Real-time Control of Video Surveillance Systems with Program Supervision Techniques

Benoit Georis, François Bremond, Monique Thonnat

Machine Vision and Applications manuscript No.
(will be inserted by the editor)

B. Georis · F. Brémond · M. Thonnat

Real-Time Control of Video Surveillance Systems with Program Supervision Techniques

the date of receipt and acceptance should be inserted later

Abstract In this article, we describe a knowledge-based controlled platform using program supervision techniques. This platform eases the creation and the configuration of video surveillance systems. Several issues need to be addressed to provide a correct system configuration: (1) to choose, among a library of programs, those which are best satisfying a given user request, (2) to assign a correct value for each program parameter, (3) to evaluate performances and to guarantee a performance rate which is satisfactory regarding end-user requirements. This platform is composed of three main components: the library of programs, the knowledge base and the control component. The knowledge is either given by experts or learnt by the system. The control is generic in the sense that it is independent of any application. To validate this platform, we have built and evaluated six video surveillance systems which are featured with three properties: adaptability, reliability and real-time processing.

Keywords Human Activity Recognition · Video Surveillance · Vision Systems

1 Introduction

Several libraries of video processing programs are available today. These techniques analyse the input video stream of one or several cameras and produce a more or less semantic description of the video content. The range of outputs can vary from a simple list of detected objects per frame up to a description of recognized events in a human-like language (e.g., "two persons are fighting"). Such a system acts as a filter to present only useful information to a human operator. Among these systems, some of them are able to work under real-time constraints and few simple industrial products exist today. Nevertheless, major issues are still encountered and video surveillance systems can still be considered at an early stage of development.

B. Georis · F. Brémond · M. Thonnat
INRIA/ORION team
2004 route des Lucioles
F-06902 Sophia-Antipolis
E-mail: firstname.lastname@sophia.inria.fr

The first important issue is that current techniques have poor performances over time. Video surveillance systems are facing a large diversity of environmental conditions such as sun, rain, fog, snow, and so on. As a consequence, the performances of a system can degrade significantly if it is not able to dynamically adapt itself in accordance with environment changes.

The second important issue arises when the same system must be installed on various sites. Most of the time, the system must be manually tuned or even worse, must undergo major changes. In both cases, the result is an extremely long and expensive process to obtain a reliable system. Another related issue is the development of new applications starting from existing ones, which is a hard task even for a video processing expert. As a matter of fact, existing programs and techniques are seldom used correctly for two main reasons. First, assumptions which have guided the design of a technique are often unclear and not formalized in a usable form by a system. Second, the control strategy (e.g., tuning of parameters) is often hidden in the code of the programs themselves. Thus, there is a need of efficient reuse and control of existing techniques.

The third important issue is the need of feeding the system with a priori knowledge. Actually, the semantic interpretation of video sequences is an intensive knowledge-based process, as the semantics is not only inside the video stream. For instance, in a bank application, without any knowledge of the security protocol, it is impossible to determine whether a rule has been infringed even if the system is able to correctly detect and track people. Moreover, the efficiency of video surveillance systems can be greatly improved by using contextual information (e.g., the geometry of the empty scene). Finally, by taking into account the high-level goal of the end-user, the system can focus the interpretation only on what is relevant to recognize the target activities. Current systems

usually do not make intensive use of all these types of knowledge.

To partly answer these issues, we propose a knowledge-based controlled platform to easily build, deploy and maintain video surveillance systems by combining existing techniques. More than error-proned, these systems should be sufficiently reliable to address end-user requirements. This article is organized as follows. Section 2 describes related works on control, section 3 describes the three main components of the knowledge-based controlled platform, together with a learning tool intended to complete the knowledge base. Finally section 4 describes the results obtained with the six video surveillance systems created with this platform.

2 Related Work

Historically, the control of video processing programs was tightly linked and not separated away from the programs themselves (i.e., hard-coded control) and the resulting systems were qualified as integrated systems [14], [21]. These early systems were designed specifically for a task and thus were not easily applicable to new domains and applications. Instead, they required a lot of changes and the developers had to start from scratch when creating a new system. In the meantime, several works in static imagery have presented techniques with an explicit model of control. We can mention the program supervision techniques which have been successfully applied to different applications, such as the recognition of complex objects, the classification of galaxies [22] or the supervision of medical imagery programs [7]. Program supervision aims at automating the use of a library of programs in order to help non-expert users to create their own applications. Indeed, sophisticated programs are conceived by specialists but are intended to be used by non-specialists who are often in charge of creating particular applications. However, these end-users usually only have a basic understanding of the available processing programs. Thanks to the program supervision, the resulting systems exhibit more flexibility and adaptability properties. More recently, together with the maturity of video processing program libraries [1], [27] and the availability of reasonable computing power resources, these works on control for static image analysis problems have inspired a new trend in the video understanding community with recent works which propose manual or automatic control architectures for video analysis.

