Noname manuscript No.
(will be inserted by the editor)

A semantic-guided and self-configurable framework for video analysis

Juan C. SanMiguel · José M. Martínez

Received: date / Accepted: date

Abstract This paper presents a distributed and scalable framework for video analysis that automatically estimates the optimal workflow required for the analysis of different application domains. It integrates several technologies related with data acquisition, visual analysis tools, communication protocols and data storage. Moreover, hierarchical semantic representations are included in the framework to describe the application domain, the analysis capabilities and the user preferences. The automatic determination of the analysis workflow is performed by selecting the most appropriate tools for each domain among the available ones in the framework by means of exploiting the relations between the semantic descriptions. The experimental results in the video surveillance domain demonstrate that the proposed approach successfully composes optimal workflows for video analysis applications.

Keywords Video analysis · Semantic analysis · Distributed framework · Automatic workflow composition · Self-configurable analysis

1 Introduction

Nowadays, advanced video analysis systems are expected to work in dynamic and different (but related) environments within a domain allowing the on-line addition or removal, when necessary, of services and analysis capabilities [26]. Specially, a growing demand has emerged in the video surveillance domain motivated by security issues in private and public places [18]. Their design presents many challenges related with scalability, portability and optimal allocation of resources. Most of the current systems are generally hand-crafted and task-specific. Hence, they are non-scalable and their deployment in different environments is limited requiring to undergo major structural changes in many situations.

Furthermore, a large amount of video processing algorithms are available as a consequence of the intensive research done during the past years. Complications arise for selecting an algorithm to perform a particular task as the algorithm performance depends on the operating conditions. As a result, the system may present high performance variations when deployed in different environments. For instance, different algorithms might be used depending on the scenario type (e.g., outdoor and indoor), the viewing distance (e.g., close and far) and the operation mode (e.g., on-line and off-line).

In this context, several notable efforts have been done to provide modular architectures for improving scalability and portability. Nevertheless, their design is based on a human operator who has to accumulate a great amount of experience related with video processing, network design, data management and so on. To simplify this task, several approaches have been proposed based on performance evaluation [19], available resources [25] and knowledge descriptions [16]. However, they are not fully automatic requiring the human intervention in most of the design stages.

In this paper, we address the above-mentioned limitations by proposing a scalable and distributed framework for video analysis that automatically estimates optimal workflows based on semantic information. This paper presents the combination of the enhancements achieved starting from previous work in the design of video analysis frameworks [31], knowledge representa-

Juan C. SanMiguel (✉) · José M. Martínez
Video Processing and Understanding Lab (VPULab)
Dept. Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid, Madrid (Spain)
E-mail: {juancarlos.sanmiguel, josem.martinez}@uam.es

tion [34] and dynamic workflows composition [35]. Firstly, the basic framework structure [31] is extended by including the support to semantic-based analysis. Secondly, the application domain and the analysis system are described by means of an extended ontology-based knowledge representation [34]. Specifically, we represent the *user* that operates with the results of the analysis process (e.g., a person, a retrieval system) as a set of preferences for such analysis and provide more detailed domain and system descriptions. Thirdly, automatic workflow composition and update are proposed for analyzing each domain based on these descriptions. For this composition, we extend the approach introduced in [35] in order to be able to select the most appropriate algorithm for performing a task when multiple choices are available by modeling this selection as a constraint satisfaction problem (CSP) [1]. Finally, we demonstrate the success of our approach for composing workflows for the video surveillance domain. Experimental results show that the best workflow is determined for each domain to successfully analyze the content also considering the user preferences (e.g., accuracy, speed).

The rest of this paper is organized as follows: section 2 reviews the related work and section 3 overviews the proposed framework. Then, section 4 presents the employed semantic descriptions and section 5 describes the workflow composition. Later, section 6 discusses the results. Finally, section 7 concludes this paper.

2 Related work

Several video analysis frameworks have been proposed by industry and academia. The requirements for designing such type of systems are the object of very active research [18]. In general, the following functionalities are desirable: 1) scalable and distributed systems, 2) real-time operation, 3) low resource consumption, 4) communication over standard networks and 5) run-time re-configuration. Traditionally, the building principles have been ad-hoc and based on expert knowledge. Thus, their portability to other settings is not easy in most of the situations. Although it is generally accepted that the semantic information can be used to improve the system performance [24,39], its successful application to video analysis is still in its early stages.

In the following sub-sections, we briefly review the existing frameworks for analysis of video events focusing on their characteristics and control of processing.

2.1 Characteristics of video event analysis frameworks

Existing approaches can be studied from several aspects. A classical distinctions consist of their purpose: they can be divided into generic and specialized. For instance, [38] proposed a framework for the video surveillance domain and [41] focused on the detection of stationary objects in underground stations. Another classification differentiates between distributed [25] and non-distributed frameworks [36]. Furthermore, they can be categorized depending on the existence of a centralized server to monitor the framework components. Initial research was focused on developing such servers for better management [40]. However, the scalability restriction motivated the design of decentralized frameworks with completely self-contained subsystems [2].

For communication issues, although most of existing approaches use their own communication protocol, some approaches use standard IP-based protocols such as RSTP [9], SOAP [13] and CORBA [36]. Moreover, the framework design is usually object-oriented and synchronous [29,38]. This approach can produce overhead at run-time and may cause communication bottlenecks. To avoid this limitation, the MASCOT method [40] was proposed to simplify the communication and allow asynchronous operation.

2.2 Control of processing

2.2.1 Manual control

Many efforts have been made to define the workflow of video applications. Current approaches provide modular architectures and specify control rules for managing the behavior of the modules [2,30,38]. They inherently support scalability and portability. However, they have some limitations due to the lack of well-defined interfaces for connecting modules and the application-dependent design of the proposed solutions (i.e., only focused on video surveillance). Hence, their use to develop video applications of diverse nature and the reuse of available algorithms is not straightforward.

Furthermore, there have been several proposals for composing generic multimedia workflows such as Microsoft Workflow Foundation (MWF) [11], Khoros [22] and GStreamer (GS) [17]. They provide intuitive end-user environments to facilitate the workflow design allowing a better understanding of it. They define interfaces for the processing modules to simplify their maintenance, reuse and update. Besides, they support parallelization and distribution. Nevertheless, they exhibit some limitations. For example, Khoros [22] does not allow iterative processes and introduces a communication

overhead between modules. MWF [11] and GS [17] are too generic requiring a great effort for developing complex applications. A common drawback is related with the dynamic behavior as they are not able to react to environment changes that may require to add or remove system components.

In addition to previously discussed limitations, manual control requires the human operator for many design tasks such as the selection of the processing modules and the appropriate algorithms as well as the specific implementation issues (e.g., resource mapping) for the different system deployments. Thus, it restricts the design to system developers or video processing experts.

2.2.2 Automatic control

Automatic control of processing aims to simplify the framework design and automate the analysis task. In current literature, we distinguish between methods based on Performance Evaluation (PE), Resource Mapping (RM) and Semantic Information (SI).

PE methods compute auto-critical functions based on the performance evaluation of the employed algorithms [7,19]. Their objective is to detect performance drops and behave accordingly (e.g., algorithm replacement, parameter adjustment). However, they evaluate performance by acquiring scene models based on ground-truth information. Therefore, their application to other settings is very restricted as they rely on training data. Furthermore, their algorithm description is restricted to input and output parameters (and their values) without containing any information about its functionality or usage. Thus, this control approach is semiautomatic as a human operator has to provide this information to define the analysis workflow.

RM methods deal with the mapping of algorithms onto resources of the framework. For instance, [25] described the complexity of the tasks to perform as their number of instructions and each processing node by its computational power. The transmission time is also considered as the available bandwidth and data exchanged. Then, a function is constructed to measure the cost of the analysis for each frame. Finally, the optimum solution to the task-node mapping is performed as a minimization over this cost function by using data about processing and transmission times for each available architectural solution. Similarly, [6] defined a reconfiguration strategy based on the load of the system processing units. Thus, the tasks are dynamically mapped onto the units that become idle. However, they do not provide solutions for adding or removing analysis capabilities. Similarly to PEs, RM methods also need the human operator to decide the structure of the task to perform and therefore, compose the analysis workflow.

SI approaches make use of semantics to explicitly or implicitly determine the structure of the framework. Explicit SI approaches define semantic-based sets of rules for selecting specific algorithms, to help the composition of workflows for video analysis. For example, [12] proposed to describe the objects and their recognition algorithms to compose simple workflows. However, it is limited to object analysis and the algorithm execution order is manually determined. Thus, this composition is semiautomatic. Similarly, [3] described an approach tailored to detect events for the soccer domain. Furthermore, [16] presented a knowledge-based controlled platform for video event analysis. However, algorithm selection is performed by the user and optimum algorithm selection is modeled as fine tuning using ground-truth data. Therefore, it has the previously mentioned drawbacks. Moreover, [27] proposed to compose workflows for simple object detection based on predefined descriptions of algorithm accuracy and user preferences. However, the structure of the workflow for each task is hand-coded and, therefore, the approach can not be automatically applied to different domains. Implicit SI approaches automatically learn the framework structure from semantic information. This information is usually given as a set of annotated training sequences. For example, [39] proposed to learn the structure of a Dynamic Bayesian Network (DBN) from training sequences annotated with semantic constraints.

Our approach fits into the explicit SI category. Compared with previous works, the major novelties of this paper are as follows. Firstly, a scalable and distributed framework provides a flexible environment for developing applications. Secondly, we extend a generic approach for providing a complete representation of the event-related semantics. Thirdly, a fully automatic composition of workflows is proposed to analyze different domains based on semantic representations of domain, system and user knowledge. Unlike existing approaches, it does not require the human intervention. Its main advantage consists in the separation of the design stages into the knowledge and algorithmic related parts. Thus, domain experts and algorithm designers can focus their efforts in the development of, respectively, more accurate knowledge models and algorithms. Table 1 compares our proposal against the reviewed literature.

