_k(z_k - H\hat{x}_k) \tag{29}$$

where $\hat{x}_{k+1}$ is computed as a linear combination of $\hat{x}_k$ and and the difference between the measures z_k and the predicted measure $H\hat{x}_k$. The $N \times M$ matrix G_k is defined as the *kalman gain* which minimize the a-posteriori estimate error:

$$e_k = x_k - \hat{x}_k \tag{30}$$

which ultimately assess the efficiency of the filter predictions. Whenever the e_k approaches zero, the measure z_k is trusted because a-posteriori validated. This approach indeed relies on the Bayes' rule, where the a-priori probability of an event is related to priors event measurements.

Other event aggregation methodologies involve auction based schemes for event aggregation and data fusion [**WW14**]. With respect to the prediction-based methods, such methodologies rely on concurrent agents, where data aggregation passes through nodes negotiations. Each node applies an opportunistic strategy, e.g., reduce workload, reduce power consumption, and a decision function to whether accept or not interactions. This is an highly active research segment, and many different approaches have been recently proposed [**HHCP08**, **ELC+13**, **ELYR14**, **SQ14**].

In [**SQ14**], an auction based mechanism is implemented to evaluate hand-offs between neighbours camera sensors. The application is clearly directed to event tracking, where multiple target might be present at the same time. Nodes are potentially assigned to different target tracking, thus they need to collaborate when targets overlap the respective field of view. The negotiation with neighborhood nodes starts when the node detects that the target is about to leave the field of view. In this case, other nodes might claim the hand-off based on the best observation match (e.g., auction). If the hand-off has success, the

processing task moves to the leading node. Similar approach yet different policy is implemented in [**ELC$^+$13**, **ELYR14**]. Here the authors still proposed an auction-based aggregation method, but nodes are competing each other to claim the hand-off whether they consider the auction "useful". The usefulness depends on how the balance between incoming and outgoing hand-offs would be modified by the current auction.

Recently, context sensitive data aggregation approaches have been proposed as ontological reasoning. With this respect, the main goal is to provide a formal way of setting up logical rules to semantically aggregate events together. A deep survey of the available has been recently published [**HLES17**] and notable implementations are [**PMDPVdW12**, **vRGG$^+$14**, **TLGB14**, **PA15**] and [**NHB04**, **SMG09**, **MDN16**] especially for SCNs. In general, the ontology definition aim at providing logical reasoning mechanisms to describe and model autonomous concurrent entities. In [**PMDPVdW12**] the ontology is applied to create a metadata standard, that might be used to structure semantic video elements available on the web. A semantic metadata is an abstract representation of image properties that reveal additional information, e.g., object class, color, speed, for higher level analysis. The metadata concept is further extended in [**vRGG$^+$14**], with formal requirements for event aggregations in a video surveillance application. In particular, the authors use the SoS (see Chapter 2) approach to describe the application as composed a set of requirements (e.g., coverage, features, events). By combining requirements together, the system can reveal higher-level events over situational patterns. For instance, this methodology can describe the occurrence of the event pattern – bag abandoned, crowded place, nobody around – as a possible threat event, which might trigger an alarm action.

Ontology semantic has been also applied to the context of SCN. Differently from the previous examples, in CPS the semantic mechanism resides within the network nodes instead of a global, omniscient entity. In [**PA15**], a spatial ontology is applied to safety-critical collision prediction test case. The spatial structure defines the physical constraints and objects trajectories as the dynamic term in the system. A traffic intersection is therefore described as the interaction between the static component "intersection" and dynamic components "vehicles". Both components have their own spatial and temporal features which define whether

a collision might occur (e.g., same spatial predictions at the same time). Approaching the SCN domain, such a methodology have been applied to model event for video surveillance, content-based video indexing and video annotation. The work in [**NHB04**], proposes a formal language (VERL) to describe and represent specific visual event in SCN. In particular, the ontology language permits to build and represent event occurrence through an aggregation of properties and relations of all relevant entities at a given moment in time. In [**SMG09**], the syntactic event representation is enforced to improve video analysis result. The authors propose a two-layer ontology, where scene ontology and the system ontology have been distinguished. The vision application results then in a combination of the scene ontology requirements, e.g., indoor surveillance, smart room surveillance, with the system ontology description, e.g., capabilities, resources, status. This distinction has been further developed in recent papers [**MDN16**], where the ontology is divided into three main components: sensing, environment and application. Although the sensing might be strictly dependent on the environment semantic, the authors also consider system reconfiguration driven by the application requirements.

2. Our distributed coordination model

As previously introduced, the coordination model is intrinsically based on interactions between computing sites. These interactions are modeled as unidirected communication links between nodes and can on-the-fly established if needed. The importance of interactions relies on the intuitive idea of collaboration: since event detections are affected by noise, by collecting multiple event observations the resulting system accuracy is increased.

Nonetheless, not all interactions are equally important: weighted connections between nodes are indeed commonplace in network based real-world problems. The concept of weighted connection topologies is a well-known method to model connection patterns for autonomous agent-based networks. In bio-inspired networks [**DA10**], similar approaches have been observed in the swarm intelligence [**BDT99**]. This is based on the observation of collective behaviours (e.g., ants movements) of of decentralized and self-organized systems such as ant colonies, swarms of bees or birds. Also latent brain activities, measured through functional Magnetic Resonance Imaging (fMRI), shows weighted connections patterns observing the neuronal response. This led the authors [**FKBR14**] to a

Markov-based linear model – called Dynamic Causal Model (DCM) – to analyse and evaluate the connectivity based on observed neuronal responses. [**SHK+07**]. The same principle can be found in [**VFPC15**], where a weighted Markov chain is deployed to predict the spreading process of epidemic disease.

Our proposed distributed coordination model [**MBKQ+16**], [**BKMQB16**], [**BKME+16**] addresses the aforementioned issues by leveraging self-organizing node interactions. Similarly to [**ES12**], an augmented adjacency matrix formalization has been developed for SCN deployments. In particular, our model addresses the following goal: (i) make the system adapt to changing environment conditions and to unpredicted events, (ii) improve the detection reliability by aggregating conform events, (iii) improve the system robustness against node malfunction [**OSFM07**] and detection glitches.

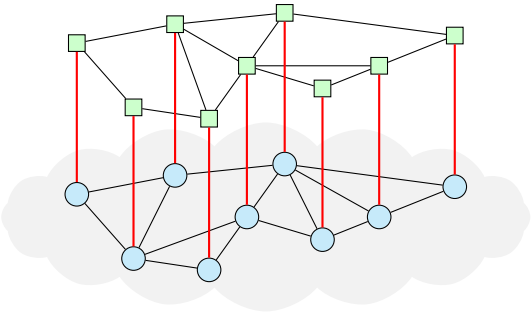


FIGURE 5.3. The system architecture.

The proposed distributed coordination model describes the functional connectivity between entities, whilst does not consider topological connectivity. In Fig. 5.3 the system connectivity architecture layers are shown. Indeed, the cloud in the bottom part illustrates the physical topology of nodes, where they are placed in the environment and how they physically communicate. This layer ultimately refers to the locality of place graph properties, previously explained in Chapter 2. On top of it, the coordination model defines which are the meaning interactions between nodes according to the event correlations. The links among green nodes in Fig. 5.3 indeed represents the existence of relations among regardless of their localization.

5.1. Adjacency Matrix. The adjacency matrices are well-known means of representing real networks and they are currently used to describe modern network protocols [**New03**, **New08**]. In our approach, we define as adjacency

matrix a $N \times N$ square matrix and in its basic form, describe the existence or not of a connection between nodes. Each element can either $\{0, 1\}$ and stands for an unidirectional link between node-i and node-j, with $i, j = (1, ..., N)$, as an edge in a finite graph. In Eq. 31, the matrix A describes a graph of five nodes ($N = 5$) and a defined connection pattern.

Although the adjacency matrix has only binary information, e.g., existence or not of a connection, already carries several information useful for later use. For instance, since each node has at least one connection (inbound or outbound), the underlying connection graph results connected. It means that each node adds information to the graph, intended as a global observing entity.

$$A_N = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{31}$$

However, the binary representation does not take into account the connection weight. Obviously when dealing with real networks [**New08**], this simplistic representation is not sufficient though. Each element summarizes the evidence gathered over time about consistent event observations by a pair of nodes. Given a set of events occurred within the network, nodes that are showing relevant event correlations establish preferential communications with a set of neighbours. The resulting augmented adjacency matrix is called as transition matrix, where each element weights the relative connection importance.

A transition matrix is then defined as stochastic matrix if its entries are non-negative real number p_{ij}, with $p_{ij} \in \mathbb{R}(0, 1]$, representing a probability of transition. Moreover, a stochastic matrix is can be right stochastic, if the sum of each row is up to 1, left stochastic, if the sum of each column is up to 1 or doubly stochastic in case both cases are true. Stochastic matrices are particularly of interest to describe a Markov chain over a finite space. Given M a stochastic matrix of dimension $N \times N$, p_{ij} an element in M and $\underline{s}$ the state vector, it describes the probability of moving from state s_i to s_j as follows:

$$\underline{s}' = M \cdot \underline{s} \rightarrow s_j = p_{ij} \cdot s_i \tag{32}$$

If $\underline{s}$ and M are stochastic, the result $\underline{s}'$ is still stochastic.

5.2. Scenario overview. In the considered scenario measures are autonomously performed by each node over its well-behaved local surroundings. The system environment results then in a manifold space composed by aggregating overlapped local measurement spaces. Each node is defined with the identifier C_n and its relative state s_n. The state s_n represents the node state in the network and its contribution to the distributed coordination.

The node state vector at time k is defined as:

$$\underline{s}^k = \{s_1, \dots, s_N\}^k \tag{33}$$

where N is the number of nodes at time k.

Moreover, each node describes its surroundings by defining a *visibility* range, namely the maximum distance within the target is detected by the node itself. On first approximation omnidirectional visibility spaces and uniform detection ranges are considered. This limit is initially introduced to simplify the mathematical formalism where further investigations are considered as future perspectives. The targets are moving events which are defined by their signatures, e.g., visual features, positions, speeds, trajectories. The target vector is defined as:

$$\underline{x}^k = \{x_1, \dots, x_M\}^k \tag{34}$$

where each component represents a target signature and M the number of targets that appear at time k. The causal evolution of $\underline{x}^k$ is described as:

$$\underline{x}' = G^k(\underline{x}^k) \tag{35}$$

where $G^k(\cdot)$ is a Markov chain process of the vector $\underline{x}^k$ and x' represents the target at interval $k+1$. In particular, G^k is a memory-less stochastic process with unobservable [1] states (as known as Markov assumption). As a consequence, the behaviour of the x_m with $m \in M$ target at time k can be evaluated as the probability distribution of the previous $k-1$ sample rather than the complete set of the target samples $\underline{x} = \{x^{k-1}, x^{k-2}, \dots\}$.

The targets are observed by the nodes through measures (if targets reside in the visibility range). Each measure is an approximation of the target features with respect to a specific node at one specific time interval. The observation

[1]The notion of observablility refers to a state that can be measured through its external appeareances. In this case, observable states are those that belongs to targets within one or more visibility ranges.