2.1 Manual control

A first class of approaches proposes to ease the construction of video surveillance systems by providing modular software architectures to the system developers or video processing experts [9]. These architectures enable to manually control video processing programs using a control module to create a system for a particular application. The developers are able to manually select plug-in programs through the use of of well-defined interfaces. They are thus able to construct a system more easily. Moreover, the video processing experts can easily modify program parameters and directly see the impact on output results. The Imalab platform [16] aims at being user-friendly for the development of new algorithms and for interactive experimentations. Imalab is composed of hundreds of classes implementing data structures as well as processing functionalities. In addition, several tools to construct systems are provided, such as an interactive programming shell and an automatic program generator intended to use seamlessly external libraries. However, even if this convenient development environment is useful to build vision systems, it is not sufficient to achieve the design of efficient systems and to understand the combination of video processing programs. In addition, the formalism used to represent the control rules is still a programming language (e.g., Scheme). In conclusion, manual control architectures cannot automatically react to environment changes by tuning some parameters or selecting other programs.

2.2 Automatic control

Very recent works address this dynamic configuration issue by proposing automatic control architectures. In [12], the author proposes an automatic parameter regulation scheme for the tracking system described in [5]. The key idea is the following. An auto-critical function is able to detect a drop in system performances with respect to a model of scene dynamics. In such a situation, a regulation module is automatically triggered to provide a new parameter setting with better performances. The scene model is created by a k-mean clustering of the spatio-temporal measurements of a training data set of 21.000 bounding boxes annotated on video sequences coming from the same camera. The auto-critical function evaluates the probability that current measurements belong to the model. If this is not the case, the regulation module is activated. This module is in charge of searching the parameter space for a better set of parameters, using a genetic algorithm. Results show that the controlled system is not able to reach the performances obtained by a manual tuning of parameters. The controlled system does not use a priori knowledge of programs and their use whereas the expert does. Moreover, this technique is highly dependent on the training set and the scene dynamics space is huge. In consequence, obtaining an appropriate model can become a very hard task. Finally, the processing time is really too long for obtaining real-time systems.

The work presented in [4] and [15] aims at creating autonomous systems having automatic initial and dy-

namic configuration facilities. The novel idea is the introduction of a task-independent controller, which is written in Scheme on the Imalab environment [16]. This controller cooperates through a well-defined interface with a decision-making module which provides task-specific knowledge. Currently, a rule-based module is able to use both learnt and user-defined rules. At the initialization step, the controller creates the dataflow (either in a bottom-up or top-down fashion) based on the information provided by agents and modules. Agents are small code snippets which are in charge of analysing input data in order to extract their characteristics (e.g., pixel intensity level). Modules are video processing routines which have the ability to auto-describe and thus provide to the controller information such as parameters and their domains, inputs and outputs. After the initialization step, the control loop is made up of 4 steps: selection, execution, evaluation and repair. The repair operates by re-running modules with different parameter settings. The module description is a first step towards a declarative representation of video processing programs. However, this description is currently limited to parameters and input/output types. It does neither contain the module functionality nor abstraction levels (e.g., typical sequence of modules). Second, the users can express control rules but their representation (i.e., Scheme) is still a programming language. Third, this control architecture can potentially help to create autonomous and adaptable systems. However, the authors acknowledge the difficulty to evaluate module results (i.e., auto-evaluation can hardly be trusted) and the difficulty to obtain efficient control rules.

3 Knowledge-Based Controlled Platform

We propose a knowledge-based controlled video understanding platform for easy creation of dedicated video surveillance systems. Although this knowledge-based architecture is common for traditional AI systems, the novelty of this paper is to adapt this architecture to the video understanding domain. Constructing knowledge bases for video understanding is still an open issue due to the large variety of real-world situations to handle and due to the lack of formalized expertise on programs. This platform is composed of three components and is illustrated in figure 1. Two additional components (the evaluation and the learning tools) enable the expert to evaluate the created systems and to learn system parameters.

The first component is a library of video processing programs. The second component is a knowledge base containing three types of knowledge: knowledge of the application domain (i.e., what the system is expected to do, the end-user goal, such as counting people), the knowledge of the scene environment and the knowledge of video processing programs. Each type of knowledge is given by different experts (e.g., application domain expert, video processing expert) using a particular formalism. A learning tool enables the video processing experts to complete the a priori knowledge. The third component is a control component (i.e., a reasoning engine) which is intended to control the programs by using the information contained in the knowledge base. This control provides mechanisms based on rules for the systems to have reactivity/adaptability abilities for the conditions taken into account by the rules. For creating a system, these three components are assembled together (e.g., by a compilation operation). The evaluation component can then be used to evaluate the system performances and help experts to improve iteratively the knowledge base. This evaluation component is not the focus of this paper and a detailed description can be found in [10]. This design process can be iterated several times to capitalize the knowledge in the knowledge base (thanks to the proposed formalisms) and finally achieve acceptable performance. For instance, the experts may have to increment the knowledge base or to add new programs in the library.