3 Overview of the proposed framework

A scalable and distributed framework has been designed for video sequence analysis. We have selected [31] for defining the basic structure. Its main features are:

Ref.	Characteristics				Control		Knowledge support	
	Purpose	Portable	Extensible	Distributed	Mode	Type	Storage	Use
[25]	Generic	NA	NA	Yes	Automatic	Resource	No	No
[36]	Specific	No	No	No	Manual	-	No	No
[21]	Generic	NA	NA	Yes	Manual	No	Yes	NA
[2]	Generic	NA	Yes	Yes	Manual	-	Yes	NA
[12]	Specific	NA	NA	NA	Semi-automatic	Semantic	NA	Yes
[9]	Generic	NA	Yes	Yes	Manual	-	Yes	Yes
[19]	Specific	NA	NA	No	Automatic	Performance	No	No
[41]	Specific	NA	NA	Yes	Manual	-	NA	NA
[16]	Generic	NA	NA	NA	Semi-automatic	Semantic	Yes	Yes
[38]	Generic	NA	Yes	Yes	Manual	-	NA	NA
[30]	Generic	Yes	Yes	Yes	Manual	-	Yes	NA
[27]	Specific	NA	NA	No	Semi-automatic	Semantic	NA	Yes
Proposed	Generic	Yes	Yes	Yes	Automatic	Semantic	Yes	Yes

Table 1 Comparative of the reviewed frameworks for video analysis. (Key. NA: Not Addressed)

- Distributed environment for prototyping and deployment of multi-camera visual analysis systems.
- Modular and multi-threaded design for real-time processing at frame level.
- Flexible configuration (cascading or parallel interconnection of processing algorithms).
- Asynchronous client/server operation mode.

We extend this approach by defining the modules required for semantic-based analysis. This framework is divided in two levels of abstraction: physical and logical. They are described in the following sub-sections.

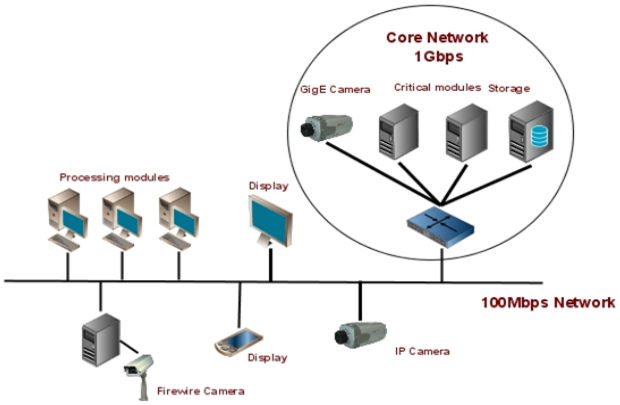

Fig. 1 Physical description of the proposed framework.

3.1 Physical part

The physical part (see Fig. 1) is composed of the required hardware: the cameras and a cluster of standard personal computers (PCs) connected together through a fast Ethernet network.

To cope with bandwidth restrictions and to allow operation at real-time, the framework architecture is composed of two networks. The main processing units are a set of rack-mounted standard PCs interconnected by a dedicated Gigabit Ethernet (core network). The other framework units (mainly processing modules) are distributed in a 100BaseT Ethernet network around the core network. Different types of cameras are plugged either to an acquisition card on a PC or directly to the Ethernet network for IP cameras. The computers are used to acquire the video, run algorithms and store the data. The main advantage of this architecture is its flexibility. Future needs in computing power can be addressed by simply adding PCs (or replacing existing ones with more powerful ones) in the cluster.

3.2 Logical part

The logical part is composed by three independent layers (see Fig. 2). Each layer is designed in a modular way and has a specific role. The different modules can be distributed in several ways allowing flexible configuration. The communication is based on a server/client model; the flow control is based on the TCP protocol. To avoid network congestion, data buffering between modules is supported at both sides. Depending on application requirements, layers can be combined into one single component with the required functionality.

3.2.1 Acquisition layer

This layer acquires the video from multiple video feeds and distributes video frame-by-frame to the entire framework using a server/client model. For performance issues, the captured data is stored in the processing layer (Shared Memory Module). Video frames are currently exchanged using baseline JPEG (ISO/IEC 10918-1) or

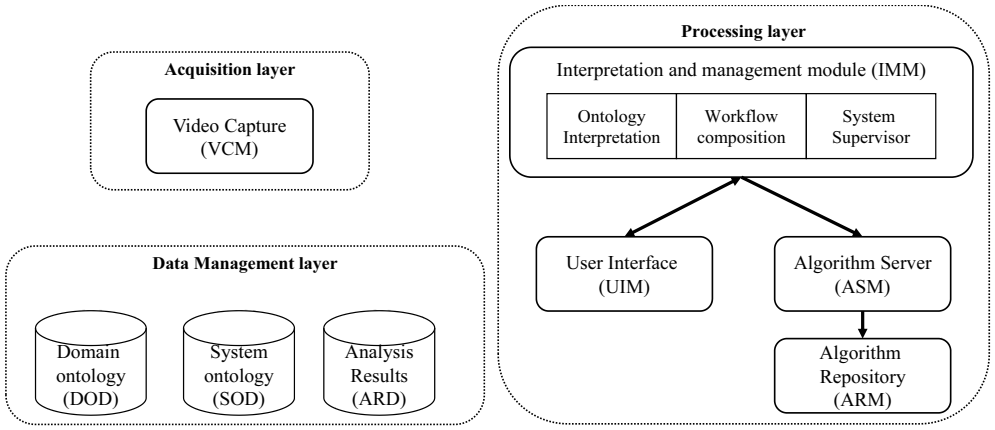

Fig. 2 Logical description of the proposed framework for video analysis

uncompressed format. A time stamp is attached to each frame at grabbing time to be used in the processing stage (e.g., tracking algorithms). Due to its modular design, the framework can easily support the addition of new camera connection protocols. Based on this framework, the system currently implemented handles IP, IEEE1394, GigE and USB protocols as well as input via video files.

3.2.2 Data Management layer

This layer is in charge of storing and distributing information required for analysis purposes. It is composed of three database sub-systems:

- The *Domain Ontology Database* (ODD) provides the domain knowledge. An overview of this information is given in section 4.
- The *System Ontology Database* (SOD) provides the capabilities of the system. The description of this information is defined in section 4.
- The *Analysis Results Database* (ARD) is in charge of managing the availability and intercommunication of analysis results between processing modules and allowing the distributed configuration of processing. Thus, this database stores the descriptions (e.g., metadata) generated by the analysis units making it available for further processing[1].

3.2.3 Processing layer

This layer analyzes the video content. A processing module corresponds to a system component responsible for some particular task not related to other layers (e.g. video analysis, video player). The modules run concurrently and asynchronously allowing to develop parallel and distributed applications. A modular design with common interfaces is defined for fast development of new algorithms within the framework. It communicates with the other layers to request and store data. Moreover, this layer includes several algorithms to solve the addressed analysis problems. They can be selected or combined depending on the application domain and the user preferences (as described in section 5).

Currently, this framework performs two tasks: semantic interpretation and video analysis, making use of the following modules (see Fig. 2):

- The *Interpretation and Management Module* (IMM) interprets the semantic information (domain and system), then combining it user preferences and finally requesting the execution of algorithms.
- The *Algorithm Server Module* (ASM) provides the processing capabilities of the framework. It makes the visual analysis tools usable through a server.
- The *Algorithm Repository Module* (ARM) indexes the available visual analysis tools and stores their compiled versions in order to provide the processing capabilities.
- The *User Interface Module* (UIM) manages the interaction with the content consumer (e.g., final user, software agent) obtaining the input from the consumer (e.g., domain to analyze) and providing the output to the consumer (e.g., video descriptions).

3.3 Analysis of a specific domain

For analyzing a specific domain, this framework performs a sequence of operations as follows:

1. *Initialization.* The UIM gets the necessary data for the analysis (e.g., domain to analyze, user preferences) and configures the IMM. Then, the IMM requests to the DOD and SOD modules the semantic information of the domain and the system.

[1] This database sub-system can be easily extended for developing query-based applications.

2. *Semantic-based workflow composition*
   (a) The IMM requests to the ASM the analysis tools available for the selected domain by using the data indexed in the ASM. Finally, the instances of the existing visual analysis tools are created and properly linked.
   (b) The IMM interprets the semantic system information to calculate the necessary resources (parameters) and to allocate memory for them in the ASM. Instances of the parameters are created and linked with the *Algorithm* instances.
   (c) The IMM interprets the ontology to select the necessary visual analysis tools between the available ones and their computation order. Then, this information is sent to the ARD (via the ASM) for the algorithm resources creation. This process is described in section 5.
3. *Analysis.* Finally, the IMM begins the sequential execution of the visual analysis tools via execution requests to the ASM. The analysis is performed until the video file has been finished or the system is turn off (for live on-line video analysis). Results obtained by each execution are stored in the ARD and are made available for further analysis or display purposes. During run-time operation, the update of the analysis workflow (addition or removal) is performed as described in section 5.

4 Semantic representation

For describing the video-related semantics, we have selected an ontology-based approach [34] that proposed a structured knowledge representation of the application domain and the analysis system. We extend this approach by detailing the domain-related context and the available algorithms as well as including the user preferences. In this section, we overview its structure and the proposed extensions.

4.1 Domain and System descriptions

The structure is composed of an upper ontology to define the hierarchy of each knowledge type that leaves explicit the information that has to be inserted for modeling. We use the *Scene* entity to represent the domain knowledge and the *System* entity to describe the analysis capabilities. Fig. 3 depicts their hierarchy.