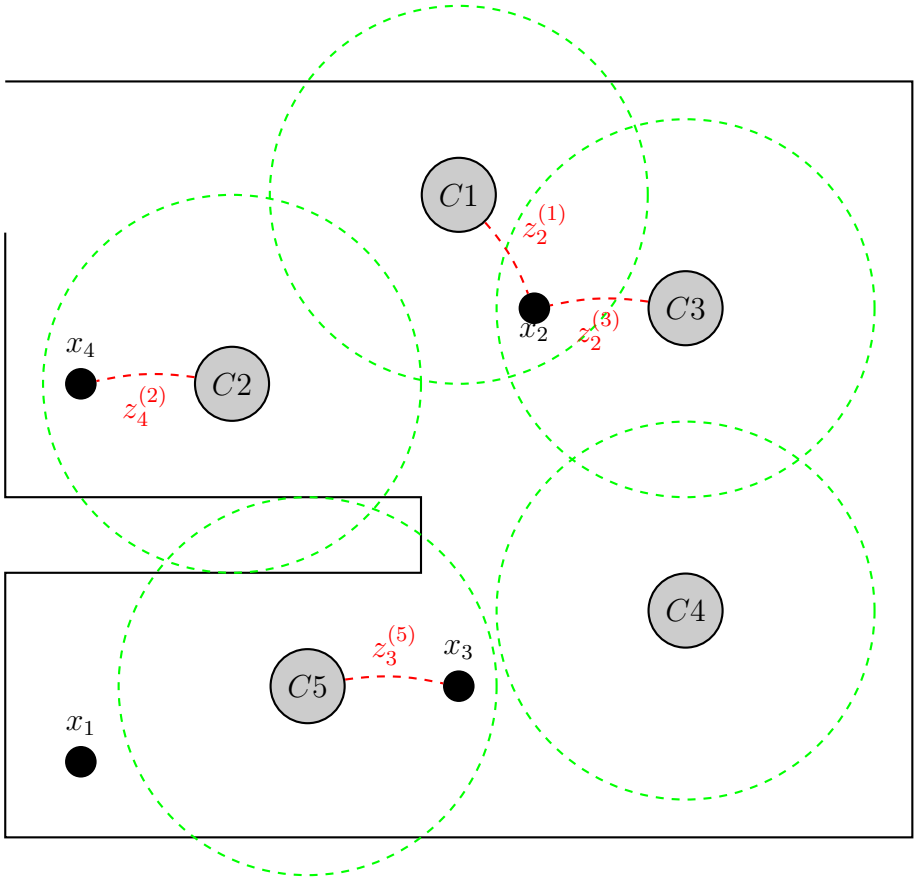


FIGURE 5.4. An instance of the system environment space, with visibility ranges as dashed circles. The environment is delimited by walls that also include physical constraints to the environment.

vector is therefore defined as follows:

$$\underline{z}^k = \{z_1, \ldots, z_M\}^k \tag{36}$$

where M still considers the available targets at time k. Since each node performs its own measures, multiple appearances of $\underline{x}^k$ could exist. In particular, seen from the generic node C_n the observation vector is defined as:

$$z_i^n = C_n\{x_i\} \tag{37}$$

where the C_n represents the $n-th$ node that processes the x_i target.

For instance, in Fig. 5.4 a system environment space is depicted. Here five nodes are instantiated $C1, \cdots, C5$ with their states $s_1, \cdots, s_5$ respectively. With this respect, targets $x_1, \cdots, x_4$ are arbitrarily placed and are moving following a

deterministic trajectory. The $z_4^{(2)}$ measure refers to the x_4 target being detected inside the visibility area of $C2$. Since x_2 belongs to the $C3$ and $C1$ visibility areas, double target measures are then retrieved. Due to the noisy environment and to the different spatial conditions, these measures are only ideally equivalents while they present some degrees of correlation. The example in Fig. 5.4 also considers physical constraints, such as walls, to better represent a real world setup, where nodes interactions are clearly influenced by the environment structure. Indeed, neighbor nodes, such as $C2$ and $C5$, might not show event correlations due to the external target constraint. The distributed coordination model, is then defined as:

$$\underline{s}^{k+1} = F^k(k, \underline{s}^k, \underline{z}^k) \tag{38}$$

where $F^k(\cdot)$ represents the state transition function. The state vector $\underline{s}^{k+1}$ at time sample $k+1$ becomes then a function of the previous state vector $\underline{s}^k$ and of the performed measures $\underline{z}^k$. Through local observations, the aim of the coordination is to reinforce the nodes interactions by approximating the G^k function with the a-posteriori evaluation F^k. In the next section, the structure of F^k is analyzed by proposing a novel methodology. In Eq. 39 the F^k function is expressed as the conditional probability with respect to the target state $\underline{x}^k$.

$$F^k \propto p(\underline{s}^k | \underline{z}^k) \tag{39}$$

Once the G^k approximation has been evaluated, F^k represents a stochastic model of the behaviour of $\underline{x}^k$ over discrete time k.

5.3. Model formalization. According to the Eq. 38, our discrete time model is formalized as a function of it previous state $\underline{s}^k$ and the input stimuli vector applied to the system $\underline{z}^k$. Assuming a total derivative of F, in Eq. 40 the dependencies in s and z are expressed.

$$\frac{dF}{dt} = \frac{\partial F}{\partial s}\frac{ds}{dt} + \frac{\partial F}{\partial z}\frac{dz}{dt} + \frac{\partial F}{\partial t}\frac{dt}{dt} \tag{40}$$

where

$$A = \frac{\partial F}{\partial s}|_z \quad \text{and} \quad B = \frac{\partial F}{\partial z}|_s \quad \text{and} \quad C = \frac{\partial F}{\partial t} \tag{41}$$

Finally, in Eq. 42 the system model equations are expressed. The vector notation is derived from the Eq. 38 and makes it suitable to multiple node instances. The scalar values A, B, C as in Eq. 40 become square matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and s^{k+1}

the state vector $\underline{s}^{k+1}$. The Eq. 42 also expresses the temporal dependence of the $\mathbf{A}^{k+1}$ term as the combination of the $\mathbf{A}$ and $\mathbf{C}$ of the previous iteration.

$$\begin{aligned} \underline{s}^{k+1} &= \mathbf{A}\underline{s} + \mathbf{B}\underline{z} \\ \mathbf{A}^{k+1} &= \mathbf{f}(\mathbf{A}, \mathbf{C}) \end{aligned} \tag{42}$$

The system dynamic $\underline{s}^{k+1}$ is driven by the input measures $\underline{z}$ in two separate contributions: (i) the response to direct stimuli through $\mathbf{B}$ matrix or (ii) the induced interactions according to the $\mathbf{C}$ matrix (as Eq. 41). The $\mathbf{A}$ matrix results from the Bayes assumption, where the next state depends on the a-posteriori probability of interaction. The matrix $\mathbf{B}$ instead represents observation space and the source term in the system. Finally, the dynamic and time-changing term $\mathbf{C}$ adds to the state matrix the induced connectivity as results of interaction pattern in the environment (it creates/removes connections coupling). In the following, the $\mathbf{A}, \mathbf{B}, \mathbf{C}$ contributions are considered to evaluate their effect on the state transition function F^k.

5.4. Stochastic connectivity. According to Eq. 42, the $\mathbf{A}$ matrix represents the next-state transition probability from the vector state $\underline{s}$. This is modeled as a stochastic adjacency matrix and contains $N \times N$ non-negative entries for every possible state transition. For a given state s_n related to the $n - th$ node, the interactions with the others can be expressed as a linear combination of the row elements of $\mathbf{A}$. These interactions are described as unidirected graphs, where connections are expressed as edges and nodes as vertex. The existence of interaction is revealed by a non-zero probability p_{ij} between the source node i and the target node j.

For instance, assume that Fig. 5.5 represents the connectivity graph of the system in Fig. 5.4. A connection exists from C_3 towards C_4 but no interaction is defined between C_2 and C_5 though. The existence of an interaction is due to the previous measured activities (e.g., events correlation) between nodes. This concept, borrowed from the biology [**SHK+07**] and social network [**VAA04**] studies, allows to represent distributed networks as functional connectivity between entities. An active communication is therefore defined by A as a probability distribution and it is not related to the physical network topology – e.g., Received Signal Strength Indicator (RSSI). For instance, far apart nodes with low RSSI unlikely

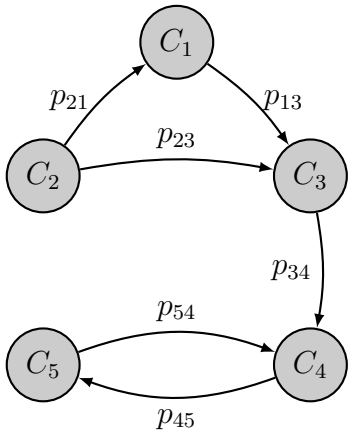


FIGURE 5.5. A connectivity graph example based on Fig. 5.4.

measures correlated events unless considering complex environment configuration. On the other side, nodes with high signal strengths might not automatically showing correlation between their measures. Thus, signal structure features can not be used to assess event correlations but they might rather considered as a complementary information to the model.

According to Eq. 42, A is a right stochastic matrix, with each row summing to 1. This property induces that the state transitions are balanced between the outgoing transitions (p_{ij} with $i \neq j$) and the steady state p_{ii} one. The lower p_{ii}, the higher is the importance of interactions. This results in a trade-off between steady state and transitions which depends on the importance given to communications rather than on the measures available in the node itself. In Eq. 43 the network connections shown in Fig. 5.5 are described.

$$A = \begin{pmatrix} p_{11} & 0 & p_{13} & 0 & 0 \\ p_{21} & p_{22} & p_{23} & 0 & 0 \\ 0 & 0 & p_{33} & p_{34} & 0 \\ 0 & 0 & 0 & p_{44} & p_{45} \\ 0 & 0 & 0 & p_{54} & p_{55} \end{pmatrix} \tag{43}$$

At system initialization, the relevant communication patterns between nodes is not usually known. The stochastic matrix A is therefore initialized to uniformly distributed probabilities. This means that each interaction is equally important and any distinction between them is to be later inferred. Thus, according to the

Eq. 42 and in absence of the source term, the system is in a steady state (the state vector $\underline{s}$ is null).

Through observations relevant correlations are collected and used to update the stochastic model of A. This event gathering phase (as in Fig. 3.1) is an on-line process that continuously measures and adapts the system response towards the likelihood expectation.

5.5. Observation matrix. While temporal event correlation drives the stochastic and the induced connectivity matrices, the spatial correlations are considered for measures. According to Eq. 44, the observation matrix B represents the observation space, as composed by the multiple measures of the target vector $\underline{z}$ performed by the N nodes.

$$\mathbf{B} = \{z_m^n\}^k \quad n \in \{1, N\}, \quad m \in \{1, M\} \tag{44}$$

In Fig. 5.6 the relationship between the observation matrix and the target vectors is shown. Since physical systems are unlikely fully observable, the B is considered as a sparse matrix. A zeroed column i means that the target x_i is not visible from the system therefore the relative measures are not considered.

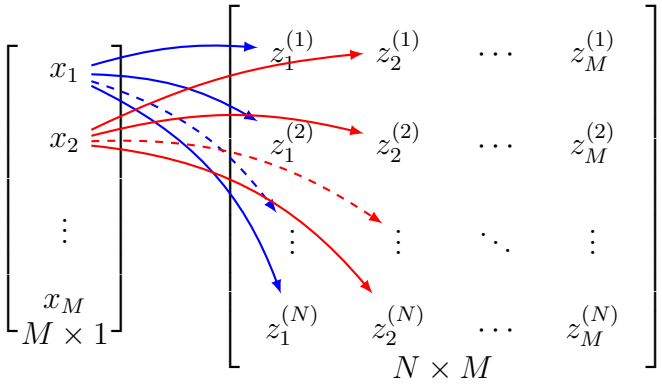


FIGURE 5.6. The relationship between the target vector $\underline{x}$ and the observation matrix B.

Each element of B represents how much two measures are correlated within the same temporal interval k.

This usually results in a geometrical evaluation between visibility ranges. The way of deriving it actually depends on the physical phenomenon under measure.