3.1 Library of Programs

The library of programs is organized following a model of the video understanding process which consists of three main tasks: 1) object detection and classification, 2) spatio-temporal analysis and 3) event recognition. The programs composing our library have been extensively explained in several papers which are given in reference. Here, the objective is to describe how we have decomposed and structured the whole processing chain into programs or typical combinations of programs (and identified their input/output data) by following the proposed video understanding model. This description can be used as a guideline by video processing experts to define their own platform.

The first task is the detection and classification of physical objects of interest which are present in the scene. This task takes one image as input (colour or black and white) and produces a list of labelled physical objects of interest. To achieve this task, several distinct steps (including options and alternatives) have been identified:

- *Image acquisition*: this step produces a digitized image coming from a video source. Several alternatives exist: frame grabbing from an acquisition card, image loading from a file, decompression of a MJPEG live stream.
- *Reference image generation*: this step creates a reference image which is used during the segmentation step. There are three ways of computing this image [24]. For instance, the reference image is selected among a list of stored images corresponding to the empty scene taken under different illumination conditions. The selected image is the one which is the nearest of a mean image computed over the N first

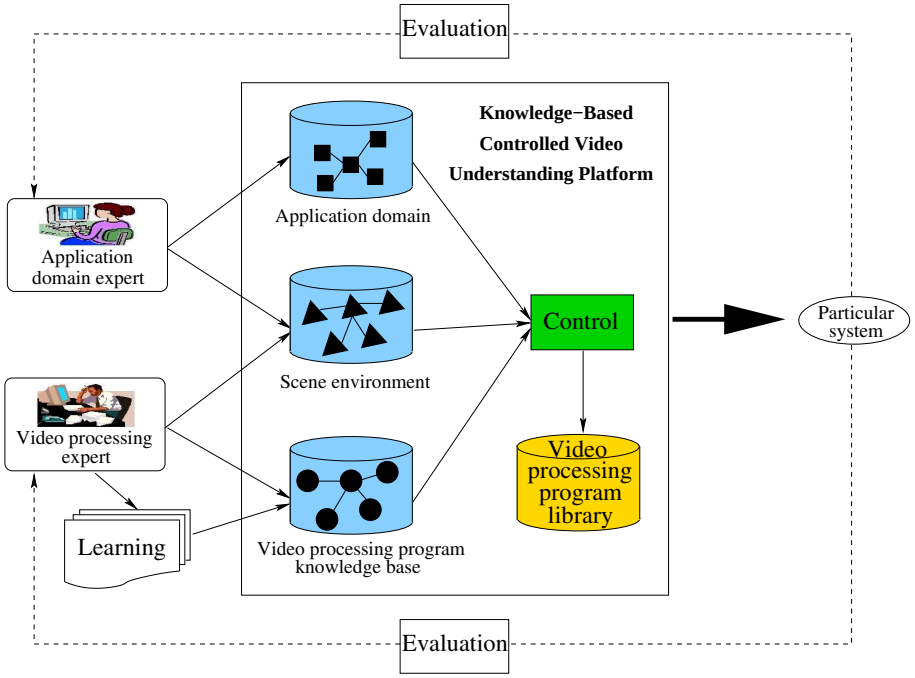


Fig. 1 A knowledge-based controlled video understanding platform used to build particular video surveillance systems. This platform is made of three main components: the knowledge base (blue), the control (green) and the library of programs (yellow).

frames of the video sequence, according to a metric based on statistical variables (e.g., standard deviation of intensity). One drawback of this technique is that it is a global approximation process.

- *Segmentation*: this step detects moving regions by subtracting the current image from the reference image. The resulting difference image is thresholded using several criteria based on pixel intensity in order to create moving regions. These moving regions (also called *blobs*) are associated with a set of 2D features like density or position. Two segmentation alternatives exist, depending on whether the chromatic information about pixels is available and usable or not.
- *Chair management*: this step is optional and works on the list of blobs [24]. It may be activated to help differentiating a blob corresponding to a moveable chair from a blob corresponding to a person, by using a priori information about contextual objects (e.g., by comparing a colour distribution of a blob with a predefined visual template of a chair).
- *Door detection*: this step is also optional and works on the list of blobs. It allows to handle the opening/closing of doors which have been specified in the 3D description of the scene (i.e., a priori knowledge). This algorithm removes from blobs the moving pixels corresponding to a door being opened or closed [3].
- *Classification*: this step takes the list of blobs and produces a list of physical objects of interest. It is composed of three successive substeps. First, a merge process tries to correct segmentation errors by regrouping small blobs corresponding to the same physical entity (e.g., person, car). Then, a split process is applied on large blobs to verify whether they could correspond to several physical entities. A set of 3D features like 3D position, width and height are computed for each blob, by using calibration information. By comparing this set of 2D and 3D features with predefined models, these blobs are classified into several predefined classes (e.g., person, group, car, truck, aircraft, unknown, noise,...). These blobs with their associated class label and their set of 3D features are called *physical objects of interest*.
- *3D position correction*: this step uses contextual knowledge to correct the 3D position of physical objects of interest which have been located at a wrong place (such as outside the boundary of the observed scene or behind a wall). This may happen when the bottom part of a person is not correctly detected (e.g., the legs can be occluded by a contextual object or badly segmented).
- *Ghost suppression*: this step aims at removing physical objects of interest which do not correspond to real objects in the scene (i.e., ghosts). For instance, some ghosts can be generated by stationary objects which have been integrated in the reference image (e.g., a car parked for hours) and which are now moving again. This technique analyses the presence or ab-