Domain knowledge is described by means of hierarchical descriptions of the scene objects (*Object* entity), their relations (*Event* entity) and additional information (*SceneContext* entity). The *Object* entity represents the physical scene objects. *Mobile* and *Contextual* objects are distinguished by their ability to initiate motion. Furthermore, *Contextual* objects are divided into *Fixed* and *Portable* objects (if they can be displaced). Therefore, events can be defined considering relations with moving entities (e.g., person-meet), stationary objects (e.g., luggage-abandon) and fixed scene parts (e.g., door-enter). The *Event* entity represents spatio-temporal relations between *Object* entities. Each *Event* entity is related to *Object* entities by the *hasObjectList* property. Furthermore, it is sub-classed depending on the number of agents involved (single and multiple) and the temporal relation with its events (simple and complex). The *SceneContext* entity defines all the information that may influence the way a scene is perceived and can not be described using the *Object* and *Event* entities.

System knowledge represents the analysis capabilities (*Algorithm* entity), their inputs/outputs (*Parameter* entity) and their organization for performing tasks (*DetectionProcedure* entity). The *Algorithm* entity represents the available visual analysis tools in the system. They are used in the detection procedures defined for the detection of objects and events. Each *Algorithm* instance is related to input/output parameter instances by the *hasInputParameter*/*hasOutputParameter* property. The *DetectionProcedure* entity represents the available processing schemes in the system for detecting the concepts described in the ontology. Each *DetectionProcedure* instance is related to appropriate *Algorithm* instances by the *hasAlgorithm* property. Finally, the *Parameter* entity describes the different inputs and outputs of the algorithms available in the system. It is sub-classed according to the available *Algorithms*.

4.2 Extensions

4.2.1 SceneContext entity

Although the *SceneContext* entity was defined in [34], it was not suggested how this information should be applied to analysis. In this work, we propose to detail its description for using such information in the automatic workflow composition. The following properties of this entity are defined:

- *Type*: indicates the nature of the scene to be analyzed with the (string) values: *outdoor* and *indoor*.
- *View-distance*: indicates the distance to the observed activity of the scene with the (string) values: *close*, *inter* and *far*.
- *Time*: indicates the time of the scene to be analyzed. For simplicity purposes, we use the following (string) values: *day*, *night* and *all*.

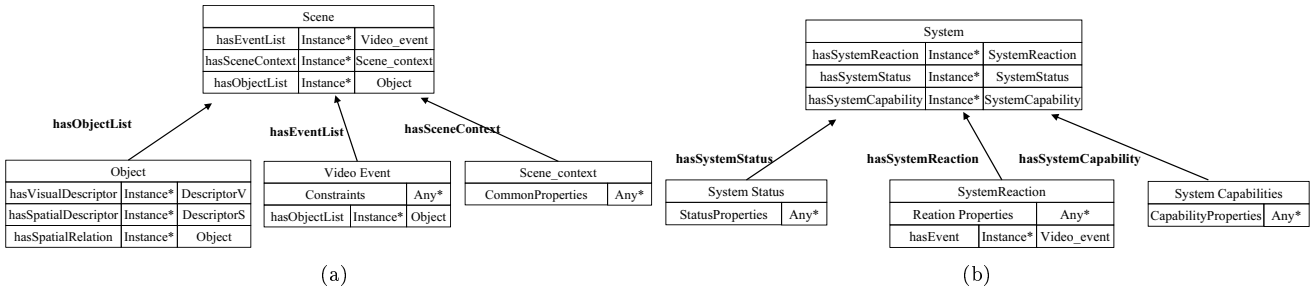

Fig. 3 Structured representation of (a) the domain knowledge (*Scene* entity) and (b) the system knowledge (*System* entity)

- *Crowded*: indicates if the scene is considered as a crowded environment with a boolean value.
- *ROI*: indicates the Region(s) Of Interest of the scene for their analysis. Currently, this information is indicated with a binary mask that indicates the ROIs with the value 1.

4.2.2 Algorithm entity

Regarding the *Algorithm* entity, inter-properties are used to connect each algorithm instance with its input and output parameters (*hasInputParameterList* and *hasOutputParameterList* properties). We propose to include some intra-properties for representing the domain and accuracy information. They are as follows:

- Domain properties. Similarly to the *SceneContext* entity, we define some properties to describe the application domain of the algorithm. They are *Type*, *View-distance*, *Time* and *Crowded*. Their values are the same as in section *4.2.1*.
- Accuracy properties. They characterize the accuracy of the algorithm. Currently, they are *processing-time*, *memory* and *accuracy*. Their possible values are *Low*, *Medium* and *High*.

Then, a hierarchy of the available *Algorithms* is defined to represent the tasks that the system can perform. Assuming that the focus of the ontology-based system is the recognition of human-related events, we have defined some categories to represent the common tasks performed within this domain. They are the following: *ImageAcquisition*, *ForegroundSegmentation*, *ShadowDetection*, *Pre-Processing*, *Post-processing*, *BlobExtraction*, *PeopleRecognition*, *GroupRecognition*, *Tracking*, *FeatureExtraction* and *EventAnalysis*.

Finally, the algorithm implementations available in the system are represented as instances of these categories. The estimation of the intra-properties of each instance can be done by using training data or human expert knowledge

4.2.3 User

We propose to include the user in the knowledge representation structure. It describes the final entity that manipulates the semantic information generated by the system. This entity can be a physical user, a query system, specific requirements for display purposes,... and it should include a description of the user interaction mode to request information to the system

As a first approach, we have defined a small set of properties to describe the user preferences. Currently, they correspond to the accuracy properties of the *Algorithm* entity. Therefore, a user may specify its preferences for *processing-time*, *memory* and *accuracy* of the system. If some properties are not specified, the highest value is assumed by default.

5 Semantic-based workflow composition

To overcome the current limitation of the ad-hoc design based on expert knowledge, we propose an automatic workflow composition for the analysis of a specific domain under certain user preferences using the visual tools available in the framework: algorithms (i.e., available techniques for solving a problem such as segmentation or shadow detection) and detection procedures (i.e., structured organization of algorithms for performing a task such as event recognition). The semantic descriptions defined in the previous section are inspected to select the most appropriate visual tools. In this section, we describe the semantic relations exploited and the workflow composition process.

5.1 Exploited entity relations

For providing such mechanism, we propose to use the properties of the entities that define domain and system knowledge to determine the visual analysis workflow for a specific modeling domain (i.e., the visual analysis tools and their associated execution order). This is performed by exploiting the relationships between the

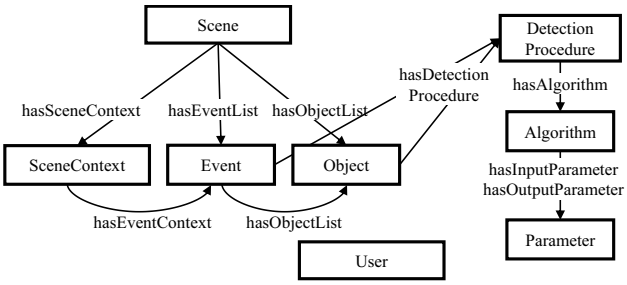

Fig. 4 Entity relations exploited for automatic workflow composition.

domain, the system and the user knowledge (depicted in Fig. 4). The key entities in this process are:

- *Object* entity that is related to *DetectionProcedure* entities by the *hasDetectionProcedure* property.
- *Event* entity that is related to *DetectionProcedure* entities by the *hasDetectionProcedure* property.
- *DetectionProcedure* entity that is related to *Algorithm* entities by the *hasAlgorithm* property.
- *Algorithm* entity that is related to *Parameter* entities by the *hasInputParameter*/*hasOutputParameter* property.
- *Parameter* entity that is sub-classed according to the available algorithms.
- *User* entity that describes the user preferences for the analysis (e.g., accuracy).
- *SceneContext* entity that defines the characteristics of the scenario (e.g., outdoor/indoor).

5.2 Automatic workflow composition

For composing the workflow, we extend the approach proposed by [35] that selects and orders the *Algorithm* entities to perform a specific task. In particular, we propose to solve the problem of selecting *Algorithm* instances for performing a task when multiple choices are available by modeling this process as a constraint satisfaction problem (CSP)[1].

Domain, self and user knowledge have to be properly defined prior to the composition of the workflow. Domain-knowledge represents the objects and events that we expect to observe in the modeled domain by means of the *Object* and *Event* entities. Their instances will be created during the domain analysis (i.e., when they are recognized) and these instances are not needed for composing the workflow. Self-knowledge is described by the *DetectionProcedure*, *Algorithm* and *Parameter* entities. As the *DetectionProcedure* entity defines the processing schemes (and not specific implementations), there is no need to create instances. However, *Algorithm* instances are needed to describe the current capabilities (e.g., two implementations of the foreground segmentation *Algorithm*). For the *Parameter* entity, no instances are needed for the workflow composition.

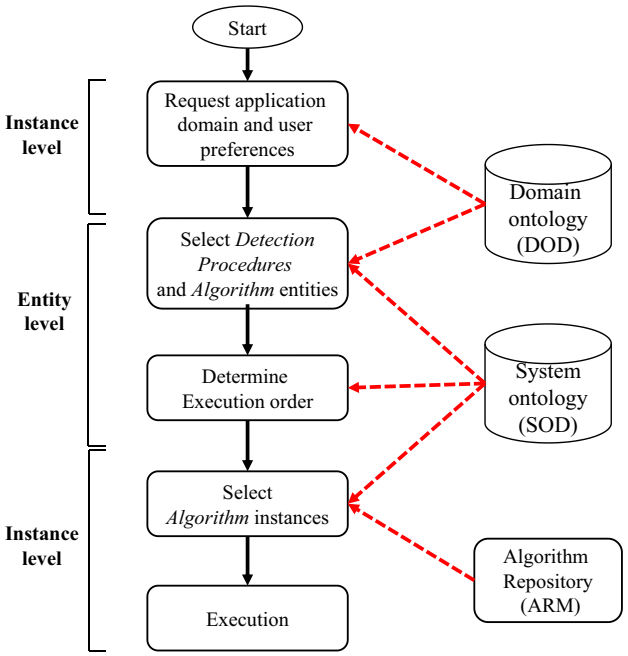

Fig. 5 Flowchart for automatic workflow composition. Red dashed lines indicate the use of semantic information.