For instance, in SCNs the spatial correlations are related the cameras' field of views intersections and can be evaluated using affine region detectors [**MTS$^+$05**] or the Simultaneous Localization And Mapping (SLAM) technique [**SSC90**]. For proprioceptive sensors, e.g. vibration, temperature, spatial correlation is rather evaluated with correlation indexes, such as Moran's I [**Mor50**].

5.6. Induced connectivity. Given a set of events occurred within the network, nodes that are showing relevant event correlations are establishing active communications among them. Each node is modeled as an autonomous agent which tries to learn the optimal neighbor selection policy from its history of interaction. Since the environment has been defined as a Markov process (according to G^k), events correlations are meaningful information to reveal repetitive patterns of behaviour. In Eq. 45, a simplified version of the Q-Learning algorithm [**Wat89**] has been used to evaluate the induced connectivity as the $\mathbf{C}$ term.

$$\mathbf{C} = \alpha(\mathbf{R} - \mathbf{A}) \tag{45}$$

where $\alpha \in \mathbb{R}[0, 1)$ is the learning rate, R is the reward observed after validating the prediction. The elements of $\mathbf{C}$ can be computed as follows:

$$c_{ij} = \alpha(r_{ij} - p_{ij}) \tag{46}$$

where p_{ij} is the likelihood that the event (measure of a target) evolves from the camera i to the camera j (see Eq. 43). The reward term r_{ij} is related to the temporal correlation analysis between available measures from camera i to j. This factor is evaluated using a Gaussian probability distribution $\mathcal{N}_{ij}(\tau_{m,ij}, \sigma^2_{ij})$ centred at the delay time expectation $\tau_{m,ij}$ between nodes. For instance, if a target x_1 is detected from the node $C1$ and then from the node $C2$, the delay time as τ_{12} can be measured. If this target is periodically detected between the nodes, an expected delay time $\tau_{m,12}$ can be also estimated. Since then, in order to evaluate whether nodes are temporally correlated or not, the measured τ_{12} is compared to the $\tau_{m,12}$ through the time delay distribution. The closer is τ_{12} to the model average, the higher correlation probability results. Then higher correlation turns to higher r_{12} reward. By iteratively updating the τ_m averages and giving the likelihood probability of correlation, the nodes strengthen their coordination

in case of correlated event and vice versa. In Eq. 47 the reward r_{ij} computation is expressed as function of the standard deviation σ_{ij}.

$$r_{ij} = \begin{cases} +1 & \text{if } \tau_{ij} - \tau_{m,ij} \leq \sigma_{ij} \\ -1 & \text{if } \tau_{ij} - \tau_{m,ij} \geq 2\sigma_{ij} \\ 0 & \text{else} \end{cases} \tag{47}$$

Since the probability distributions $\mathcal{N}_{ij}$ are on-line computed, the reward evaluations are continuously updated to meet environmental variations and to detect abnormal events that might occur. The third case in Eq. 47, which brings a neutral reward, has been introduced as an hysteresis guard that prevents reward oscillations.

3. Evaluation

In this section, a simple smart sensing application is considered to evaluate the methodology presented in the previous section. In the proposed evaluations, we show how the system is autonomously coordinating the nodes interactions as results of specific event patterns.

The system is composed of a set of nodes which are able to perform measurements within their own visibility range and to communicate with others. For instance, this would be the case of a distributed video-surveillance deployment, where each node is a SC able to detect visual targets. No knowledge about node positions, visibility ranges and neighborhoods are needed during system setup.

The simulations are performed using the discrete event simulator OMNeT++ [**Var08**], installed in a PC workstation. The simulation tool allows virtual node instances with custom typologies and configuration. In the following evaluations, the different setups are considered with different node placement. The main goal of these evaluations is to show the effect of nodes coordination to reach a stable state without assumptions on the setup environment. Moreover, the system is also evaluated in term of robustness with respect to message losses (as an uniform density of probability of losing messages) and to glitches detection (which causes false alarms).

The state transition probabilities p_{ij} are initially setup to $1/N$ (where N is the number of the available nodes, as in Eq. 33). Assuming this uniform probability results in an initially unbiased state transition policy. Moreover,

the target trajectories are deterministic with constant motion speeds and the observation matrix $\mathbf{B}$ is considered equivalent to an identity matrix.

In the followings, the state transition probability is then computed according to Eq. 46 as:

$$p'_{ij} = p_{ij} + \alpha(r_{ij} - p_{ij}) \tag{48}$$

where α is fixed to 0.2 and the reward is computed as in Eq. 47. The expected delays time $\tau_{m,ij}$ are evaluated as a weighted moving average over the last measures by each node in order to reduce the effect of outliers.

In Fig. 5.7 the overall coordination algorithm is finally shown. The spatial correlation acts as the external stimuli term, that triggers a new state prediction. Along with the input stimuli, the status is evaluated with the most probable transition, based on the temporal correlation. In case the prediction is true, the corresponding transition probabilities are modified according to Eq. 48.

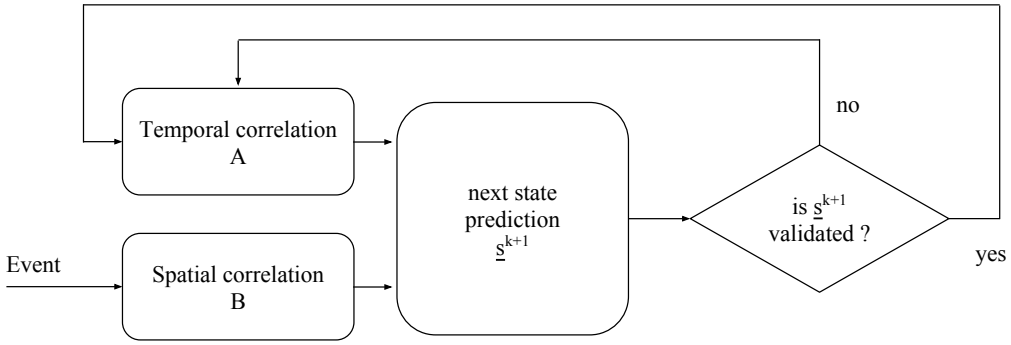


FIGURE 5.7. The coordination algorithm.

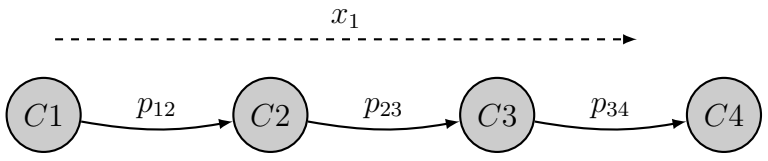


FIGURE 5.8. Aligned nodes configuration.

5.1. Aligned nodes. In Fig. 5.8 the first setup configuration is shown. In particular, the nodes $C_1, \cdots, C_4$ are aligned to the target x_1 trajectory. The

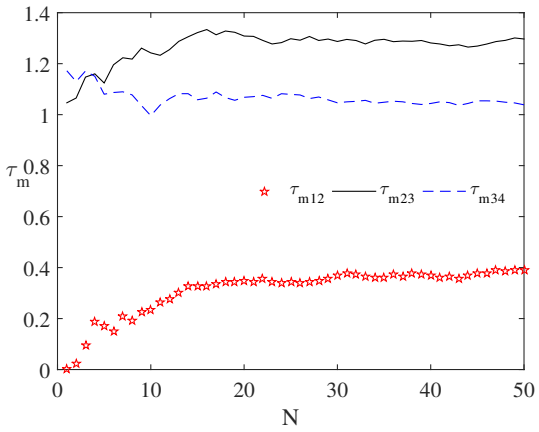
target is repeatedly following the same path, from the left to the right side (the dashed line in Fig. 5.8). In Eq. 49, the initial setup is shown. According to Eq. 42, the p_{1j} are initially setup to 1/4, this results in equal probability of event transition from C_1. Note that the p_{11} probability is non-null to represent a steady state transition, where the next state prediction involves the same node. In the first assumption, the nodes do not share spatial correlation between their measure, thus the matrix B is assumed as the identity matrix.

$$A = \begin{pmatrix} 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \end{pmatrix} B = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{49}$$

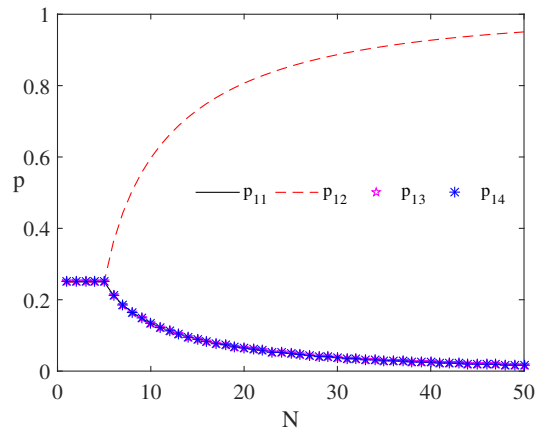
In Fig. 5.9 the simulation results are shown for 50 time samples. In particular, in Fig. 5.9a the expected time τ_{m_ij} are shown. The time delay measures are continuously updated with new entries and when a stable average is achieved, the delay time expectation provides relevant event correlation information, which incrementally reinforces the probability of transitions. Obviously the model relies on the assumption that event tracks follow a predefined path, e.g., pedestrian along the walkways. On the contrary, if the paths are randomly chosen, we expect a random delay time expectation as a consequence. The τ_{m_ij} values reflect the arbitrary localization of nodes in the simulation, where for instance C_1 looks closer to C_2 than C_2 to C_3 and C_3 to C_4 are. In Fig. 5.9b the resulting state transition probabilities from node C_1 are computed. According to Eq. 48, the learning algorithm rewards those transitions where time delay expectations fall into the delay histogram distribution (Eq. 47). Seen by node C_1, only the transition $C_1 \rightarrow C_2$ is revealed, thus the relative transition probability p_{12} is incrementally increased. Given the constrained simulation scenario, the probability asymptotically reaches 1, while others decrease to zero. The final A matrix outcome is as follows:

$$A = \begin{pmatrix} 0 & \uparrow p_{12} & \downarrow p_{13} & \downarrow p_{14} \\ \downarrow p_{21} & 0 & \uparrow p_{23} & \downarrow p_{24} \\ \downarrow p_{31} & \downarrow p_{32} & 0 & \uparrow p_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{50}$$

where upside and downside arrows describe the increase or decrease probability trends with the simulation time respectively.



(A) τ_m in the ideal case

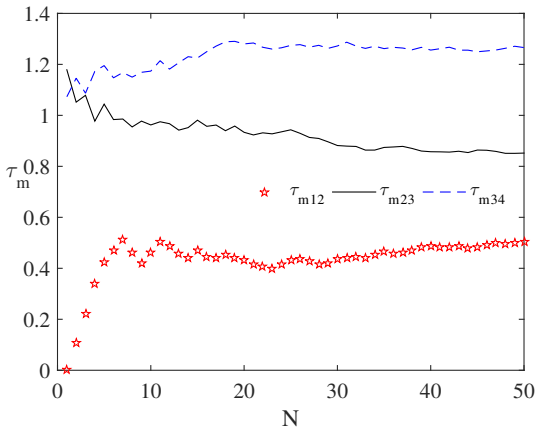


(B) p_{1j} in the ideal case

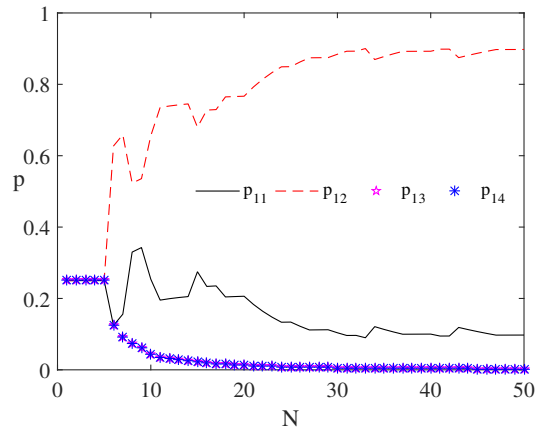
FIGURE 5.9. The system dynamic seen from node C_1 in the aligned nodes setup.