sence of gradient on the object contour in the current image.
- *Reference image updating*: this step works on the reference image and tries to integrate environment changes appearing in the current image. A simple and fast technique consists in blending slightly each reference pixel with a small coefficient (e.g., 0.01). A complementary program consists in discriminating real objects from regions of persistent change in the image (e.g., a new poster on the wall or a newspaper on the table) and to integrate only these regions [24]. This alternative is time consuming and requires to compute several features.

Once this first task is performed, the list of physical objects of interest is then processed by the **spatio-temporal analysis** task. This task is also composed of several possible steps:

- *Frame to frame tracking*: this step is performed for each video stream in case of a multi-camera configuration (e.g., in sequence or in parallel). The objective is to link from frame to frame all physical objects of interest [11]. The decision to create a link is based on several criteria such as the amount of overlap, the 3D distance between centers of gravity of objects. The output of this step is a graph containing the detected physical objects of interest updated over time and a set of links between objects detected at time t and objects detected at time $t-1$. A physical object of interest with temporal links towards previous ones is called a *tracked physical object of interest*. This graph provides all the possible trajectories of an object.
- *Fusion and Synchronization*: depending on the camera configuration, this step may be activated. Graphs of physical objects of interest coming from the different cameras with overlapped fields of view are fused together in order to obtain a unique representation. This technique uses combination matrices (combining several compatibility criteria) to establish the good association between the different views of a same object. A physical object of interest detected by a camera may be fused with one or more physical objects of interest seen by other cameras, or can be simply kept alone or destroyed if classified as noise. A temporal synchronization is sometimes necessary when the different cameras are not synchronized by a specific hardware. In such a situation, this step may decide to pause the processing on a camera to wait for the other ones. The output of this step is a graph of *fused tracked physical objects of interest*. They contain all the temporal links of the original objects which have been fused together and their 3D features are the weighted mean of the original 3D features. Weights are computed in function of the distances of original objects from the corresponding camera. In this way, the resulting 3D features are generally more accurate than original ones.
- *Long-term tracking*: this step works on the (fused) graph of objects. Depending on the events to recognize, several alternatives are activated [2], [8]. All of them rely on the same idea: they first compute a set of paths (in the graph) representing the possible trajectories of objects to track (e.g., isolated individuals, groups of people, crowd). Then they track the physical objects of interest with a predefined delay T to compare the evolution of the different paths. At each frame, the best path to update the physical object characteristics is chosen.

Once physical objects of interest are tracked, the **event recognition** task is performed. Depending on the type of events to recognize, different alternatives are used:

- For short events dealing with uncertainty, Bayesian networks can be used [17].
- For events with a large variety of visual invariants (e.g., *fighting*), AND/OR trees can be used [8]. Visual invariants are visual features which characterize a given event independently of the scene and of an algorithm. For instance, for a *fighting* event, some visual invariants are an erratic trajectory of a group of people, or one person lying down on the ground or important relative dynamics inside the group.
- Finally, for events involving multiple physical objects of interest and complex temporal relationships, the technique used is based on a constraint network whose nodes correspond to sub-events and whose edges correspond to temporal constraints [26]. Temporal constraints are propagated inside the network to avoid an exponential combination of the recognized sub-events. For each frame, events are recognized incrementally (i.e., temporal constraints are checked), starting from the simplest ones up to the more complex. This technique uses a declarative language to specify events to recognize. The output of this last task is a list of recognized events.

In order to supervise the library, we have defined a common interface for each elementary program. This interface is called a *module*. A module is a C++ static class which defines the necessary low-level communication functions between the programs and the control component. The control is thus able to invoke the execution of a video processing task and to assign values to parameters. In addition, we have extracted internal parameters away from the programs, which is a prerequisite for program reuse [19]. Finally, a module access function enables the control engine to retrieve a data when needed. In conclusion, this interface enables to seamlessly integrate and use programs coming from other video processing libraries. This is an important feature to extend the capabilities of the systems (e.g., to handle more complex user needs such as posture analysis) while avoiding modifying the control component.