Type	Properties
SceneContext	[Type=outdoor, View-distance=far, Time=day, Crowded=No, ROI=No, hasSpatialContext=null, hasObjectContext=null, hasEventContext=null]
User	[Processing-time=high, Memory=low, Accuracy=high]
Event	[Appear, Inside-zone]
Object	[Person, Car]

Table 2 Example of the data requested by the framework for automatic workflow composition.

The proposed composition method works at entity and instance levels. It is divided in four stages: data request, visual analysis tools selection (for *DetectionProcedure* and *Algorithm* entities), execution order determination and selection of *Algorithm* instances. Fig. 5 depicts the flowchart of the performed operations.

5.2.1 Data requesting

Firstly, the framework requests data for the application domain and the user preferences. Then, instances of the *SceneContext* and the *User* entities are created and used for composing the workflow. Furthermore, a domain description has to be available to define the entities to recognize (*Event* and *Object*). Table 2 depicts an example of such information.

1. **IF** an *Event* e_1 has objects $O = \{o_1, ..., o_n\}$ as a part of its description **AND** objects $O = \{o_1, ..., o_n\}$ have detection procedures $DP_O = \{dp_{O1}, ..., dp_{On}\}$ respectively **THEN** E_1 *hasDetectionProcedure* $DP_{E_1} = \{dp_{O1}, ..., dp_{On}\}$.
2. **IF** an *Event* e_1 has sub-events $SE = \{se_1, ..., se_n\}$ as a part of its description **AND** sub-events $SE = \{se_1, ..., se_n\}$ have detection procedures $DP_{SE} = \{dp_{SE1}, ..., dp_{SEn}\}$ respectively **THEN** e_1 *hasDetectionProcedure* $DP_{E_1} = \{dp_{SE1}, ..., dp_{SEn}\}$.
3. **IF** an *Object* o_1 has sub-objects $SO = \{so_1, ..., so_n\}$ as a part of its description **AND** sub-objects $SO = \{so_1, ..., so_n\}$ have detection Procedures $DP_{SO} = \{dp_{SO1}, ..., dp_{SOn}\}$ respectively **THEN** o_1 *hasDetectionProcedure* $DP_{O_1} = \{dp_{SO1}, ..., dp_{SOn}\}$.
4. **IF** a *DetectionProcedure* dp_1 has algorithms $A_{DP_1} = \{a_1, a_2\ a_3\}$ and as a part of its description **AND** a *DetectionProcedure* dp_2 has algorithms $A_{DP_2} = \{a_3, a_4\ a_5\}$ as a part of its description **THEN** the set of algorithms to use is $A = \{a_1, a_2\ a_3\ a_4\ a_5\}$.

Fig. 6 F-logic rules for selecting the visual analysis tools through exploiting the relationship between the entities of the system representation.

5.2.2 Visual analysis tools selection

This selection is performed by inspecting the properties of the sub-entities of the *Event* and *Object* entities defined for each domain. This phase should be considered as the integration of domain and system knowledge and it is automatically performed each time the framework is requested to analyze a specific domain. The aim of this stage is to extract the needed *Algorithms* entities, a_i, by inspecting the *DetectionProcedures* entities associated to each *Event* and *Object* entity.

This process is based on rules that exploit the transitivity properties between the entities defined in the ontology. These rules define the mapping between the visual analysis tools and the relevant entities to be detected in the modeled domain. Among the available choices in the literature, we have decided to use F-Logic [12] motivated by its easy use and understanding. Firstly, we have defined three rules to select all the necessary procedures (*DetectionProcedures* entities) to analyze a specific domain. Then, a fourth rule is included to select the *Algorithm* entities to apply from the selected *DetectionProcedures*. This rule is applied in pairs to all the selected detection procedures. Fig. 6 describes these four rules. Finally, the selected *Algorithm* entities conform the set of visual analysis tools to be executed and their properties (i.e., their inputs and outputs) are used to compute their execution order. Currently, each selected entity is used once in the composed workflow and, therefore, loops are not possible. For specific sequences of operations (e.g., loops), they have to be encapsulated into one single algorithm in order to be used by the proposed approach.

5.2.3 Execution order determination

After selecting the *Algorithm* entities for the domain analysis, their execution order is determined to define the analysis workflow of the framework. Its computation inspects the related *Parameter* entities (through the properties *hasInputParameter* and *hasOutputParameter* of each *Algorithm* entity). The key idea is to define a set of input parameters, select the *Algorithm* entities that can be used with this input set and study the possible relations between the selected ones. Priorities and sub-priorities are assigned depending on their relations to establish a sub-order of *Algorithms* entities with the same order. Then, the set of input parameters is extended with the output parameters of the selected *Algorithm* entities and the process is repeated with the new input set. This process is done from the minimum set of inputs, composed by the input image (named *frame-rgb*), until the list of selected *Algorithm* entities is finished. Prior to detailing the algorithm for computing the execution order, we define the following sets, algorithm types and operations on them:

Definition 1 $\mathbb{A}$ is a set that represents the *Algorithm* entities selected in the visual tool selection process. $\mathbb{P}$ represents a set of *Parameter* entities. p_j and a_i describe, respectively, a specific *Parameter* or *Algorithm* entity. $\mathbb{A}_\mathbb{A}$, $\mathbb{A}_\mathbb{I}$ and $\mathbb{A}_\mathbb{S}$ are three sets that contain selected *Algorithm* entities for the operations of Accumulation, extraction from a set of Input parameters and the determination of the Sub-order. The execution order is represented by the integer variable o.

Definition 2 For *Algorithm* entities that has the same execution order, we distinguish:

- Filtering *Algorithms*: they have the same input and output ($a_i \subset \mathbb{A}_\mathbb{S} / Input(a_i) \equiv Output(a_i)$).
- Processing *Algorithms* type 1: they do not have the same input and output. Additionally, their output is contained in their input ($a_i \subset \mathbb{A}_\mathbb{S} / Input(a_i) \neq Output(a_i)$ AND $Output(a_i) \subset Input(a_i)$).
- Processing *Algorithms* type 2: they do not have the same input and output. Additionally, their output is not contained in their input ($a_i \subset \mathbb{A}_\mathbb{S} / Input(a_i) \neq Output(a_i)$ AND $Output(a_i) \not\subset Input(a_i)$).

Definition 3 For operating with *Algorithm* and *Parameter* instances, we define the following functions:

- $Input(a_i) = \{p_j \subset \mathbb{P} / a_i\ hasInputParameter\ p_j\}$
- $Output(a_i) = \{p_j \subset \mathbb{P} / a_i\ hasOutputParameter\ p_j\}$
- $card(\mathbb{A})$ =number of elements in the set $\mathbb{A}$
- $AssignOrder(o, a_i) \Rightarrow$ assigns the execution order o to the algorithm i.

The full execution order determination procedure is described in the Algorithm 1.

Algorithm 1 Execution order determination.

Input: Domain knowledge description D and selected *Algorithm* entities $\mathbb{A} = \{a_i\}$.
Output: Order o_i of each Algorithm entity a_i

begin
  Set $\mathbb{A}_\mathbb{F} = \{\emptyset\}$,$\mathbb{A}_\mathbb{S} = \{\emptyset\}$ and $o = 1$//Variable initialization
  Set $\mathbb{P} = \{frame\_rgb\}$//raw image as initial input parameter
  While $\mathbb{A}_\mathbb{S} \neq \mathbb{A}$
    $\mathbb{A}_\mathbb{I} = \{a_i \subset \mathbb{A} / Input(a_i) \equiv \mathbb{I}\}$ //select all algorithms that have determined input parameters
    if $card(\mathbb{A}_\mathbb{I}) = 1$**then**
      $AssignOrder(o, a_i)$
      $o = o + 1$
      $\mathbb{A}_\mathbb{S} = \mathbb{A}_\mathbb{S} \cup \mathbb{A}_\mathbb{I}$
    else
      //Determine the type of the *Algorithm* entities
      for each filtering algorithm $a_j \subset \mathbb{A}_\mathbb{I}$ **do**
        $AssignOrder(o, a_j)$
        $\mathbb{A}_\mathbb{S} = \mathbb{A}_\mathbb{S} \cup \{a_j\}$
      end for
      $o = o + 1$
      for each processing algorithm type 1 $a_j \subset \mathbb{A}_\mathbb{I}$ **do**
        $AssignOrder(o, a_j)$
        $\mathbb{A}_\mathbb{S} = \mathbb{A}_\mathbb{S} \cup \{a_j\}$
      end for
      $o = o + 1$
      for all processing algorithm type 2 $a_j \subset \mathbb{A}_\mathbb{I}$ **do**
        $AssignOrder(o, a_j)$
        $\mathbb{A}_\mathbb{S} = \mathbb{A}_\mathbb{S} \cup \{a_j\}$
      end for
      $o = o + 1$
    end if
    Set $\mathbb{A}_\mathbb{A} = \mathbb{A}_\mathbb{A} \cup \mathbb{A}_\mathbb{S}$ and $\mathbb{A}_\mathbb{S} = \{\emptyset\}$//Accumulate the processed algorithms in $\mathbb{A}_A$
    Set $\mathbb{P} = \{frame\_rgb, Output(\mathbb{S})\}$//Update the process input parameters
  end while
end

5.2.4 Selection of Algorithm *instances*

After selecting the *Algorithm* entities and computing their execution order, *Algorithm* instances have to be chosen for composing the workflow. We use the prior knowledge about the domain to be analyzed (*SceneContext* entity), the constraints imposed by the user (*User* entity) and the existing *Algorithm* instances.