In Fig. 5.10b and 5.10a the same system configuration is evaluated in case of random communication failures. This setup considers the impact of message losses that might occur in real world applications. With respect to the ideal case, several τ_m evaluations are not properly dispatched to other nodes, as a noise term in the time delay expectations. This results in a slower system response, where also transition probabilities are affected. Although the p_{12} still shows the most

probable transition, the state s_{11} absorbs the missed transitions through a non-null p_{11}. The result is intuitive: when nodes are not able to associate an event transition to other nodes, the event is considered stable in the node itself.



(A) τ_m with message loss



(B) p_{1j} with message loss

FIGURE 5.10. The system dynamic seen from node C_1 in the aligned nodes setup.

5.2. Non-aligned nodes. The second test case introduces nodes $C_1, \cdots, C_4$ with a non aligned configuration. In this case, also the event x_1 trajectory might change between the two dashed lines in Fig. 5.11. The A and B matrix initial

setup is unchanged with respect to the previous setup, hence equal state transition probabilities are initially considered and no spatial correlation assumed.

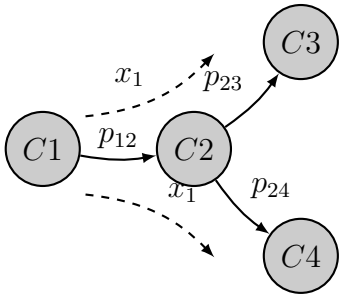
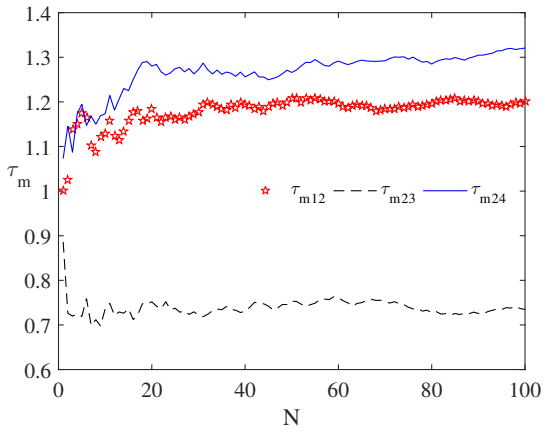


FIGURE 5.11. Non-aligned nodes configuration.

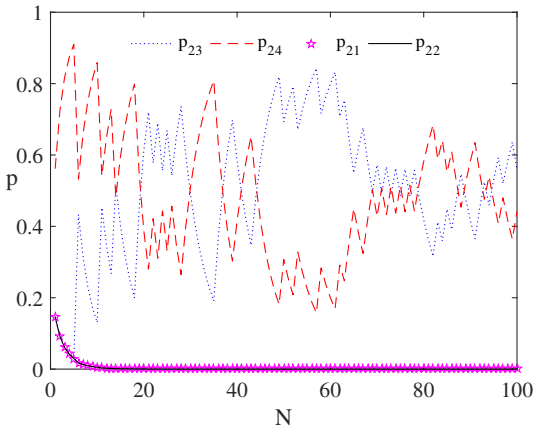
Once again, the evaluation is presented in the Fig. 5.12 as function of the delay time expectation τ_m and stochastic connectivity A. In this case, the environment presents a degree of variability, where the target event might follow two different tracks. What is relevant to evaluate is the capability of the system to reach stable transition probabilities despite the variable event sources. In this case, ideal and message losses simulations have been extended to 100 and 1200 samples respectively.

The event trajectory is modeled as a binary random variable, with expected value equal to η. Although the variability of the trajectory, the time delay expectations in Fig. 5.12a are indeed consistent to what expected, since delays are related to node positioning while it does not refer to trajectories. The variability instead affect the state transitions from C_2, in Fig. 5.12b, where the reward factor is applied each time the next state prediction fails. The system rapidly rejects the contributions of $s_2 1$ and $s_2 2$, while the transition probability associated to s_{23} and s_{24}, p_{23} and p_{24}, are being complementary modified. Finally, the p_{23} and p_{24} probabilities asymptotically converges to η, in this case equal to 0.5.

In Fig. 5.13a the message loss impact are shown for the non aligned configuration. The τ_m values show same expectation as in the ideal case, because the τ_m evaluation only considers confirmed event detections between nodes. A message loss or different trajectory appears in the same way for the delay evaluations. Conversely, the resulting state transition probability is affected. In Fig. 5.13b, the transition probabilities seen by C_2 are shown. As expected, the state s_{22} absorbs the system fluctuation by reducing the transition rate. In the simulation, the transition probabilities p_{23} and p_{24} converge towards 0.4 in 1200 simulation



(A) τ_m in the ideal case



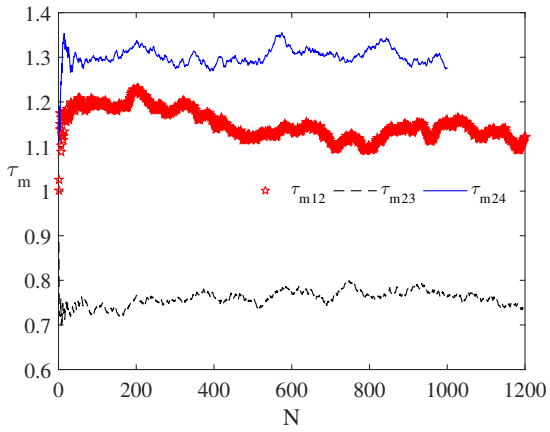
(B) p_{2j} in the ideal case

FIGURE 5.12. The system dynamic seen from node C_2 in the non-aligned nodes setup.

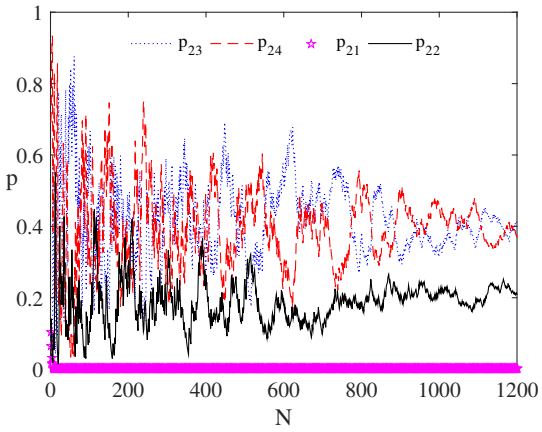
steps. Finally, the system achieves the steady state transition vector yet it does take more time to converge (the time scale has been extended to 1000 samples) with a non null steady state probability p_{22} rate.

4. Model extension

When referring specifically to SCN domain, several considerations can be added to the aforementioned coordination model. For instance, we refer to the illustrative example in Fig. 5.14, where a set of cameras are placed road-side to



(A) τ_m with message loss



(B) p_{2j} with message loss

FIGURE 5.13. The system dynamic seen from node $C2$ in the non-aligned nodes setup.

follow pedestrian movements. This setup has been defined as *static network* category in Chapter 2, where the place and link graphs are not supposed to change once they achieve the stable configuration. A "stable" configuration is reached when the event correlation components, temporal and spatial, are retrieved according to Eq.42.

Temporal correlation indeed relies on the event interaction between two nodes with respect to the same event category. In Fig. 5.14, the rightmost camera reveals a specific pedestrian silhouette $pedestrian(t_0)$ at time t_0, which is eventually

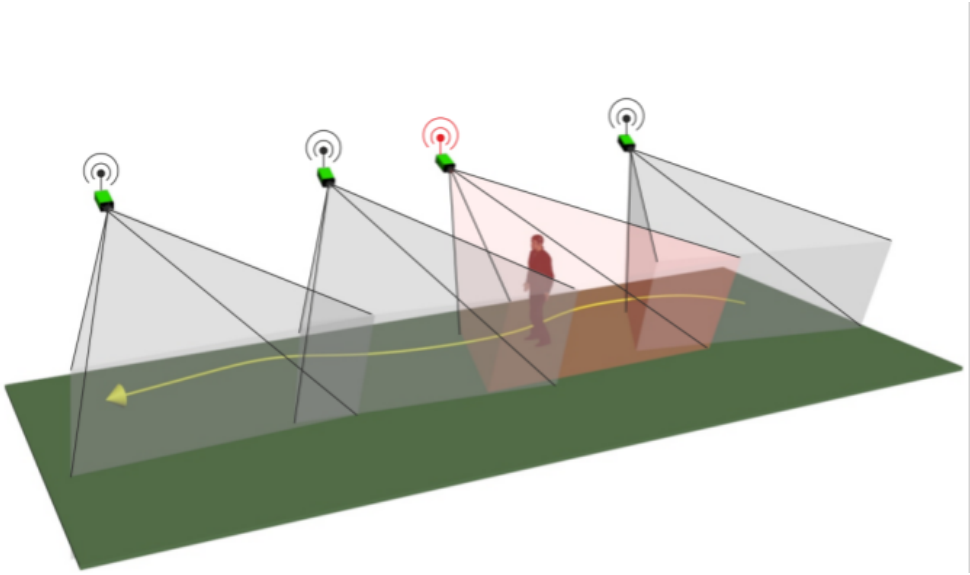


FIGURE 5.14. Smart Camera deployment example.

revealed by the next camera at time $t_0 + \tau$, where τ is the delay time expectation as in Eq.46. In case τ shows a stochastic distribution, the τ expectation reveals a temporal relation between the relative nodes. This aspect obviously requires strict assumptions that should be satisfied:

- (i) timely event detection,
- (ii) reliable event detection,
- (iii) unequivocally identifiable and temporal invariant event.

Both (i) and (ii) imposes critical timing and processing constraints to the SC architecture. These broader events robustness issues, initially addressed in Chapter 3, are challenging the computing architecture to provide timely and reliable visual events detections in low-end hardware platforms. In order to address these issues, in Chapter 4 an optimized smart camera architecture is presented.

Moreover, (iii) introduces the aspect of event re-identification. Once the event, e.g., pedestrian passing, has been revealed by the first camera, how can it be safely identified by next one? In other words, how can the next node assess that the event $pedestrian(t_0+\tau)$ is equivalent to the $pedestrian(t_0)$? Due to the variability experienced in visual features, these questions are more important especially when a dense event space is considered. The computer vision algorithm, besides the detection, should be also able to extract unique features that are sufficient to label the event and make them available for other nodes.

Spatial correlation instead acts as the reinforce term in the event detection. Whenever the neighbours camera field-of-views result partially overlapped, a spatial correlation occurs. Since the respective visual information are related, same event labels can be simultaneously detected. The relation between events is not temporal anymore, but rather spatial. In Eq. 44, the B matrix infers spatial correlation as function the intersection of the respective *visibility ranges*, where multiple appearances of the same target event are simultaneously detected.

CHAPTER 6

Conclusions

This thesis discussed the architecture organization and model formalization of the next generation, video-enabled, CPS. These vision-enabled devices, known in literature as SCs, are nowadays challenging the processing and communication models to perform complex, computationally intensive processing operations with limited computational resources. At first, in Chapter 2 we proposed an architecture formalism to describe and assess the structure of a pervasive network of computing nodes. Nodes structures and iterations are therefore described, enabling discrete evaluations through a linear architecture model.