3.2 Knowledge Base

There are three main categories of knowledge which are important to distinguish and to formalize. First, the knowledge of *the application domain* has to be taken into account to enable a goal-directed control and to ensure that systems will match end-user expectations. The process of video understanding is intended to match the perceived observations with predefined models of events in order to produce a semantic description of a scene. We are using a formalism for event modelling, initially proposed in [26], which represents video events by the following three parts:

- Physical objects: all real world objects present in the scene observed by cameras.
- Components: list of sub-events involved in the event. They are optional.
- Constraints: relations between physical objects and/or components.

Thanks to this formalism, we are able to model a large diversity of events for various applications and to capitalize this expertise in the platform with a knowledge base of event models. When constructing an event model library, we first select a set of primitive events. In a second time, we can build more complex events as illustrated in figure 2 which are a combination of already defined events.

composite_eventbank_attack
physical_objects:
((employee: Person), (robber: Person), (z1: Entrance),
(z2: Back_Counter), (z3: Infront_Counter), (z4: Safe))
components:
((c1: primitive_state inside_zone(employee, z2))
(c2: primitive_event changes_zone(robber, z1, z3))
(c3: primitive_event changes_zone(employee, z2, z4))
(c4: primitive_event changes_zone(robber, z3, z4)))
constraints: ((c2 during c1)
(c2 before c3)
(c1 before c3)
(c2 before c4)
(c4 during c3))

Fig. 2 Composite event with 2 persons using primitives inside_zone and changes_zone. The bank employee is behind the counter. The robber enters the bank agency, goes towards the counter and stays in front of it. Then, both people go to the safe.

Second, the knowledge of *the scene environment* has to be taken into account to obtain systems which can adapt to environment changes. Most people agree [25], [20] on the utility of this information to achieve more efficient systems. Nevertheless, few people use this information because it is difficult to acquire it and to link it to programs efficiently (i.e., in a declarative manner, not inside program code). For instance, in an indoor monitoring application, it is often useful to manage doors as they provide useful information to avoid a confusion between door motion and person motion. The transparency attribute of a door is important to know whether a moving region is due to a person moving in front of an opaque and closed door or moving behind the door but seen by cameras as the door is transparent. The state (i.e., closed, opened or in transition) of the door is also a valuable information to decide whether a moving region corresponds to a person or to a noise generated by the movement of the door. Finally, the colour and/or texture information can be used to better discriminate people passing in front of the door. We can see with this non-exhaustive description that the video processing programs can take into account a lot of different contextual information (i.e., the same diversity exists for chairs, escalators, and so on). Thanks to a formalism we can precisely describe the relations between a detailed list of attributes (e.g., fragile, transparent) and the video processing programs (e.g., if transparent then increase the parameter by 10). We have modelled several concepts using a frame formalism: the camera, the scene (e.g., 3D geometric model, illumination type), the expected physical objects (e.g., geometric model, type) and the video sequences (e.g, frame rate). These descriptions are important to enable a reactive (or data-driven) control as they can guide the processing for taking decisions (e.g., to apply a filtering stage if a video sequence is qualified as *noisy*). Frames are well adapted because they allow to describe internal properties (e.g., type, size), structural properties (e.g., subparts) and relation between objects. For instance, figure 3 describes the *Camera* frame, expressed in the YAKL language [18], which currently contains 14 attributes. All attribute values of this model are a priori unknown and this model will refer to a particular camera instance once a value will be assigned for each attribute.

Third, the knowledge of *the video processing programs* must be formalized to capitalize expertise over time and to obtain efficient systems. In this context, the use of a formalism guided by an ontology is a convenient way for experts to express their knowledge which can be directly used by a system. We have selected the formalism [23] which is dedicated to knowledge representation for program supervision. The formalism includes both frame representation and production rules. Production rules are very modular (e.g., to add, modify or remove from the knowledge base) and readable (e.g., *if-then* relationship). The main concept of this formalism is an *operator*. Operators represent programs or typical combinations of programs (with a description of their data and parameters) and a list of criteria to guide the reasoning process. They are used by the reasoning engine to produce a *plan* (i.e., a partial ordered list of operators to be executed). They are implemented by a com-

Object Type {
name Camera
comment "model of a camera"
Attributes

String	**name**	type
	range	[analog digital IP]
	default	IP
String	**name**	model
	default	AXIS2420
String	**name**	name
String	**name**	motion_characteristic
	range	[static moving PTZ]
	default	static
String	**name**	orientation
	range	[top_view side_view]
String	**name**	field_of_view
	range	[narrow normal large]
	default	normal
String	**name**	perspective
	range	[normal high]
String	**name**	scene_distance
	range	[close normal far]
Float	**name**	X3D_position
Float	**name**	Y3D_position
Float	**name**	Z3D_position
Integer	**name**	x_resolution
Integer	**name**	y_resolution
Float	**name**	frame_rate
	range	[1 25]

Fig. 3 The *Camera* frame describing the concept of camera. Currently, it contains 14 attributes. The YAKL syntax keywords are highlighted.