We propose to model this selection as a constraint satisfaction problem (CSP) [1]. Thus, we define the satisfaction problem as a triple $\langle X, D, C \rangle$ where $X = X_D \cup X_A = \{x_1, ..., x_M, x_{M+1}, ..., x_{M+N}\}$ is a superset that describes the properties of the *Algorithm* instances. It is composed of the set $X_D = \{x_1, ..., x_M\}$ that describes the domain-related properties and the set $X_A = \{x_1, ..., x_N\}$ that describes the accuracy-related properties. $D = \{d_1, ..., d_{N+M}\}$ is the set of $N + M$ domain values for each property (i.e., possible values). Hence, the properties of an *Algorithm* instance j of an entity i, t_{ij}, are defined as a mapping $V_{ij} : X \rightarrow D$. Constraints are represented as pairs $\langle T, R \rangle$ where T is a $(M + N)$ set of properties (i.e., the intra-properties of the *SceneContext* and the *User* entity) and R is a $(M+N)$-ary relation on D. We assume the same number of elements in the sets X and T (i.e., all the properties of the *Algorithm* instances and the constraints are defined), and the same listing order of these properties. Furthermore, we consider one constraint for each application domain to be analyzed.

Then, instead of looking for an *Algorithm* instance that completely satisfies a constraint, we define a global scoring function F for each instance j of the *Algorithm* entity i to provide a satisfaction score as follows:

$$Score_{ijd} = F(V_{ij}, C) \tag{1}$$

where V_{ij} represents the set of valued properties of the *Algorithm* instance t_{ij} and C is the set that describes the constraint in terms of the domain properties and the user preferences. For a more readable notation, we have omitted the sub-indexes i and j for describing an *Algorithm* instance, and used V instead of V_{ij}. Moreover, we also use v_m instead of $v_m(V)$ for representing the m property of the *Algorithm* instance V. Then, the global function F is defined as follows:

$$F(V, C) = \sum_{m=1}^{m=M} f_d(v_m, c_m) + \sum_{n=M+1}^{n=M+N} f_a(v_n, c_n) \tag{2}$$

where f_d and f_a are the local scoring functions for, respectively, the domain and the accuracy properties; V and C describe, respectively, the *Algorithm* instance and the constraint; v_m and c_m represent their domain properties; v_n and c_n represent, respectively, their accuracy properties.

The domain local scoring function assigns a score considering the domain properties of the *Algorithm* instance and the constraint. It is defined as follows:

$$f_d(v_m, c_m) = \begin{cases} 0 \; if \;\; v_m = c_m \; OR \; c_m = \{any\} \; OR \\ \qquad\qquad\qquad v_m = \{any\} \\ 1 \; if \qquad\qquad v_m \neq c_m \end{cases} \tag{3}$$

For the accuracy local function, we first transform the values of the properties $D_a = d_{M+1,...N+M}$ (*Low*, *Medium* and *High*, see section 4) into a scalar domain using a simple relation, $s : D_a \rightarrow \mathbb{N}$, defined as follows:

$$s(d_n) = \begin{cases} 1 \; if \quad d_n = \{Low\} \\ 2 \; if \; d_n = \{Medium\} \\ 3 \; if \quad d_n = \{High\} \end{cases} \tag{4}$$

Instances of Algorithm i	Domain properties				Accuracy properties		
	Type	View	Time	Crowded	Proc.Time	Memory	Accuracy
Method 1	Outdoor	Far	Day	No	Low	Low	Medium
Method 2	Outdoor	Close	Day	No	Medium	Medium	Medium
Method 3	Indoor	Inter	Day	No	Medium	High	Medium
Method 4	Any	Inter	Day	Yes	High	Medium	High

(a)

Domain	Scene Context			
	Type	View	Time	Crowded
D0	Indoor	Inter	Day	No

User	User Preferences		
	Proc.Time	Memory	Accuracy
U0	Medium	Low	Medium

(b)

Instances for Algorithm i	Domain scores (f_d)				Accuracy scores (f_a)			Score $F(V,C)$
	Type	View	Time	Crowded	Proc.Time	Memory	Accuracy	
Method 1	1	1	0	0	0	0	0	2
Method 2	1	1	0	0	0	1	0	3
Method 3	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**
Method 4	0	0	0	1	1	1	1	4

(c)

Fig. 7 Example of the proposed method for selecting an instance of an *Algorithm* entity. Data corresponds to (a) the *Algorithm* instances descriptions, (b) the *SceneContext* and *User* (preferences) entities and (c) instance scores (final selection is marked in bold).

Then, this local scoring function is defined as:

$$f_a(v_n, c_n) = \begin{cases} 1 \; if \; s(v_n) - s(c_n) > 0 \\ 0 \; if \; s(v_n) - s(c_n) \leq 0 \end{cases} \quad (5)$$

where $f_a(.)$ assumes that the value property *High* is the worst case and the value *Low* is the best case. Observe that this assumption is true for the *Processing-time* and *Memory* properties. However, it has the opposite meaning for the *Accuracy* property (*High* value is the best case). In this situation, we just switch the conditions to invert the result of the scoring function.

Finally, instance selection is performed by using the minimum a posteriori criterion:

$$V_i^{sel} = \underset{j}{argmin}(Score_{ijd}) \quad (6)$$

where $Score_{ijd}$ is the satisfaction score of the instance j of the *Algorithm* entity i for a d domain. If more than one instances are selected, we accumulate the difference of their *Accuracy* properties ($s(v_n) - s(c_n)$) to decide which one is the most suitable for the analysis.

Fig. 7 depicts an example of the process for selecting instances of an *Algorithm* entity. As it can be observed, four instances are available with different domain and accuracy properties (see Fig. 7(a)). Then, the information corresponding to the *SceneContext* and *UserPreferences* entities is provided for a specific domain (D0) and user (U0) as shown in Fig. 7(b). Finally, the method 3 is chosen for composing the workflow after applying the global scoring function. Fig. 7(c) shows the scores obtained for each available instance.

5.3 On-line workflow update

The on-line insertion and removal of new analysis tools into the frameworks is performed by the user or other applications via the UIM module. The insertion operation is performed by adding the new data (domain or self knowledge), creating the instances corresponding to the new data and computing the execution order of each new visual analysis tool added. If new capabilities are introduced for an existing *Algorithm* entity, the scoring function is applied to it and if its score is lower than the current instance being used, the added one is incorporated in the workflow and the other is removed. If new domain knowledge is inserted (e.g., new events to detect), the entire process has to be repeated for composing a new workflow. Similarly, the removal operation differs whether it affects to domain or self knowledge. A removal of domain knowledge will require to recompute the composition of the workflow. A removed *Algorithm* instance will be replaced by the available instance with the lowest score. Its main advantage is that the remaining tools (the ones that are not removed) are not eliminated from the workflow avoiding the destruction and creation of resources. In conclusion, real-time workflow update is can be achieved for including or removing analysis capabilities.

6 Experimental validation

To evaluate the proposed approach, we present three experiments focused on detecting abandoned objects for video surveillance. The first one considers the analysis of different domains. The second one studies its performance under variations of the user preferences. Finally, the last one includes and removes processing capabilities. In addition, we provide a comparison with a state-of-art approach that has been designed manually. In this section, we describe the common setup for all the experiments (the evaluation criteria, the available instances and the workflow at entity-level), their definition and the obtained results.

6.1 Setup

The proposed approach has been implemented in C++ using OpenCV[2] for video analysis and in Java (only the IMM module) using the OWL Protegé API[3] for ontology handling. Tests were performed on two PCs (P-IV 2.8GHz and 1GB RAM) connected via a Gigabit LAN (respectively used for ontology and video processing).

For comparison purposes, we have selected a state-of-art approach for detecting abandoned objects in video surveillance [32] (from now on *fixed workflow*). It represents the related literature that is manually designed based on the expert knowledge of the task. It is composed of a sequential combination of the following stages: foreground detection, noise removal, blob extraction, blob tracking, static object detection, people recognition and event detection. Further details of the techniques implemented for each stage are provided in [32].

For evaluating the event detection accuracy, we use the Precision (P) and Recall (R) measures. Precision is the ratio between the correct and the total number of detections. Recall is the ratio between the correct detections and the total number of annotations. We have defined an annotated event as detected if there is a detection that satisfies the following constraints: the overlapped duration in frames between them is more than 50% and the mean overlapped area between them is more than 50% (calculated in the overlapped frames).

6.2 Processing library

A library of visual analysis tools is available for domain analysis. As a first approach, we have focused on the abandoned object detection task for video surveillance. Table 3 lists the currently implemented tools.

[2] http://sourceforge.net/projects/opencv/
[3] http://protege.stanford.edu/

For the *Algorithm* entity, we have described the common analysis stages (foreground detection, shadow removal, blob extraction, blob tracking, people recognition and event analysis). Regarding the event analysis, we have implemented a basic rule-based approach for detecting the events of interest (e.g., abandoned object) similarly to [32]. It defines a set of rules over the data generated by the preceding analysis stages of the event detection. The recognition of complex events is avoided as it is out of the scope of this paper. Then, instances of the corresponding *Algorithm* entities are created for each implemented technique. For example, four instances are defined for the *PeopleRecognition* entity to describe the implementations based on aspect ratio [14], ellipse fitting [14], shoulder location [14] and edges [15]. For the *Parameter* entity, we have sub-classed this entity to define the inputs/outputs of the *Algorithm* entities.

For the *DetectionProcedure* entity, we have included entities to describe the processing schemes for detecting the defined *Object* and *Event* entities. Appropriate links to *Algorithm* entities are established by using the *hasAlgorithm* property. Moreover, they are also assigned to the corresponding *Object* and *Event* entities by using the *hasDetectionProcedure* property.

6.3 Workflow creation at entity level

For the detection of specific objects and events (e.g., abandoned objects), the building of the workflow at entity level is possible as it only requires the description of the system capabilities. In other words, it is able to select and order the required analysis stages (entities). Then, the workflow at instance level (i.e., the particular algorithm for each stage) has to be selected depending on the domain properties and the user preferences. In this sub-section, we describe this workflow creation at entity level that is later used as starting point in the three experiments.