Once the behavioural structure has been formally described, in Chapter 3 the event detection algorithm concept is proposed. In using events instead of raw data transmission, the network relies on a distributed understanding that permits scalable, computationally efficient system. Events convey particular meanings of the scene understanding yet they might require global understanding to recover higher level inferences according to an aggregation model.

In Chapter 4 we propose an heterogeneous smart camera architecture. It encompasses the Chapter 2 architecture formalism with a reconfigurable heterogeneous platform. By leveraging the hardware software co-design paradigm, the proposed system is able to unleash the potential of the hybrid processing architecture opening the way to a newer technological trend called Internet of Reconfigurable Things (IoT). Embedded, flexible, vision-enabled devices are then capable to interact each other through local network in order to exchange semantic information, notifications or directly processing operators.

This part opens the way to the interaction model, which is proposed in Chapter 5. We proposed a novel event correlation method tailored to smart sensing application, where nodes are autonomously coordinating their interactions. Once the model has been defined, each local event can be classified by inferring correlations with its neighbor events, thus enabling event prediction properties. With

respect to common sensor deployments, a coordination algorithm ease the system installation, allows higher level data aggregation and improves the system reliability over the time.

Concerning the perspectives, many are the future outcomes. At first, the coordination model clearly requires further evaluation with real-world event generations. This thesis has shown that such a model is suitable for distributed event understanding for specific network topologies. However, a dynamic network of moving computing sites is not yet considered. This extension would be extremely of interest when dealing with mobile vision nodes, e.g., vehicles, within a static network infrastructure. Secondly, the place graph dynamic would be also considered. In such a way, computing operators can move between sites in order to optimize the activity costs or to absorb processing peaks from other computing sites.

Publication list

Book chapters

Matteo Petracca, Claudio Salvadori, Andrea Azzarà, Daniele Alessandrelli, Stefano Bocchino, Luca Maggiani, Paolo Pagano, Asier Perallos, Unai Hernandez-Jayo, Enrique Onieva, and Ignacio Julio García-Zuazola. *Middleware Solution to Support ITS Services in IoT-based Visual Sensor Networks*, pages 149–166. John Wiley & Sons, Ltd, 2015

Research reports

Maxime Pelcat, Karol Desnos, Luca Maggiani, Yanzhou Liu, Julien Heulot, Jean-François Nezan, and Shuvra S. Bhattacharyya. Models of Architecture. Research Report PREESM/2015-12TR01, 2015, December 2015

International Journals

Federico Civerchia, Stefano Bocchino, Claudio Salvadori, Enrico Rossi, Luca Maggiani, and Matteo Petracca. Industrial internet of things monitoring solution for advanced predictive maintenance applications. *Journal of Industrial Information Integration*, 2017

Luca Maggiani, Cédric Bourrasset, Jean-Charles Quinton, François Berry, and Jocelyn Sérot. Bio-inspired heterogeneous architecture for real-time pedestrian detection applications. *Journal of Real-Time Image Processing*, pages 1–14, 2016

Cédric Bourrasset, Luca Maggiani, Jocelyn Sérot, and François Berry. A dataflow object detection system for FPGA-based smart camera. *IET Circuits,*

Devices and Systems, 2016

Maxime Pelcat, Cédric Bourrasset, Luca Maggiani, and François Berry. Design productivity of a high level synthesis compiler versus hdl. In *2016 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (IC-SAMOS 2016)*, 2016

Luca Maggiani, Cedric Bourrasset, Matteo Petracca, Francois Berry, Paolo Pagano, and Claudio Salvadori. HOG-Dot: A parallel kernel-based gradient extraction for embedded image processing. *Signal Processing Letters, IEEE*, 22(11):2132–2136, 2015

International Conferences

Luca Maggiani, Lobna Ben Khelifa, Jean-Charles Quinton, Matteo Petracca, Paolo Pagano, and François Berry. Distributed coordination model for smart sensing applications. In *Proceedings of the 10th International Conference on Distributed Smart Camera*, pages 110–115. ACM, 2016

Lobna Ben Khelifa, Luca Maggiani, Jean-Charles Quinton, and François Berry. Ant-cams network: a cooperative network model for silly cameras. In *Proceedings of the 10th International Conference on Distributed Smart Camera*, pages 104–109. ACM, 2016

Lobna Ben Khelifa, Luca Maggiani, Ayoub ElHazzaz, Jean Charles Quinton, and François Berry. Ant-cam network: Demo. In *Proceedings of the 10th International Conference on Distributed Smart Camera*, pages 216–217. ACM, 2016

Maxime Pelcat, Karol Desnos, Luca Maggiani, Yanzhou Liu, Julien Heulot, Jean-François Nezan, and Shuvra S Bhattacharyya. Models of architecture: Reproducible efficiency evaluation for signal processing systems. In *Signal Processing Systems (SiPS), 2016 IEEE International Workshop on*, pages 121–126. IEEE,

2016

F Civerchia, E Rossi, L Maggiani, S Bocchino, C Salvadori, and M Petracca. Lightweight error correction technique in industrial ieee802. 15.4 networks. In *Industrial Electronics Society, IECON 2016-42nd Annual Conference of the IEEE*, pages 6044–6048. IEEE, 2016

Luca Maggiani, Cédric Bourrasset, François Berry, Jocelyn Sérot, Matteo Petracca, and Claudio Salvadori. Parallel image gradient extraction core for FPGA-based smart cameras. In *Proceedings of the 9th International Conference on Distributed Smart Camera*, pages 128–133. ACM, 2015

Luca Maggiani, Claudio Salvadori, Matteo Petracca, Paolo Pagano, and Roberto Saletti. Reconfigurable architecture for computing histograms in real-time tailored to FPGA-based smart camera. In *Industrial Electronics (ISIE), 2014 IEEE 23rd International Symposium on*, pages 1042–1046. IEEE, 2014

Luca Maggiani, Claudio Salvadori, Matteo Petracca, Paolo Pagano, and Roberto Saletti. Reconfigurable FPGA architecture for computer vision applications in smart camera networks. In *Distributed Smart Cameras (ICDSC), 2013 Seventh International Conference on*, page 6. IEEE, 2013

Bibliography

[AAS+15] Sheeraz A Alvi, Bilal Afzal, Ghalib A Shah, Luigi Atzori, and Waqar Mahmood. Internet of multimedia things: Vision and challenges. *Ad Hoc Networks*, 2015.

[AKC+07] A. Abbo, R. Kleihorst, V. Choudhary, L. Sevat, P. Wielage, S. Mouy, and M. Heijligers. Xetal-ii: A 107 gops, 600mw massively-parallel processor for video scene analysis. In *Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International*, pages 270–602, 2007.

[AKC+08] Anteneh A Abbo, Richard P Kleihorst, Vishal Choudhary, Leo Sevat, Paul Wielage, Sebastien Mouy, Bart Vermeulen, and Marc Heijligers. Xetal-ii: a 107 gops, 600 mw massively parallel processor for video scene analysis. *IEEE Journal of Solid-State Circuits*, 43(1):192–201, 2008.

[Alt92] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[ARABL17] Daniel Alonso-Román, Marco Asensio, and Baltasar Beferull-Lozano. Consensus-based distributed state estimation of biofilm in reverse osmosis membranes by wsns. In *Proceedings of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks*, pages 7–10. ACM, 2017.

[ARBL16] Daniel Alonso-Román and Baltasar Beferull-Lozano. Adaptive consensus-based distributed kalman filter for wsns with random link failures. In *Distributed Computing in Sensor Systems (DCOSS), 2016 International Conference on*, pages 187–192. IEEE, 2016.

[AW09] Emile Aarts and Reiner Wichert. *Ambient intelligence*. Springer, 2009.

[Bae05] Jos CM Baeten. A brief history of process algebra. *Theoretical Computer Science*, 335(2):131–146, 2005.

[BDT99] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm intelligence: from natural to artificial systems*. Number 1. Oxford university press, 1999.

[BHA+03] Marcel Bosc, Fabrice Heitz, Jean-Paul Armspach, Izzie Namer, Daniel Gounot, and Lucien Rumbach. Automatic change detection in multimodal serial mri: application to multiple sclerosis lesion evolution. *Neuroimage*, 20(2):643–656, 2003.

[BKDB10] Sebastian Bauer, Sebastian Köhler, Konrad Doll, and Ulrich Brunsmann. FPGA-GPU architecture for kernel SVM pedestrian detection. In *Computer*

Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on, pages 61–68. IEEE, 2010.

[BKME+16] Lobna Ben Khelifa, Luca Maggiani, Ayoub ElHazzaz, Jean Charles Quinton, and François Berry. Ant-cam network: Demo. In *Proceedings of the 10th International Conference on Distributed Smart Camera*, pages 216–217. ACM, 2016.

[BKMQB16] Lobna Ben Khelifa, Luca Maggiani, Jean-Charles Quinton, and François Berry. Ant-cams network: a cooperative network model for silly cameras. In *Proceedings of the 10th International Conference on Distributed Smart Camera*, pages 104–109. ACM, 2016.

[BMP+10] Majid Bahrepour, Nirvana Meratnia, Mannes Poel, Zahra Taghikhaki, and Paul JM Havinga. Distributed event detection in wireless sensor networks for disaster management. In *Intelligent Networking and Collaborative Systems (INCOS), 2010 2nd International Conference on*, pages 507–512. IEEE, 2010.

[BMS+13] Cédric Bourrasset, Luca Maggiani, Jocelyn Sérot, François Berry, and Paolo Pagano. Distributed FPGA-based smart camera architecture for computer vision applications. In *International Conference on Distributed Smart Cameras 2013*, 2013 Seventh International Conference on Distributed Smart Cameras (ICDSC 2013), Palm Springs, United States, October 2013. IEEE/ACM.

[BMSB16] Cédric Bourrasset, Luca Maggiani, Jocelyn Sérot, and François Berry. A dataflow object detection system for FPGA-based smart camera. *IET Circuits, Devices and Systems*, 2016.

[BOHS14] Rodrigo Benenson, Mohamed Omran, Jan Hosang, and Bernt Schiele. Ten years of pedestrian detection, what have we learned? In *Computer Vision-ECCV 2014 Workshops*, pages 613–627. Springer, 2014.

[BP66] Leonard E Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6):1554–1563, 1966.

[BRH13] Charlotte Blair, Neil M Robertson, and Danny Hume. Characterizing a heterogeneous system for person detection in video using histograms of oriented gradients: Power versus speed versus accuracy. *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, 3(2):236–247, 2013.

[BS06] John Boardman and Brian Sauser. System of systems-the meaning of of. In *System of Systems Engineering, 2006 IEEE/SMC International Conference on*, pages 6–pp. IEEE, 2006.

[Cam16] Cambridge University Press. Cambridge Dictionary. http://dictionary.cambridge.org, 2016.

[CB96] Pol R Coppin and Marvin E Bauer. Digital change detection in forest ecosystems with remote sensing imagery. *Remote sensing reviews*, 13(3-4):207–234, 1996.

[CBS+17] Federico Civerchia, Stefano Bocchino, Claudio Salvadori, Enrico Rossi, Luca Maggiani, and Matteo Petracca. Industrial internet of things monitoring solution for advanced predictive maintenance applications. *Journal of Industrial Information Integration*, 2017.

[CCC+06] Dan Chalmers, Matthew Chalmers, Jon Crowcroft, Marta Kwiatkowska, Robin Milner, Eamonn O'Neill, Tom Rodden, Vladimiro Sassone, and Morris Sloman. Ubiquitous computing: experience, design and science. In *Ubiquitous Computing: 8th International Conference*, 2006.