Primitive Operator { name a name
Functionality name of a functionality
Characteristics list of characteristics
Input Data
list of input arguments
Input Parameters
list of parameter arguments
Output data
list of output arguments
I−O relations
Preconditions
list of conditions on input data
Postconditions
list of conditions on output data
Initialization Criteria
list of initialization criteria
Assessment Criteria
list of assessment criteria
Adjustment Criteria
list of adjustment criteria
Repair Criteria
list of repair criteria
Call
language C++
syntax...endsyntax

Fig. 4 General syntax of a primitive operator in the video processing program knowledge base. The YAKL syntax keywords are highlighted.

bination of frames and production rules with the YAKL knowledge description language. There are two types of operators: **primitive operators** and **composite operators**. A **primitive operator** represents a particular video processing program (i.e., a *module*). The general syntax of a primitive operator is given in figure 4.

A **composite operator** represents a particular combination of programs (i.e., *modules*). It is a skeleton of plan provided by experts. Composite operators correspond to abstract functionalities and describe the network of known possible connections between sub-operators (choice, sequence, entailment, repetition,...) in order to achieve a given goal. These functionalities are linked to requests. A request is a query for an abstract functionality on particular data in a particular environment. In addition to the common information of primitive operators, a composite operator is composed of:

- Control information about the type of decomposition into sub-operators.
- References to the sub-operators by their names.
- Data flow information between father and sons and between sons in a sequential decomposition.
- Additional criteria: **choice criteria** and **optional criteria**.

There are several ways to model the knowledge. For instance, a program which has an adjustable parameter to select among two different techniques (e.g., a merge program which can be either specialized for persons or for vehicles) can be modelled either by two separate primitive operators or by a unique primitive operator which has an adjustable parameter to select the technique. We advise video processing experts to model the reality as closely as possible to guarantee the reusability. We thus prefer to define two primitive operators and a composite operator to choose between the two alternatives, as the adjustable program parameter effectively corresponds to a choice and not to a parameter which can be adjusted like a threshold.

Once video processing programs, combinations of programs and program parameters are represented by operators, various criteria can then be defined by experts by mean of production rules. These criteria represent the

expertise on how to use programs correctly for achieving a given request. There are four main types of criteria:

- Choice criteria: they are attached to composite operators. Choice criteria select the most pertinent operator(s) among the list of sub-operators according to the description of the end-user goal and the scene environment. For instance, the merging process can be chosen according to the type of the expected physical objects of interest in the scene (e.g., persons versus vehicles). In case the choice corresponds to decide whether or not to apply a sub-operator in a sequential decomposition type, this criterion is called an *optional* criterion.
- Initialization criteria: they contain information to assign a value to an operator parameter. For instance, a 3D distance threshold for a tracking operator can be set knowing the usual walking speed of a human being.
- Assessment criteria: they assess whether obtained results are satisfactory or not. For instance, an unsupervised evaluation may assign a *bad* assessment label to a blob merge process in case the output is a unique huge blob covering the whole image. In this situation, this bad evaluation is a diagnosis that a problem was encountered during the processing.
- Repair criteria: a repair stage can be triggered either with a bad performance evaluation or a detection of an environmental change. Two types of repair criterion can be distinguished: a *parameter adjustment* criterion or a *repair* criterion. An adjustment criterion enables to dynamically change a parameter value. A repair criterion typically creates a new selection of operators. For instance, in case the people density increases when people get out of a train in a subway, a global repair criterion can switch from an individual tracker operator to a crowd tracker operator.

An example of a choice criterion for a composite operator and an initialization criterion for a primitive operator is illustrated in figure 5.

The current hierarchy of 38 operators is represented in figure 6. To deepen the control and reasoning capabilities of systems, we should achieve a finer granularity which requires to identify and to model data and concepts which are manipulated by smaller operators.

```
Composite Operator { name ImageAcquisition
   ...
   Choice Criteria
   Rule {
      name acquisition_choice1
      Let camera a Camera
      If camera.type == "IP"
      Then use_operator IPAcquisition }
   ...
}
Primitive Operator { name PersonMerge
   ...
   Initialization Criteria
   Rule {
      name max_distance1
      Let camera a Camera
      If camera.perspective == "high"
      Then max_distance := 20          }
   ...
}
```

Fig. 5 A choice and initialization criteria belonging respectively to a composite and a primitive operator. The YAKL syntax keywords are highlighted.