Visual analysis system creation First, the *hasDetectionProcedure* property of the *Person* and *Inside-zone* entities is examined by using the first three rules defined in sub-section 5.2.2. Then, all *DetectionProcedures* are listed and the repeated ones are eliminated. Finally, algorithm selection is easily performed by applying the fourth rule defined in sub-section 5.2.2 to all the *DetectionProcedures* listed. As a result of this procedure, the following *Algorithm* entities are selected: foreground detection, shadow elimination, connected component analysis, blob tracking, stationary blob detection, people recognition and the corresponding routine that models the abandoned object event.

Algorithm instances	Domain properties				Accuracy properties		
	Type	View	Time	Crowded	Proc.Time	Memory	Accuracy
***ForegroundDetection* entity**							
SG [20]	Indoor	Any	Day	No	Low	Low	Low
Gamma [20]	Indoor	Any	Day	No	Low	Low	Medium
Filtering+Gamma[10]	Indoor	Any	Day	Yes	Low	Low	Medium
GMM [20]	Any	Any	Day	No	Medium	Medium	Medium
***ShadowRemoval* entity**							
Deterministic non-model (HSV)[28]	Any	Any	Day	Any	Low	Low	Medium
Statistical non-parametric (RGB) [28]	Any	Any	Day	Any	High	Low	High
Adaptive-HSV (AHSV)[33]	Any	Any	Day	Any	High	Medium	High
***BlobExtraction* entity**							
Connected component (CC)[37]	Any	Any	Day	Any	Low	Low	High
Spatial-blob distance [23]	Any	Inter-Far	Any	No	Low	Low	Low
Color-blob distance [23]	Any	Inter	Any	No	Low	Low	Medium
Kalman [23]	Any	Inter-Far	Day	No	Low	Medium	Low
Meanshift [23]	Any	Close-Inter	Day	Yes	Medium	Medium	High
***PeopleRecognition* entity**							
Edge [15]	Any	Inter	Day	No	Medium	Medium	High
Ellipse [14]	Any	Far	Day	No	Medium	Low	Medium
Aspect ratio [14]	Any	Inter	Day	No	Low	Low	Low
Ghost [14]	Any	Close	Day	No	High	Low	Medium
***StationaryBlobDetection* entity**							
No-tracking-based [5]	Any	Any	Day	Yes	Medium	Medium	High
Tracking-based [4]	Any	Inter	Day	No	Low	Medium	Low
Sampling-based [4]	Any	Inter-Far	Day	No	Medium	Medium	Low

Table 3 Summary of visual analysis tools available in the proposed framework.

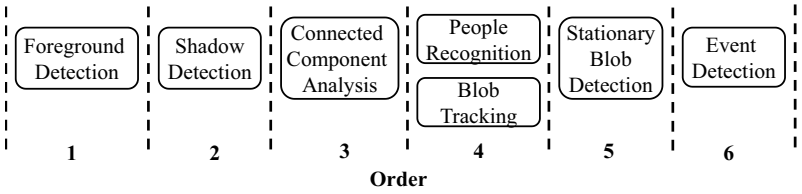

Fig. 8 Composed workflow at entity level for the experiments (selected *Algorithm* entities and their execution order).

Execution order determination As described in subsection 5.2.3, the *hasInputParameter* and *hasOutputParameter* properties are used to determine the execution order. First, the selected *Algorithms* that can be applied using the initial *Parameter* (i.e., *frame-rgb*) are examined. As a result, *ForegroundDetection* is selected as the first entity. Then, the second phase selects the *ShadowRemoval* and *BlobExtraction* entities. Hence, rules for collision are applied to determine that the former is applied in the first place (Rule 1) before the *BlobExtraction*. A third phase selects the *BlobTracking* and the *PeopleRecognition* entities. They are identified as type 2 (see sub-section 5.2.3) so their execution order is the same (i.e., they can be executed in parallel). Then, *StationaryBlobDetection* is selected as it uses data from the *BlobTracking* entity. Finally, detection routines for the selected event are included in the last order. Fig. 8 depicts the obtained workflow. Observe that the structure of the created workflow at entity level is very close to the *fixed workflow* of [32]. However, it has been automatically computed without requiring prior knowledge from the application designer as opposed to [32]. Furthermore, an unnecessary stage has been removed (noise removal) and an additional stage has been included (shadow detection) to maximize the current analysis capabilities. Its main advantage is the identification of the analysis stages that can be run in parallel or in sequential mode and its capability to select the optimum algorithms for each situation (e.g., application domain, user preferences).

Domain	Scene Context				Entities to detect		Dataset		
	Type	View	Time	Crowded	Event	Object	Dataset	#sequences	#events
D1	Indoor	Inter	Day	Yes	Abandoned-object	-	AVSS2007	4	8
D2	Indoor	Inter	Day	No	Abandoned-object	-	PETS2006	10	10
D3	Outdoor	Far	Day	No	Abandoned-object	-	CANTATA	9	12
D4	Outdoor	Inter	Day	No	Abandoned-object	-	HERMES	4	8

(a)

User	User Preferences		
	Proc.Time	Memory	Accuracy
U0	Low	Low	Medium

(b)

Fig. 9 Input data for composing the workflow in different domains for abandoned object detection (first experiment). Data corresponds to the (a) domain descriptions (in terms of the *SceneContext* and the associated content) and (b) the *User* entity.

6.4 Experimental results

In this sub-section, we describe the results of the three experiments and a computational cost comparative.

6.4.1 Analysis of different domains

For the first experiment, we have modeled four domains that represent real scenarios for detecting abandoned objects in video surveillance. The first two (D1 and D2) consist on indoor sequences at an intermediate view-distance with varying densities of moving objects (D1 is crowded whereas D2 is not). In particular, we have selected sequences from the AVSS2007[4] and PETS2006[5] datasets for, respectively, D1 and D2. The other two domains (D3 and D4) represent outdoor sequences at, respectively, intermediate and far view-distance. Data for D3 and D4 has been selected from, respectively, the CANTATA[6] and the HERMES[7] datasets. Fig. 9 summarizes the four domain models, the available content for abandoned object detection and the modeled user (U0). Sample frames are shown in Fig. 10.

Starting from the workflow determined at entity-level in sub-section 6.3 for the abandoned object detection task, the full workflow creation (i.e., at instance-level) requires the selection of the appropriate *Algorithm* instances. As proposed in sub-section 5.2.4, we perform this selection as a CSP problem and compute the scores of the available instances to measure their suitability for each domain. Finally, the instances with lowest scores are selected for the execution. Table 4 summarizes these results. Moreover, Table 5 shows an example of selection process for the D1 domain. As it can be observed, the computed satisfaction scores measure their suitability for this particular domain. For each type of entity, the lowest scores indicate the instances to be selected for the analysis of the D1 domain under the user preferences defined in Fig. 9.

Fig. 10 Sample frames for the modeled domains. From top-left to bottom-right: D1 (AVSS2007 dataset), D2 (PETS2006), D3 (CANTATA) and D4 (HERMES) domains.

Table 6 presents and compares the event detection results of the fixed and the composed workflows for each domain. Although the user preferences were set to an intermediate level of accuracy, the proposed approach outperformed the *fixed workflow* in all the analyzed domains. The proposed automatic composition introduced a dynamic behavior that enhanced the *fixed workflow* in two aspects. First, it modified the workflow structure by including additional processing stages from available capabilities (e.g., shadow detection) and by removing unnecessary stages (e.g., noise removal in the *fixed workflow*). Second, the selection of the optimum algorithms (for each stage and domain) allowed to maximize the performance of the tools employed and, therefore, improved the outcome of the resulting workflow. Among the major effects, a significant increase of the precision is observed (i.e., a reduction of the number of wrong detections). It can be explained due to the

[4] http://www.avss2007.org/
[5] http://www.cvg.rdg.ac.uk/PETS2006/
[6] http://www.multitel.be/~va/cantata/LeftObject/
[7] http://iselab.cvc.uab.es/indoor-cams

Algorithm entity	Selected instances for each domain			
	D1	D2	D3	D4
ForegroundDetection	Filtering+Gamma	Gamma	GMM	GMM
ShadowRemoval	HSV	HSV	HSV	HSV
BlobExtraction	CC	CC	CC	CC
BlobTracking	Spatial-blob distance	Spatial-blob distance	Color-blob distance	Spatial-blob distance
PeopleRecognition	Edge	Edge	Ellipse	Edge
StationaryBlobDetection	No-tracking-based	Tracking-based	Sampling-based	Tracking-based
EventDetection	Abandoned object detection	Abandoned object detection	Abandoned object detection	Abandoned object detection

Table 4 Composed workflows for the modeled domains in the first experiment. Data describes the selected instances (i.e., specific algorithm) for each *Algorithm* entity (i.e., processing stage).

Algorithm instances	Domain scores (f_d)				Accuracy scores (f_a)			Score
	Type	View	Time	Crowded	Proc.Time	Memory	Accuracy	$F(V,C)$
ForegroundDetection								
SG [20]	0	0	0	1	0	0	1	2
Gamma [20]	0	0	0	1	0	0	0	1
Filtering+Gamma[10]	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
GMM [20]	0	0	0	1	1	0	0	2
***ShadowRemoval* entity**								
Deterministic non-model (HSV)[28]	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Statistical non-parametric (RGB) [28]	0	0	0	0	1	0	0	1
Adaptive-HSV[33]	0	0	0	0	1	1	0	2
***BlobExtraction* entity**								
Connected component [37]	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
***BlobTracking* entity**								
Spatial-blob distance [23]	0	0	0	1	0	0	1	2
Color-blob distance [23]	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**
Kalman [23]	0	0	0	1	0	1	1	3
Meanshift [23]	0	0	0	0	1	1	0	2
***PeopleRecognition* entity**								
Edge [15]	0	1	0	1	1	1	0	4
Ellipse [14]	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**2**
Aspect ratio [14]	0	1	0	1	0	0	1	3
Ghost [14]	0	1	0	1	1	0	0	3
***StationaryBlobDetection* entity**								
No-tracking-based [5]	**0**	**0**	**0**	**0**	**1**	**1**	**0**	**2**
Tracking-based [4]	0	0	0	1	0	1	1	3
Sampling-based [4]	0	0	0	1	1	1	1	4

Table 5 Algorithm instance selection for the D1 domain. Data correspond to the satisfaction scores for each instance (final selection is marked in bold).

inclusion of the shadow removal analysis as well as the use of the appropriate algorithms for foreground segmentation and stationary blob detection for each analyzed domain. Their main advantage in the application is the reduction of the amount of data to process and, therefore, it decreases the likelihood of wrong event detection. The low accuracy of the obtained results for the D1 domain is explained by the difficulty of the proposed task in crowded environments.