[CCT+15] Shao-Yi Chien, Wei-Kai Chan, Yu-Hsiang Tseng, Chia-Han Lee, V Srinivasa Somayazulu, and Yen-Kuang Chen. Distributed computing in iot: System-on-a-chip for smart cameras as an example. In *Design Automation Conference (ASP-DAC), 2015 20th Asia and South Pacific*, pages 130–135. IEEE, 2015.

[CFK+14] Ching-Hui Chen, Julien Favre, Gregorij Kurillo, Thomas P. Andriacchi, Ruzena Bajcsy, and Rama Chellappa. Camera networks for healthcare, teleimmersion, and surveillance. *Computer*, 47(5):26–36, 2014.

[CRM+16] F Civerchia, E Rossi, L Maggiani, S Bocchino, C Salvadori, and M Petracca. Lightweight error correction technique in industrial ieee802. 15.4 networks. In *Industrial Electronics Society, IECON 2016-42nd Annual Conference of the IEEE*, pages 6044–6048. IEEE, 2016.

[DA10] Falko Dressler and Ozgur B. Akan. A survey on bio-inspired networking. *Computer Networks*, 54(6):881 – 900, 2010.

[DeL08] Daniel A DeLaurentis. Understanding transportation as a system of systems problem. *System of systems engineering*, pages 520–541, 2008.

[DLRB08] Judith Dahmann, J Lane, G Rebovich, and K Baldwin. A model of systems engineering in a system of systems context. In *Proceedings of the Conference on Systems Engineering Research, Los Angeles, CA, USA (April 2008)*, 2008.

[DT05] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893, 2005.

[DWSP12] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(4):743–761, 2012.

[EG09] Markus Enzweiler and Darieu M Gavrila. Monocular pedestrian detection: Survey and experiments. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(12):2179–2195, 2009.

[ELC+13] L. Esterle, P.R. Lewis, H. Caine, Xin Yao, and B. Rinner. Camsim: A distributed smart camera network simulator. In *Self-Adaptation and Self-Organizing Systems Workshops (SASOW), 2013 IEEE 7th International Conference on*, pages 19–20, Sept 2013.

[ELYR14] Lukas Esterle, Peter R Lewis, Xin Yao, and Bernhard Rinner. Socio-economic vision graph generation and handover in distributed smart camera networks. *ACM Transactions on Sensor Networks (TOSN)*, 10(2):20, 2014.

[EMM91] Howard Eisner, John Marciniak, and Ray McMillan. Computer-aided system of systems (s2) engineering. In *Systems, Man, and Cybernetics, 1991.'Decision Aiding for Complex Systems, Conference Proceedings., 1991 IEEE International Conference on*, pages 531–537. IEEE, 1991.

[ENLM11] Roger D Eastman, Nathan S Netanyahu, and Jacqueline Le Moigne. Survey of image registration methods., 2011.

[ES12] Uğur Murat Erdem and Stan Sclaroff. Event prediction in a hybrid camera network. *ACM Trans. Sen. Netw.*, 8(2):16:1–16:27, March 2012.

[EVGW+10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[FKBR14] Karl J. Friston, Joshua Kahan, Bharat Biswal, and Adeel Razi. A dcm for resting state fmri. *NeuroImage*, 94(0):396 – 407, 2014.

[FKSK15] Michał Fularz, Marek Kraft, Adam Schmidt, and Andrzej Kasiński. The architecture of an embedded smart camera for intelligent inspection and surveillance. In *Progress in Automation, Robotics and Measuring Techniques*, pages 43–52. Springer, 2015.

[Fou] HSA Foundation. Harmonizing the industry around heterogeneous computing.

[FPF17] Alexey G Finogeev, Danila S Parygin, and Anton A Finogeev. The convergence computing model for big sensor data mining and knowledge discovery. *Human-centric Computing and Information Sciences*, 7(1):11, 2017.

[GBMP13] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660, 2013.

[GC14] Daniel Griffith and Yongwan Chun. Spatial autocorrelation and spatial filtering. In *Handbook of regional science*, pages 1477–1507. Springer, 2014.

[GEN+16] A Gardel, F Espinosa, R Nieto, JL Lázaro, and I Bravo. Wireless camera nodes on a cyber-physical system. In *Proceedings of the 10th International Conference on Distributed Smart Camera*, pages 31–36. ACM, 2016.

[Geo12] George Kyriazis. Heterogeneous system architecture: A technical review. "http://developer.amd.com/wordpress/media/2012/10/hsa10.pdf" `http://developer.amd.com/wordpress/media/2012/10/hsa10.pdf`, 2012.

[GJNC+14] Sebastian Gruenwedel, Vedran Jelaca, Jorge Oswaldo Nino-Castaneda, Peter van Hese, Dimitri van Cauwelaert, Dirk van Haerenborgh, Peter Veelaert, and Wilfried Philips. Low-complexity scalable distributed multicamera tracking of humans. *ACM Trans. Sen. Netw.*, 10(2):24:1–24:32, January 2014.

[GLC00] Guodong Guo, Stan Z Li, and Kapluk Chan. Face recognition by support vector machines. In *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, pages 196–201. IEEE, 2000.

[GM07] Dariu M Gavrila and Stefan Munder. Multi-cue pedestrian detection and tracking from a moving vehicle. *International journal of computer vision*, 73(1):41–59, 2007.

[Gre10] Adam Greenfield. *Everyware: The dawning age of ubiquitous computing*. New Riders, 2010.

[Gri03] D.A. Griffith. *Spatial Autocorrelation and Spatial Filtering: Gaining Understanding Through Theory and Scientific Visualization*. Advances in Spatial Science. Springer, 2003.

[GT07] T. Gandhi and M.M. Trivedi. Pedestrian protection systems: Issues, survey, and challenges. *Intelligent Transportation Systems, IEEE Transactions on*, pages 413–430, 2007.

[Gup01] Pallav Gupta. Hardware-software codesign. *IEEE Potentials*, 20(5):31–32, 2001.

[GYJW10] Quansheng Guan, F Richard Yu, Shengming Jiang, and Gang Wei. Prediction-based topology control and routing in cognitive radio mobile ad hoc networks. *IEEE Transactions on Vehicular Technology*, 59(9):4443–4452, 2010.

[HAPP12] Markus Happe, Andreas Agne, Christian Plessl, and Marco Platzner. Hardware/software platform for self-aware compute nodes. In *Proc. Workshop on Self-Awareness in Reconfigurable Computing Systems (SRCS)*, pages 8—9, 2012.

[HDW17] Harry H Hilton, Steven J D'Urso, and Noe Wiener. Designer systems of systems: A rational integrated approach of system engineering to tailored aerodynamics, aeroelasticity, aero-viscoelasticity, stability, control, geometry, materials, structures, propulsion, performance, sizing, weight, cost. In *Transdisciplinary Perspectives on Complex Systems*, pages 49–84. Springer, 2017.

[HF85] Joseph Y Halpern and Ronald Fagin. A formal model of knowledge, action, and communication in distributed systems: preliminary report. In *Proceedings of the fourth annual ACM symposium on Principles of distributed computing*, pages 224–236. ACM, 1985.

[HHCP08] Jianwei Huang, Zhu Han, Mung Chiang, and H Vincent Poor. Auction-based resource allocation for cooperative communications. *IEEE Journal on Selected Areas in Communications*, 26(7), 2008.

[HJK+00] Ahmed Hemani, Axel Jantsch, Shashi Kumar, Adam Postula, Johnny Oberg, Mikael Millberg, and Dan Lindqvist. Network on chip: An architecture for billion transistor era. In *Proceeding of the IEEE NorChip Conference*, volume 31, page 11, 2000.

[HKHK15] Gwang-Soo Hong, Byung-Gyu Kim, Young-Sup Hwang, and Kee-Koo Kwon. Fast multi-feature pedestrian detection algorithm based on histogram of oriented gradient using discrete wavelet transform. *Multimedia Tools and Applications*, 2015.

[HLES17] Pablo Hernandez-Leal, Hugo Jair Escalante, and L Enrique Sucar. Towards a generic ontology for video surveillance. In *Applications for Future Internet*, pages 3–7. Springer, 2017.

[HLP13] Markus Happe, Enno Lübbers, and Marco Platzner. A self-adaptive heterogeneous multi-core architecture for embedded real-time video object tracking. *J. Real-Time Image Process.*, 8(1):95–110, March 2013.

[HM05] Tony Hoare and Robin Milner. Grand challenges for computing research. *The Computer Journal*, 48(1):49–52, 2005.

[HMY+15] Chunsheng Hua, Yasushi Makihara, Yasushi Yagi, Shun Iwasaki, Keisuke Miyagawa, and Bo Li. Onboard monocular pedestrian detection by combining spatio-temporal HOG with structure from motion algorithm. *Machine Vision and Applications*, 26(2-3):161–183, 2015.

[Hos16] M Shamim Hossain. Patient status monitoring for smart home healthcare. In *Multimedia & Expo Workshops (ICMEW), 2016 IEEE International Conference on*, pages 1–6. IEEE, 2016.

[HPD09] Mohamed Hussein, Fatih Porikli, and Larry Davis. A comprehensive evaluation framework and a comparative study for human detectors. *Intelligent Transportation Systems, IEEE Transactions on*, 10(3):417–427, 2009.

[HSH+13] Michael Hahnle, Frerk Saxen, Matthias Hisung, Ulrich Brunsmann, and Konrad Doll. FPGA-based real-time pedestrian detection on high-resolution images. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference on*, pages 629–635, 2013.

[HSS+14] Chih-Ming Hsieh, Farzad Samie, M. Sammer Srouji, Manyi Wang, Zhonglei Wang, and Jörg Henkel. Hardware/software co-design for a wireless sensor network platform. In *Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis*, CODES '14, pages 1:1–1:10. ACM, 2014.

[Jai10] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.

[Jen06] Ole Høgh Jensen. Mobile processes in bigraphs. *Monograph available at http://www.cl.cam.ac.uk/˜ rm135/Jensen-monograph.html*, 2006.

[JGDE08] Andreas Janecek, Wilfried N Gansterer, Michael Demel, and Gerhard Ecker. On the relationship between feature selection and classification accuracy. *FSDM*, 4:90–105, 2008.

[Joa99] Thorsten Joachims. Svmlight: Support vector machine. `http://svmlight.joachims.org/`, 1999.

[KBN12] Ariane Keller, Daniel Borkmann, and Stephan Neuhaus. Towards self-adaptation in reconfigurable network nodes. In *Proceeding of the Workshop on Self-Awareness in Reconfigurable Computing Systems (SRCS'12)*, page 1014, 2012.

[KBNH14] Ariane Keller, Daniel Borkmann, Stephan Neuhaus, and Markus Happe. Self-awareness in computer networks. *International Journal of Reconfigurable Computing*, 2014:10, 2014.

[KDM+14] AT Kamal, C Ding, AA Morye, JA Farrell, and Amit K Roy-Chowdhury. An overview of distributed tracking and control in camera networks. In *Wide Area Surveillance*, pages 207–234. Springer, 2014.

[Kin11] Samuel Kinsley. Anticipating ubiquitous computing: Logics to forecast technological futures. *Geoforum*, 42(2):231–240, 2011.

[KJBB09] Mitchell C Kerman, Wei Jiang, Alan F Blumberg, and Samuel E Buttrey. Event detection challenges, methods, and applications in natural and artificial systems. Technical report, DTIC Document, 2009.

[KKN14] Samina Khalid, Tehmina Khalil, and Shamila Nasreen. A survey of feature selection and feature extraction techniques in machine learning. In *Science and Information Conference (SAI), 2014*, pages 372–378. IEEE, 2014.

[KKWS15] Sung Hoon Kim, Jung Su Kim, Vincent Wan, and Il Hong Suh. Automotive adas camera system configuration using multi-core microcontroller. Technical report, SAE Technical Paper, 2015.