3.3 Learning

Usually, knowledge bases are constructed manually by experts. In order to cope with missing or incomplete a priori knowledge, we propose to use offline learning techniques to learn values defined in rules (e.g., thresholds) relative to the video processing programs, in function of the scene environment. The rules contained in the knowledge base remain static whereas the values contained in the fact base are dynamic(e.g., rule thresholds can be adjusted during the processing). The proposed learning method proceeds in three steps. The first step is the identification of a set of parameters (related to a given video processing funtionality) which depend on a criterion characterizing the system environment. For instance, it can be a subset of parameters of a motion detector which are sensitive to the image contrast. The second step is the study of the variability of the selected criterion characterizing the environment (e.g., the image contrast). This study can be performed offline by a clustering algorithm (e.g., k-mean, PCA, ascending hierarchical classification) creating several classes according to a metric. For instance, the metric can be the euclidean distance between two histograms which are representing the image contrast. For each created class, we compute the set of optimal parameters using ground truth. Then, the system is online able to dynamically retrieve and use the optimal set of parameter values corresponding to the class which has the smallest distance with the current value of the criterion. For instance, the system computes online an intensity histogram on the current image and retrieve the class label learnt offline which has the closest histogram from the current one.

The main critical step is the selection of a metric. Most of the time, this metric does not take into account the variability of the real world and may create wrong element labels. There is room here to discover new techniques for quantifying abstract characteristics of environments of video surveillance systems. Thanks to this learning mechanism, we can obtain a self-diagnosis property for the system. The idea is to raise an online signal when the system has not been able to find an appro-