Domain analyzed	Workflow approaches			
	Fixed		Proposed	
	P	R	P	R
D1	.20	.37	.40	.50
D2	.53	.70	.72	.80
D3	.63	1	.80	1
D4	.50	.75	.70	.75
Average	.46	.71	.65	.76

Table 6 Event detection results for the domains modeled in the first experiment.

Algorithm entity	Selected instances for each modeled user		
	U1	U2	U3
ForegroundDetection	GMM	GMM	Gamma
ShadowRemoval	A-HSV	RGB	HSV
BlobExtraction	CC	CC	CC
BlobTracking	Spatial-blob distance	Color-blob distance	Color-blob distance
PeopleRecognition	Edge	Ellipse	Ellipse
StationaryBlobDetection	No-tracking-based	No-Tracking-based	Tracking-based
EventDetection	Abandoned object detection	Abandoned object detection	Abandoned object detection

Table 8 Composed workflows for the modeled users in the second experiment. Data describes the selected instances (i.e., specific algorithm) for each *Algorithm* entity (i.e., processing stage).

User	User Preferences		
	Proc.Time	Memory	Accuracy
U1	High	High	High
U2	Medium	Medium	Medium
U3	Low	High	Medium

Table 7 Different users modeled for composing the workflow to analyze the D3 domain (second experiment).

Workflow approach	User	Accuracy		Cost (ms/frame)	Memory (MB)
		P	R		
Fixed	-	.63	1	65	215
Proposed	U1	.85	1	90	315
	U2	.80	.91	70	240
	U3	.66	1	45	280

Table 9 Results of the different user preferences defined in the second experiment for the analysis of the D3 domain.

6.4.2 Variation of user preferences

In the second experiment, we study the performance of the proposed approach under different user preferences. In particular, we have selected the D3 domain and we have defined the preferences of three users focused on the following criteria: processing time, memory consumption and accuracy. The first user is centered on the accuracy allowing high computational cost and memory consumption. The second user has medium level preferences for the three criteria. The third user is concerned about the processing time with an intermediate accuracy (without restricting memory consumption). A description of such preferences is provided in Table 7. This experiment is useful to understand the adaptation of the proposed approach to different aspects of the analysis preferred by the user.

After defining the users, the satisfaction scores required for instance selection are computed using each set of user preferences and the entity-level workflow described in sub-section 6.3. Here, only the accuracy scores of each instance are affected by the preferences defined by each user. As the domain to analyze is the same for the four users, the domain scores remain the same. The selected instances are listed in Table 8.

Table 9 presents the results of the second experiment. As is shown in the table, the proposed approach is able to adjust its capabilities and consumption according to the user preferences. Compared to the fixed approach [32], the produced workflows showed better performance for each preference imposed by the user (i.e., high accuracy and low processing time). The workflow of the selected state-of-art approach obtained results regardless the user preferences showing the limitations of its fixed structure for selecting the focus of the analysis. For the user U1, the composed workflow obtained the best accuracy among the compared ones (the workflows generated for the other users and the fixed one). Moreover, the resulting workflow for the user U3 highly reduced the execution time of the analysis with respect to the *fixed workflow* and without decreasing the accuracy. However, an impact in the accuracy is noticed as the focus of this workflow is on the processing time. Finally, the workflow for the U2 user had intermediate values in the three evaluated aspects.

6.4.3 Modification of capabilities

In the third experiment, we test the proposed approach against changes of the available capabilities of the framework. Two types of changes are possible at entity or instance level if they are related to, respectively, stages or algorithms. The former affects to the structure of the workflow stages whereas the latter influences in the selection of algorithms for a particular stage. Here, we concentrate on both changes by including new algorithms (instances) and removing existing stages (entities). For this experiment, we use the description of the D3 domain and the U1 user.

Addition For this situation, we include a new algorithm (i.e., instance) in the framework for the *ForegroundDetection* entity. In particular, we have included the KDE algorithm [20] in the Algorithm Server Module (ASM)

Algorithm instances	Domain scores (f_d)				Accuracy scores (f_a)			Score
	Type	View	Time	Crowded	Proc.Time	Memory	Accuracy	$F(V,C)$
ForegroundDetection								
Single Gaussian [20]	1	0	0	0	0	0	1	2
Gamma [20]	1	0	0	0	0	0	1	2
Filtering+Gamma[10]	1	0	0	1	0	0	1	3
GMM [20]	0	0	0	0	0	0	1	1
KDE [20]	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Table 10 Algorithm instance selection for the *ForegroundDetection* entity after including the new algorithm instance KDE (considering the properties of the D3 domain and the U1 user preferences). Data correspond to the satisfaction scores for each instance (final selection is marked in bold).

Workflow approach	Accuracy		Cost (ms/frame)	Memory (MB)
	P	R		
Fixed	.63	1	65	215
Proposed without addition	.85	1	90	315
Proposed with addition	.92	1	93	360

Table 11 Results for the addition of new capabilities for foreground analysis (third experiment).

and its description in the System Ontology Database (SOD). This algorithm has the following properties: Type (any), View (any), Time (any), Crowded (no), Proc. Time (medium), Memory (high) and Accuracy (high). As there is no change of entities, the structure of the workflow is not altered. Then, the satisfaction scores of the new algorithm are computed in order to decide if it is more suitable than the algorithm employed currently. The obtained scores are listed in Table 10. As it can be observed, the algorithm is more appropriate for the current analysis. Hence, the previous algorithm (GMM) is removed and the new one (KDE) is used in the workflow. The results comparing the fixed workflow and the proposed one (with and without adding the new capability) are summarized in Table 11. They demonstrate that resulting workflow increased the accuracy of the final system (as preferred by the user). In particular, the number of false positives was decreased from 2 to 1.

Removal In this case, we have decided to remove the shadow removal capability (the entity and all the available instances) and recompute the workflow. This modification affects the structure of the workflow and, therefore, the whole process is repeated for constructing a new workflow at entity and instance levels. The resulting workflow after the removal is depicted in Fig. 11 and its results are presented in Table 12. As it can be observed, the removal of this stage decreased the precision of the resulting workflow.

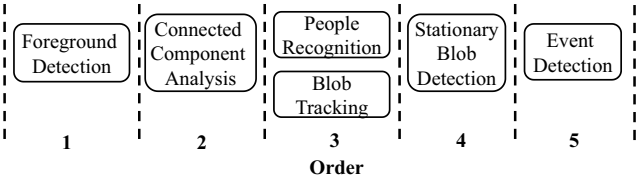

Fig. 11 Composed workflow at entity level for the abandoned object detection tasks after the removal of the shadow detection capability.

Workflow approach	Accuracy		Cost (ms/frame)	Memory (MB)
	P	R		
Fixed workflow	.63	1	65	215
Proposed without removal	.85	1	90	315
Proposed with removal	.75	1	78	290

Table 12 Results for the addition of new capabilities for foreground analysis (third experiment).

6.5 Computational cost comparative evaluation

A comparison has been done to study the additional computational cost introduced by proposed approach. The selected state-of-art approach built the fixed workflow in approximately 4500 ms whilst our approach took 10500 ms. An increase around 233% was observed due to the semantic-based workflow composition. Furthermore, this time depends on the amount of information encoded in the description of the domain and the framework capabilities. Thus, higher knowledge bases (i.e., more domain descriptions or system capabilities) will imply more delay for creating the workflow. However, this time could be considered as inappreciable for the analysis of long sequences or 24-hour operating systems. Regarding the processing of each frame, the proposed approach allows to set the focus of this criterion defining a maximum and a minimum processing time (that corresponds to the Proc. Time criterion to, respectively, the values *Low* and *High*). Table 13 shows the measured times.

Workflow approach	System creation	Frame processing	
		Max	Min
Base	4500	68	63
Proposed	10500	90	45
Difference	+233	+32.5	-40

Table 13 Computational time comparative results (ms).

7 Summary and conclusions

This paper has described a distributed framework for video analysis that allows flexible and dynamic configuration at run-time. It provides support for acquiring, transmitting, processing and storing data. Additionally, it defines a flexible environment to develop video-based applications via easy component integration.

Furthermore, we have presented how the formalization of knowledge relevant to video analysis (in terms of domain and capabilities) can be used to automatically compose and update the analysis workflow for a specific domain. This composition is performed by analyzing the relations between the entities defined for each application domain, the system capabilities (i.e., visual analysis tools available) and the user preferences. This process is divided in four stages: data request, selection of the algorithm entities to apply, determination of their execution order and selection of the appropriate instances. A rule-based approach is applied to extract the entities relevant to solve the analysis problem and compute their execution order. The selection process for specific algorithm implementations is modeled as a constraints satisfaction problem (CSP). Experimental results show that the proposed method operates at the same performance level as a similar hand-defined workflow adding a low delay for initialization.

The main advantage of this framework is the integration of ontology-based descriptions and video analysis tools. Any domain described by the ontology can be analyzed with the proposed framework. It adapts to analyze different domains addressing the properties of the domain to analyze and requirements of the user. Moreover, the design of such kind of frameworks is separated in two parts: domain-knowledge-related and algorithmic-related parts. Domain experts and algorithm designers can focus their efforts in the development of more accurate models or algorithms.