[KMS08] Alexander Klaser, Marcin Marszałek, and Cordelia Schmid. A spatio-temporal descriptor based on 3d-gradients. In *BMVC 2008-19th British Machine Vision Conference*, pages 275–1. British Machine Vision Association, 2008.

[Koe84] Jan J Koenderink. The structure of images. *Biological cybernetics*, 50(5):363–370, 1984.

[KPdlTR11] Yana Esteves Krasteva, Jorge Portilla, Eduardo de la Torre, and Teresa Riesgo. Embedded runtime reconfigurable nodes for wireless sensor networks applications. *IEEE Sensors Journal*, 11(9):1800–1810, 2011.

[KPL+10] Ariane Keller, Bernhard Plattner, Enno Lübbers, Marco Platzner, and Christian Plessl. Reconfigurable nodes for future networks. In *Proc. IEEE Globecom Workshop on Network of the Future (FutureNet)*, pages 372–376. IEEE, December 2010.

[KS94] Sandeep Kumar and Eugene H Spafford. An application of pattern matching in intrusion detection. 1994.

[KSH+09] R. Kadota, H. Sugano, M. Hiromoto, H. Ochi, R. Miyamoto, and Y. Nakamura. Hardware architecture for hog feature extraction. In *Intelligent Information Hiding and Multimedia Signal Processing, 2009. IIH-MSP '09. Fifth International Conference on*, pages 1330–1333, 2009.

[KVH16] Georgios Keramidas, Nikolaos Voros, and Michael Hübner. *Components and Services for IoT Platforms: Paving the Way for IoT Standards*. Springer, 2016.

[KWG17] Yoram Koren, Wencai Wang, and Xi Gu. Value creation through design for scalability of reconfigurable manufacturing systems. *International Journal of Production Research*, 55(5):1227–1242, 2017.

[KY95] George Klir and Bo Yuan. *Fuzzy sets and fuzzy logic*, volume 4. Prentice hall New Jersey, 1995.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LM06] James J Leifer and Robin Milner. Transition systems, link graphs and petri nets. *Mathematical Structures in Computer Science*, 16(06):989–1047, 2006.

[LMA+12] Fabrice Lemonnier, Philippe Millet, Gabriel Marchesan Almeida, Michael Hubner, Jürgen Becker, Sébastien Pillement, Olivier Sentieys, Martijn Koedam, Shubhendu Sinha, Kees Goossens, et al. Towards future adaptive multiprocessor systems-on-chip: An innovative approach for flexible architectures. In *Embedded Computer Systems (SAMOS), 2012 International Conference on*, pages 228–235. IEEE, 2012.

[LV94] Gad M Landau and Uzi Vishkin. Pattern matching in a digitized image. *Algorithmica*, 12(4-5):375–408, 1994.

[LYD+15] Wei Liu, Bing Yu, Chengwei Duan, Liying Chai, Huai Yuan, and Hong Zhao. A pedestrian-detection method based on heterogeneous features and ensemble of multi-view–pose parts. *Intelligent Transportation Systems, IEEE Transactions on*, 16(2):813–824, 2015.

[MBB+15] Luca Maggiani, Cédric Bourrasset, François Berry, Jocelyn Sérot, Matteo Petracca, and Claudio Salvadori. Parallel image gradient extraction core for FPGA-based smart cameras. In *Proceedings of the 9th International Conference on Distributed Smart Camera*, pages 128–133. ACM, 2015.

[MBKQ+16] Luca Maggiani, Lobna Ben Khelifa, Jean-Charles Quinton, Matteo Petracca, Paolo Pagano, and François Berry. Distributed coordination model for smart sensing applications. In *Proceedings of the 10th International Conference on Distributed Smart Camera*, pages 110–115. ACM, 2016.

[MBP+15] Luca Maggiani, Cedric Bourrasset, Matteo Petracca, Francois Berry, Paolo Pagano, and Claudio Salvadori. HOG-Dot: A parallel kernel-based gradient extraction for embedded image processing. *Signal Processing Letters, IEEE*, 22(11):2132–2136, 2015.

[MBQ+16] Luca Maggiani, Cédric Bourrasset, Jean-Charles Quinton, François Berry, and Jocelyn Sérot. Bio-inspired heterogeneous architecture for real-time pedestrian detection applications. *Journal of Real-Time Image Processing*, pages 1–14, 2016.

[MDN16] Roberto Marroquin, Julien Dubois, and Christophe Nicolle. Wisenet-smart camera network interacting with a semantic model: Phd forum. In *Proceedings of the 10th International Conference on Distributed Smart Camera*, pages 224–225. ACM, 2016.

[Mil99] Robin Milner. *Communicating and mobile systems: the pi calculus*. Cambridge university press, 1999.

[Mil09] Robin Milner. *The space and motion of communicating agents*. Cambridge University Press, 2009.

[MMN+11] M Magrini, D Moroni, C Nastasi, P Pagano, M Petracca, G Pieri, C Salvadori, and O Salvetti. Visual sensor networks for infomobility. *Pattern Recognition and Image Analysis*, 21(1):20–29, 2011.

[Mor50] Patrick AP Moran. Notes on continuous stochastic phenomena. *Biometrika*, 37(1):17–23, 1950.

[MSP+13] Luca Maggiani, Claudio Salvadori, Matteo Petracca, Paolo Pagano, and Roberto Saletti. Reconfigurable FPGA architecture for computer vision applications in smart camera networks. In *Distributed Smart Cameras (ICDSC), 2013 Seventh International Conference on*, page 6. IEEE, 2013.

[MSP+14] Luca Maggiani, Claudio Salvadori, Matteo Petracca, Paolo Pagano, and Roberto Saletti. Reconfigurable architecture for computing histograms in real-time tailored to FPGA-based smart camera. In *Industrial Electronics (ISIE), 2014 IEEE 23rd International Symposium on*, pages 1042–1046. IEEE, 2014.

[MSPC12] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, and Imrich Chlamtac. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497 – 1516, 2012.

[MTS+05] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiri Matas, Frederik Schaffalitzky, Timor Kadir, and Luc Van Gool. A comparison of affine region detectors. *International journal of computer vision*, 65:43–72, 2005.

[NB72] John A Nelder and R Jacob Baker. Generalized linear models. *Encyclopedia of statistical sciences*, 1972.

[New03] Mark EJ Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

[New08] Mark EJ Newman. The mathematics of networks. *The new palgrave encyclopedia of economics*, 2(2008):1–12, 2008.

[NHB04] Ram Nevatia, Jerry Hobbs, and Bob Bolles. An ontology for video event representation. In *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW'04. Conference on*, pages 119–119. IEEE, 2004.

[NIJP14] Aziz Nasridinov, Sun-Young Ihm, Young-Sik Jeong, and Young-Ho Park. Event detection in wireless sensor networks: Survey and challenges. In *Mobile, Ubiquitous, and Intelligent Computing*, pages 585–590. Springer, 2014.

[NPSM14] Ido Nevat, Gareth W. Peters, Francois Septier, and Tomoko Matsui. Estimation of spatially correlated random fields in heterogeneous wireless sensor networks. *Transactions on Signal Processing , IEEE*, xx(x), 2014.

[NYSF12] Takafumi Nishikawa, Junji Yoshida, Toshiyuki Sugiyama, and Yozo Fujino. Concrete crack detection by multiple sequential image filtering. *Computer-Aided Civil and Infrastructure Engineering*, 27(1):29–47, 2012.

[O+13a] World Health Organization et al. Global status report on road safety 2013: supporting a decade of action: summary. 2013.

[O+13b] World Health Organization et al. *Pedestrian safety: A road safety manual for decision-makers and practitioners*. World Health Organization, 2013.

[OPH96] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 29(1):51–59, 1996.

[OSFM07] R. Olfati-Saber, J.A. Fax, and R.M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, Jan 2007.

[OSJ12] Reza Olfati-Saber and Parisa Jalalkamali. Coupled distributed estimation and control for mobile sensor networks. *Automatic Control, IEEE Transactions on*, 57(10):2609–2614, 2012.

[P+04] Margie Peden et al. World report on road traffic injury prevention, 2004.

[PA15] Leonard Petnga and Mark A Austin. Spatial ontologies and models for safety-critical cyber-physical systems. In *Complex Systems Engineering (ICCSE), 2015 International Conference on*, pages 1–6. IEEE, 2015.

[PBMB16] Maxime Pelcat, Cédric Bourrasset, Luca Maggiani, and François Berry. Design productivity of a high level synthesis compiler versus hdl. In *2016 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (IC-SAMOS 2016)*, 2016.

[PDM+15] Maxime Pelcat, Karol Desnos, Luca Maggiani, Yanzhou Liu, Julien Heulot, Jean-François Nezan, and Shuvra S. Bhattacharyya. Models of Architecture. Research Report PREESM/2015-12TR01, 2015, December 2015.

[PDM+16] Maxime Pelcat, Karol Desnos, Luca Maggiani, Yanzhou Liu, Julien Heulot, Jean-François Nezan, and Shuvra S Bhattacharyya. Models of architecture: Reproducible efficiency evaluation for signal processing systems. In *Signal Processing Systems (SiPS), 2016 IEEE International Workshop on*, pages 121–126. IEEE, 2016.

[PE07] Julia W Patriarche and Bradley J Erickson. Change detection & characterization: A new tool for imaging informatics and cancer research. *Cancer informatics*, 4, 2007.

[PEK+15] C. Piciarelli, L. Esterle, A. Khan, B. Rinner, and G. Foresti. Dynamic reconfiguration in camera networks: a short survey. *Circuits and Systems for Video Technology, IEEE Transactions on*, (99):1–1, 2015.

[Pic04] Massimo Piccardi. Background subtraction techniques: a review. In *Systems, man and cybernetics, 2004 IEEE international conference on*, volume 4, pages 3099–3104. IEEE, 2004.

[PLJC14] C. Perera, C.H. Liu, S. Jayawardena, and M. Chen. A survey on internet of things from industrial market perspective. *Access, IEEE*, 2:1660–1679, 2014.

[PMC+13] Chiara Poletto, Sandro Meloni, Vittoria Colizza, Yamir Moreno, and Alessandro Vespignani. Host mobility drives pathogen competition in spatially structured populations. *PLoS Comput Biol*, 9(8):e1003169, 2013.

[PMDPVdW12] Chris Poppe, Gaëtan Martens, Pieterjan De Potter, and Rik Van de Walle. Semantic web technologies for video surveillance metadata. *Multimedia Tools and Applications*, 56(3):439–467, 2012.

[PSA+15] Matteo Petracca, Claudio Salvadori, Andrea Azzarà, Daniele Alessandrelli, Stefano Bocchino, Luca Maggiani, Paolo Pagano, Asier Perallos, Unai Hernandez-Jayo, Enrique Onieva, and Ignacio Julio García-Zuazola. *Middleware Solution to Support ITS Services in IoT-based Visual Sensor Networks*, pages 149–166. John Wiley & Sons, Ltd, 2015.

[QG11] Jean-Charles Quinton and Bernard Girau. Predictive neural fields for improved tracking and attentional properties. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1629–1636. IEEE, 2011.

[QT08] F. Qureshi and D. Terzopoulos. Smart camera networks in virtual reality. *Proceedings of the IEEE*, pages 1640–1656, 2008.

[RAAKR05] Richard J Radke, Srinivas Andra, Omar Al-Kofahi, and Badrinath Roysam. Image change detection algorithms: a systematic survey. *IEEE transactions on image processing*, 14(3):294–307, 2005.

[RES+15] Bernhard Rinner, Lukas Esterle, Jennifer Simonjan, Georg Nebehay, Roman Pflugfelder, Gustavo Fernandez Dominguez, and Peter R Lewis. Self-aware and self-expressive camera networks. *Computer*, 48(7):21–28, 2015.

[RLSS10] Ragunathan Raj Rajkumar, Insup Lee, Lui Sha, and John Stankovic. Cyber-physical systems: the next computing revolution. In *Proceedings of the 47th Design Automation Conference*, pages 731–736. ACM, 2010.

[Ros16] Christian Rosenke. The exact complexity of projective image matching. *Journal of Computer and System Sciences*, 82(8):1360–1387, 2016.

[RWS+08] B. Rinner, T. Winkler, W. Schriebl, M. Quaritsch, and W. Wolf. The evolution from single to pervasive smart cameras. In *Proceedings of ACM/IEEE International Conference on Distributed Smart Cameras*, pages 1–10, 2008.

[San93] Davide Sangiorgi. Expressing mobility in process algebras: first-order and higher-order paradigms. 1993.

[Sat01] Mahadev Satyanarayanan. Pervasive computing: Vision and challenges. *Personal Communications, IEEE*, 8(4):10–17, 2001.

[Sat17] Mahadev Satyanarayanan. The emergence of edge computing. *Computer*, 50(1):30–39, 2017.

[SBH16] Farzad Samie, Lars Bauer, and Jörg Henkel. Iot technologies for embedded computing: A survey. In *Hardware/Software Codesign and System Synthesis (CODES+ ISSS), 2016 International Conference on*, pages 1–10. IEEE, 2016.

[SBM06] Zehang Sun, George Bebis, and Ronald Miller. On-road vehicle detection: A review. *IEEE transactions on pattern analysis and machine intelligence*, 28(5):694–711, 2006.

[SG99] Chris Stauffer and W Eric L Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, pages 246–252. IEEE, 1999.

[SHK+07] KlaasE Stephan, LeeM Harrison, StefanJ Kiebel, Olivier David, WillD Penny, and KarlJ Friston. Dynamic causal models of neural system dynamics: current state and future extensions. *Journal of Biosciences*, 32(1):129–144, 2007.

[SHTE08] Junji Sakai, Inoue Hiroaki, Sunao Torii, and Masato Edahiro. Multitasking parallel method for high-end embedded appliances. *IEEE micro*, 28(5), 2008.

[SMG09] Juan Carlos SanMiguel, José M Martinez, and Álvaro Garcia. An ontology for event detection and its application in surveillance video. In *Advanced Video and Signal Based Surveillance, 2009. AVSS'09. Sixth IEEE International Conference on*, pages 220–225. IEEE, 2009.

[SMS+14] J.C. SanMiguel, C. Micheloni, K. Shoop, G.L. Foresti, and A. Cavallaro. Self-reconfigurable smart camera networks. *Computer*, 47(5):67–73, May 2014.

[SQ14] Wiktor Starzyk and Faisal Z. Qureshi. A negotiation protocol with conditional offers for camera handoffs. In *Proceedings of the International Conference on Distributed Smart Cameras*, ICDSC '14, pages 17:1–17:7. ACM, 2014.

[SRB01] Chris Savarese, Jan M Rabaey, and Jan Beutel. Location in distributed ad-hoc wireless sensor networks. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 4, pages 2037–2040. IEEE, 2001.

[SSC90] Randall Smith, Matthew Self, and Peter Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*, pages 167–193. Springer, 1990.

[SSR+16] C Soell, L Shi, J Roeber, M Reichenbach, R Weigel, and A Hagelauer. Low-power analog smart camera sensor for edge detection. In *Image Processing (ICIP), 2016 IEEE International Conference on*, pages 4408–4412. IEEE, 2016.

[ST12] Babak Shirmohammadi and Camillo J. Taylor. Self-localizing smart camera networks. *ACM Trans. Sen. Netw.*, 8(2):11:1–11:24, March 2012.

[SW16] Chris Stary and Dominik Wachholder. System-of-systems support—a bigraph approach to interoperability and emergent behavior. *Data & Knowledge Engineering*, 105:155–172, 2016.

[TKH+08] Kei Tanaka, Yasuyo Kita, Masayuki Hirafuji, Seishi Ninomiya, et al. An image change detection application for field server. *Proc. of IAALD-AFITA-WCCA*, pages 49–54, 2008.

[TLGB14] Mohammed Yassine Kazi Tani, Adel Lablack, Abdelghani Ghomari, and Ioan Marius Bilasco. Events detection using a video-surveillance ontology and a rule-based approach. In *European Conference on Computer Vision*, pages 299–308. Springer, 2014.

[TLL07] Xiao-jun Tan, Jun Li, and Chunlu Liu. A video-based real-time vehicle detection method by classified background learning. *World transactions on engineering and technology education*, 6(1):189, 2007.

[TMM+16] Christos Tzelepis, Zhigang Ma, Vasileios Mezaris, Bogdan Ionescu, Ioannis Kompatsiaris, Giulia Boato, Nicu Sebe, and Shuicheng Yan. Event-based media processing and analysis: A survey of the literature. *Image and Vision Computing*, 53:3–19, 2016.

[TPD15] Russell Tessier, Kenneth Pocek, and Andre DeHon. Reconfigurable computing architectures. *Proceedings of the IEEE*, 103(3):332–354, 2015.

[TTY+15] Jun Tanabe, Sano Toru, Yutaka Yamada, Tomoki Watanabe, Mayu Okumura, Manabu Nishiyama, Tadakazu Nomura, Kazushige Oma, Nobuhiro Sato, Moriyasu Banno, et al. 18.2 a 1.9 tops and 564gops/w heterogeneous multicore soc with color-based object classification accelerator for image-recognition applications. In *Solid-State Circuits Conference-(ISSCC), 2015 IEEE International*, pages 1–3. IEEE, 2015.

[TW17] Thomas N Theis and H-S Philip Wong. The end of moore's law: A new beginning for information technology. *Computing in Science & Engineering*, 19(2):41–50, 2017.

[VAA04] Mehmet C Vuran, Özgür B Akan, and Ian F Akyildiz. Spatio-temporal correlation: theory and applications for wireless sensor networks. *Computer Networks*, 45(3):245–259, 2004.

[Vap82] Vladimir Vapnik. *Estimation of Dependences Based on Empirical Data: Springer Series in Statistics (Springer Series in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982.

[Var08] András Varga. Omnet++ discrete event simulation system. 2008.

[VBDO+14] Leandro A Villas, Azzedine Boukerche, Horacio ABF De Oliveira, Regina B De Araujo, and Antonio AF Loureiro. A spatial correlation aware algorithm to perform efficient data collection in wireless sensor networks. *Ad Hoc Networks*, 12:69–85, 2014.

[VFPC15] Eugenio Valdano, Luca Ferreri, Chiara Poletto, and Vittoria Colizza. Analytical computation of the epidemic threshold on temporal networks. *Physical Review X*, 5(2):021005, 2015.

[vRGG+14] Jeroen van Rest, FA Grootjen, Marc Grootjen, Remco Wijn, Olav Aarts, ML Roelofs, Gertjan J Burghouts, Henri Bouma, Lejla Alic, and Wessel

Kraaij. Requirements for multimedia metadata schemes in surveillance applications for security. *Multimedia tools and applications*, 70(1):573–598, 2014.

[Wat89] Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, University of Cambridge England, 1989.

[WB95] Greg Welch and Gary Bishop. An introduction to the kalman filter. 1995.

[WDA+12] Georg Wittenburg, Norman Dziengel, Stephan Adler, Zakaria Kasmi, Marco Ziegert, and Jochen Schiller. Cooperative event detection in wireless sensor networks. *IEEE Communications Magazine*, 50(12), 2012.

[WDWS10] Georg Wittenburg, Norman Dziengel, Christian Wartenburger, and Jochen Schiller. A system for distributed event detection in wireless sensor networks. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 94–104. ACM, 2010.

[Wei91] Mark Weiser. The computer for the 21st century. *Scientific american*, 265(3):94–104, 1991.

[Wer74] Paul John Werbos. Beyond regression: new tools for prediction and analysis in the behavioral science. *Ph. D. Thesis, Harvard University*, 1974.

[WGB99] Mark Weiser, Rich Gold, and John Seely Brown. The origins of ubiquitous computing research at parc in the late 1980s. *IBM systems journal*, 38(4):693, 1999.

[WL14] Jiuqing Wan and Li Liu. Distributed data association in smart camera networks using belief propagation. *ACM Trans. Sen. Netw.*, 10(2):19:1–19:24, January 2014.

[WLP15] Si Wu, Robert Laganière, and Pierre Payeur. Improving pedestrian detection with selective gradient self-similarity feature. *Pattern Recognition*, 48(8):2364–2376, 2015.

[Wm15] W Hwu Wen-mei. *Heterogeneous System Architecture: A new compute platform infrastructure*. Morgan Kaufmann, 2015.

[WR09] Thomas Winkler and Bernhard Rinner. Pervasive smart camera networks exploiting heterogeneous wireless channels. In *Proceedings of the 2009 IEEE International Conference on Pervasive Computing and Communications*, PERCOM '09, pages 1–4. IEEE Computer Society, 2009.

[WS15] Dominik Wachholder and Chris Stary. Enabling emergent behavior in systems-of-systems through bigraph-based modeling. In *System of Systems Engineering Conference (SoSE), 2015 10th*, pages 334–339. IEEE, 2015.

[WW14] Jie Wu and Yunsheng Wang. *Opportunistic Mobile Social Networks*. CRC Press, 2014.

[WWW14] Fang Wu, Dian Hong Wang, and Yong Wang. Collaborative nodes strategy for target tracking in two-tier wireless multimedia sensor networks. *Journal of Networks*, 9(7), 2014.

[YPW+15] Shihong Yao, Shaoming Pan, Tao Wang, Chunhou Zheng, Weiming Shen, and Yanwen Chong. A new pedestrian detection method based on combined HOG and LSS features. *Neurocomputing*, 151:1006–1014, 2015.

[YSK+15] Reecha Yadav, Vinuchackravarthy Senthamilarasu, Krishnan Kutty, Vinay Vaidya, and Sunita Ugale. A review on day-time pedestrian detection. Technical report, SAE Technical Paper, 2015.

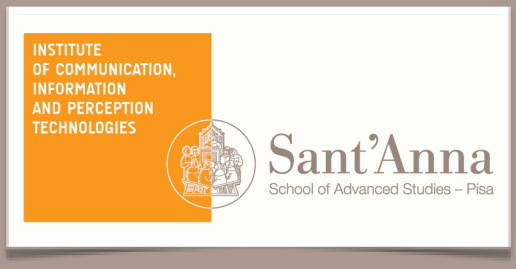
[ZBC+14] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, and Michele Zorzi. Internet of things for smart cities. *IEEE Internet of Things journal*, 1(1):22–32, 2014.

[ZF03] Barbara Zitova and Jan Flusser. Image registration methods: a survey. *Image and vision computing*, 21(11):977–1000, 2003.

[ZHYT11] Junge Zhang, Kaiqi Huang, Yinan Yu, and Tieniu Tan. Boosted local structured hog-lbp for object localization. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1393–1400. IEEE, 2011.

[ZLY14] Yaying Zhang, Xiuqing Lu, and Chen Ye. Distributed collaborative vehicle tracking in embedded smart camera networks. In *Dependable, Autonomic and Secure Computing (DASC), 2014 IEEE 12th International Conference on*, pages 214–218, Aug 2014.

[ZW94] Ramin Zabih and John Woodfill. Non-parametric local transforms for computing visual correspondence. In *European conference on computer vision*, pages 151–158. Springer, 1994.


INSTITUTE
OF COMMUNICATION,
INFORMATION
AND PERCEPTION
TECHNOLOGIES
Sant'Anna
School of Advanced Studies – Pisa