Moreover, the proposed approach is suitable for distributed settings due to its scalable nature. For example, a multi-camera network scenario would benefit from having nodes specifically designed to each task (capture, processing, storage, visualization) whilst they are running in parallel. Replication of network nodes allows to increase their capabilities (e.g., additional computational power by including more algorithm severs mapped to different nodes). However, efficient coordination strategies are required for optimal usage of resources [25]. In addition, the distribution capability of the proposed approach is affected by the amount of data transmitted as it is frame-based. A bottleneck might occur when deploying large networks. Similarly to the smart cameras approach [8], a solution might be to embed the capture, processing and communication tasks into one single device. Hence, only metadata and visualization data should be transmitted.

As future work, we will investigate on the automatic distribution of the selected algorithms between the available processing units in the framework as well as on the application of the proposed framework to other video analysis tasks.

Acknowledgements This work has been partially supported by the Spanish Government (TEC2011-25995), by the Consejería de Educación of the Comunidad de Madrid and by The European Social Fund.

References

1. Apt, K.R.: Principles of Constraint Programming. Cambridge University Press (2003)
2. Avanzi, A., Bremond, F., Tornieri, C., Thonnat, M.: Design and assessment of an intelligent activity monitoring platform. EURASIP Journal on Applied Signal Processing pp. 2359–2374 (2005)
3. Bai, L., Lao, S., Jones, G., Smeaton, A.: Video semantic content analysis based on ontology. In: Proc. of Int. Machine Vision and Image Processing Conf., pp. 117–124. Maynooth (Ireland) (2007)
4. Bayona, A., SanMiguel, J., Martinez, J.: Comparative evaluation of stationary foreground object detection algorithms based on background subtraction techniques. In: Proc. of IEEE Int. Conf. on Advanced Video and Signal based Surveillance, pp. 25–30. Genoa (Italy) (2009)
5. Bayona, A., SanMiguel, J., Martinez, J.: Stationary foreground detection using background subtraction and temporal difference in video surveillance. In: Proc. of IEEE Int. Conf. on Image Processing, pp. 4657–4660. Hong Kong (China) (2010)
6. Binotto, A., de Freitas, E., Pereira, C., Stork, A., Larsson, T.: Real-time task reconfiguration support applied to an uav-based surveillance system. In: Proc. of the Int. Multiconf. on Computer Science and Information Technology, pp. 581–588. Wisia (2008)
7. Bins, J., List, T., Fisher, R.B., Tweed, D.: An intelligent and task-independent controller for video sequence analysis. In: Proc. of the Int. Workshop on Computer Architecture for Machine Perception, pp. 172–177. Palermo (Italy) (2005)
8. Bramberger, M., Doblander, A., Maier, A., Rinner, B., Schwabach, H.: Distributed embedded smart cameras for surveillance applications. Computer **39**(2), 68–75 (2006)
9. Carincotte, C., Desurmont, X., Ravera, B., Bremond, F., Orwell, J., Velastin, S., Odobez, J., Corbucci, B., Palo, J., Cernocky, J.: Toward generic intelligent knowledge extraction from video and audio: the eu-funded caretaker project. In: Proc. of the Int. Conf. on Imaging for Crime Detection and Prevention, pp. 1–6. London (UK) (2006)

10. Cavallaro, A., Steiger, O., T. Ebrahimi, T.: Semantic video analysis for adaptive content delivery and automatic description. IEEE Trans. on Circuits and Systems for Video Technology **15**(10), 1200–1209 (2005)
11. Chappell, D.: Introducing Microsoft Windows Workflow Foundation: An early look. MSDN article (2005)
12. Dasiopoulou, S., Mezaris, V., Kompatsiaris, I., Papastathis, V.K., Strintzis, M.: Knowledge-assisted semantic video object detection. IEEE Trans. on Circuits and Systems for Video Technology **15**(10), 1210–1224 (2005)
13. Detmold, H., Dick, A., Falkner, K., Morrison, R.: Scalable surveillance software architecture. In: Proc. of IEEE Int. Conf. on Advanced Video and Signal based Surveillance, pp. 1–6. Sidney (Australia) (2006)
14. Fernandez, J., Martinez, J., Garcia, M.: Robust people detection by fusion of evidence from multiple methods. In: Proc. of IEEE Int. Workshop on Image Analysis for Multimedia Interactive Services, pp. 55–58. Klangenfurt(Austria) (2008)
15. Garcia, A., Martinez, J.: Robust real time moving people detection in surveillance scenarios. In: Proc. of IEEE Int. Conf. on Advanced Video and Signal based Surveillance, pp. 241–247. Boston (USA) (2010)
16. Greoris, B., Bremond, F., Thonnat, M.: Real-time control of video surveillance systems with program supervision techniques. Machine Vision and Applications **18**(3-4), 189–205 (2007)
17. GStreamer Organization: The online documentation of GStreamer,. http://gstreamer.net/documentation/
18. Haering, N., Venetianer, P., Lipton, A.: The evolution of video surveillance: an overview. Machine Vision and Applications **19**(5-6), 279–290 (2008)
19. Hall, D.: Automatic parameter regulation of perceptual systems. Image and Vision Computing **24**(8), 870–881 (2006)
20. Herrero, S., Bescos, J.: Background subtraction techniques: Systematic evaluation and comparative analysis. In: Proc. of Advanced Concepts for Intelligent Vision Systems, pp. 33–42 (2009)
21. Jaspers, E., Wijnhoven, R., Albers, R., Nesvadba, J., Lukkien, J., Sinitsyn, A., Desurmont, X., Pietarila, P., Palo, J., Truyen, R.: Candela - storage, analysis and retrieval of video content in distributed systems. In: Proc. of Int. Workshop on Adaptive Multimedia Retrieval, pp. 112–127. Glasgow (UK) (2005)
22. Konstantinides, K., Rasure, J.: The khoros software development environment for image and signal processing. IEEE Trans. on Image Processing **3**(3), 243–252 (1994)
23. Maggio, E., Cavallaro, A.: Video tracking: theory and practice. Wiley (2011)
24. Maillot, N., Thonnat, M., Boucher, A.: Towards ontology-based cognitive vision. Machine Vision and Applications **16**(1), 33–40 (2004)
25. Marcerano, L., Oberti, F., Foresti, G., Regazzoni, C.: Distributed architectures and logical-task decomposition in multimedia surveillance systems. Proc. of IEEE **89**, 1419–1440 (2001)
26. Mehmet, A., Choukair, Z.: Dynamic, adaptive and reconfigurable systems overview and prospective vision. In: Proc. of IEEE Int. Conf. on Distributed Computing Systems, pp. 84–89. Providence (USA) (2003)
27. Nadarajan, G.: Semantics and planning based workflow composition and execution for video processing. Ph.D. thesis, University of Edinburgh, UK (2010)
28. Prati, A., Mikic, I., Trivedi, M., Cucchiara, R.: Detecting moving shadows: Algorithms and evaluation. IEEE Trans. On Pattern Analysis and Machine Intelligence **25**(7), 918–923 (2003)
29. Ramesh, V.: Real-time vision at siemens corporate research. In: Proc. of IEEE Int. Conf. on Advanced Video and Signal based Surveillance, pp. 300–305. Como (Italy) (2005)
30. Saini, M., Kankanhalli, M., Jain, R.: A flexible surveillance system architecture. In: Proc. of IEEE Int. Conf. on Advanced Video and Signal based Surveillance, pp. 571–576. Genova (Italy) (2009)
31. San Miguel, J., Bescos, J., Martinez, J., Garcia, A.: Diva: A distributed video analysis framework applied to video-surveillance systems. In: Proc. of IEEE Int. Workshop on Image Analysis for Multimedia Interactive Services, pp. 207–210. Klangenfurt(Austria) (2008)
32. San Miguel, J., Martinez, J.: Robust unattended and stolen object detection by fusing simple algorithms. In: Proc. of IEEE Int. Conf. on Advanced Video and Signal based Surveillance, pp. 18–25. Santa Fe (USA) (2008)
33. San Miguel, J., Martinez, J.: Shadow detection in video surveillance by maximizing agreement between independent detectors. In: Proc. of the IEEE Int. Conf. on Image Processing, pp. 1141–1444. Cairo (Egypt) (2009)
34. San Miguel, J., Martinez, J., Garcia, A.: An ontology for event detection and its application in surveillance video. In: Proc. of the IEEE Int. Conf. on Advanced Video and Signal based Surveillance, pp. 220–225. Genova (Italy) (2009)
35. SanMiguel, J., Martinez, J.: Dynamic video surveillance systems guided by domain ontologies. In: Proc. of the Int. Conf. on Imaging for Crime Detection and Prevention, pp. 1–6. London (UK) (2009)
36. Siebel, N., Maybank, S.: The advisor visual surveillance system. In: Proc. of IEEE Eur. Conf. on Computer Vision, pp. 103–111. Prage (Czech Republic) (2004)
37. Szeliski, R.: Computer Vision: Algorithms and Applications. Springer-Verlag London Limited (2011)
38. Tian, Y.l., Brown, L., Hampapur, A., Lu, M., Senior, A., Shu, C.f.: Ibm smart surveillance system (s3): event based video surveillance system with an open and extensible framework. Machine Vision and Applications **19**, 315–327 (2008)
39. Town, C.: Ontological inference for image and video analysis. Machine Vision and Applications **17**(2), 94–115 (2006)
40. Varela, M., Velastin, S.: Real-time architecture for a large distributed surveillance system. In: Proc. of IEE Intelligent Distributed Systems, pp. 41–45. London (UK) (2004)
41. Venetianer, P.L., Zhang, Z., Yin, W., Lipton, A.J.: Stationary target detection using objectvideo surveillance system. In: Proc. of IEEE Int. Conf. on Advanced Video and Signal based Surveillance, pp. 242–247. London (UK) (2007)