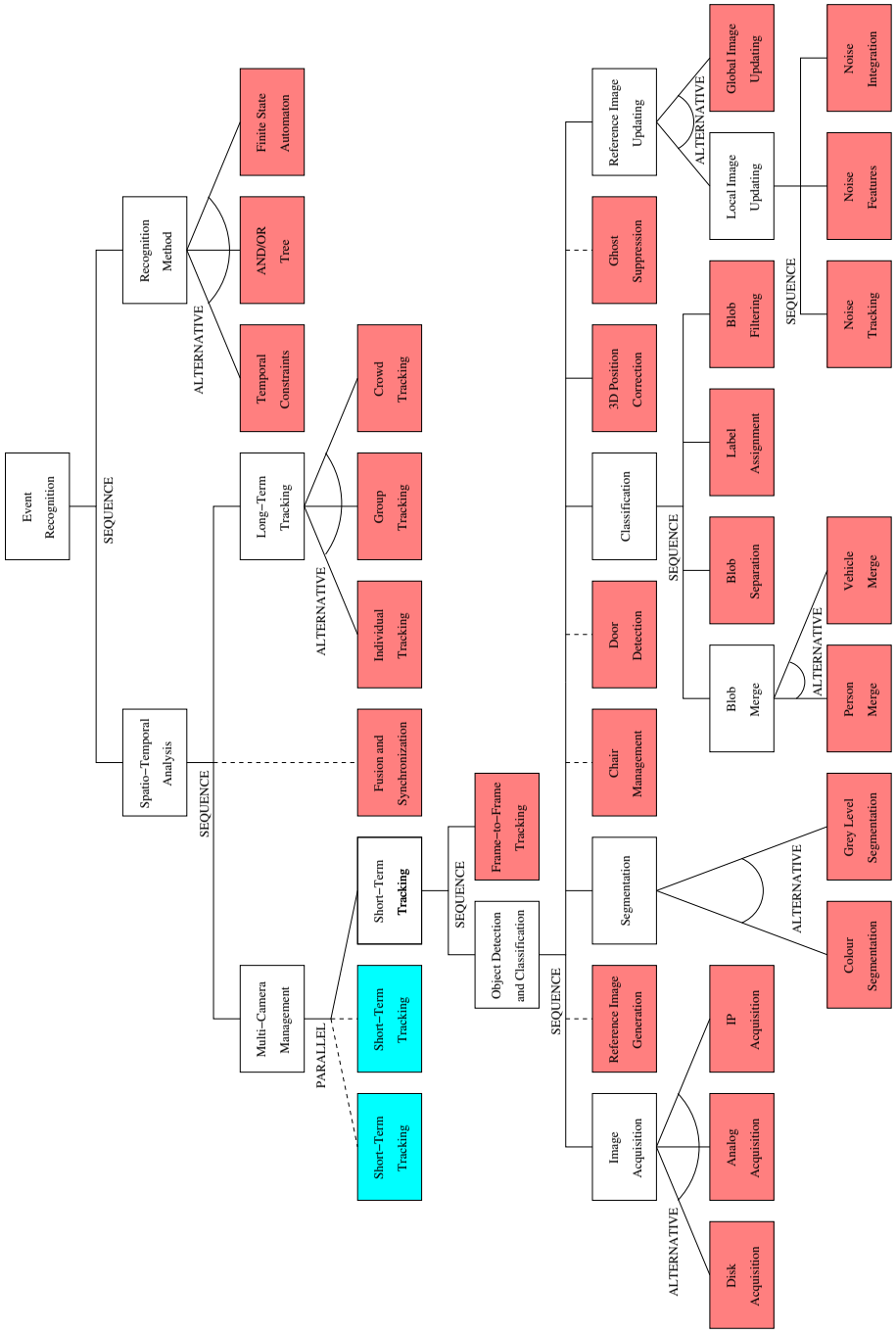


Fig. 6 The hierarchy of operators corresponding to the VSIP library of video processing programs. A plain (resp. dotted) line represents a mandatory (resp. optional) operator. Composite operators are represented by white squares while primitive operators are represented by pink squares. A green operator means that it is a duplicate of its corresponding brother (and of the full sub-hierarchy). Thus, it is not a primitive operator.

priate class label several times. This signal is a mean to alert the end-user that the system has encountered a new situation/configuration which requires to run offline a supervised learning on this new configuration. Thus, an evolution towards incremental learning can be investigated.

We have applied the general method described above to adapt the *ColourSegmentation* operator parameters to illumination changes arising from an outdoor camera. First, we have chosen the image contrast as a representative criterion influencing the segmentation results. The image contrast is here characterized by the distribution

of grey level intensities of an intensity histogram over the whole image. To study the variability of the image contrast, we have recorded several hours of video sequences of a same scene corresponding to different illumination configurations: different moments in a day, different days and different months. An intensity histogram is computed for each image taken every five minutes over the whole range of video sequences. Then, we have used a standard ascending hierarchical classification algorithm in order to cluster these histograms. We have chosen the euclidean distance as similarity measure between two histograms. After having identified several classes, we have to learn optimal parameter values for each class. In order to compare with segmentation output results, we have defined an accurate pixel-based ground truth corresponding to the real people evolving in the scene. To obtain optimized parameters, we have used the efficient Downhill Simplex method, even if we do not have a guarantee of obtaining a global minimum. Currently, we maximize S which is defined as the sum of correctly detected pixels (true positives) and correctly non-detected pixels (true negatives) over the total number of pixels (true positives, true negatives, false negatives and false positives). Finally, during the live system execution, the system computes the best match between the current intensity histogram and the histogram class representatives and then uses the optimal values corresponding to the selected class. This dynamic parameter adaptation is realized by the reasoning engine described in the previous sections.

3.4 Control Component

The role of the control component (i.e., reasoning engine) is to exploit all the knowledge (a priori or learnt) contained in the knowledge base in order to produce an efficient plan of programs. The reasoning engine starts with a user request, explores the different possibilities (in the hierarchy of operators) and computes the best one, with respect to the available information. The underlying reasoning mechanism is a Hierarchical Task Network planning technique interleaved with sophisticated control of execution (i.e., assessment and repair phases illustrated by steps 5, 6 and 7 in figure 7). The reasoning engine executes dynamically for each operator in the hierarchy presented in figure 6, four phases which are illustrated in figure 7:

1. Planning: this phase analyses the end-user request in order to select the best operator. Then, this phase creates a plan to solve the identified problem.
2. Program execution: this phase runs the video processing programs that have been ordered by the previous phase after having assigned values to parameters. This phase produces results.
3. Result evaluation: this phase detects potential problems concerning the returned results. This phase produces assessments on results. If results are good enough, the process can go on.
4. Repair step: if the assessment on results is negative, this phase decides which corrective actions are appropriate to undergo. This may lead to either reconsidering the arrangement of programs or changing some parameter values.

A main challenge to have an effective dynamic control of the system is to be able to provide the required knowledge expressing the decisions that the system must take online in order to execute tasks for improving its own performance.

This model of control enables to implement five control strategies:

- Goal-directed behaviour: the operator identification phase takes into account the end-user request or criteria. For instance, a choice criterion will decide to use the *GroupTracking* operator for recognizing a *fighting* event.
- Data-driven behaviour: the description of input data (e.g., video sequence, scene environment) in order to take decisions. For instance, the parameterization of primitive operators can depend on current data (e.g., colour versus black and white images).
- Closed-loop behaviour: this strategy corresponds to a feedback of information from high-level to low-level operators: the new processed result data becomes available information for the reasoning at the next time steps.
- Local or global repair behaviour: this strategy first needs the evaluation of results. Only a bad evaluation triggers the execution of repair criteria. In a local repair strategy, a repair criterion decides to re-execute the operator and triggers parameter adjustment criteria. In a global repair strategy, a repair criterion can either transmit the problem to a father operator in the hierarchy or return to the last choice.
- Real-time behaviour: using the timing characteristic of operators (e.g., *costly* operator) to enforce an execution in a specified time. Unfortunately, the current engine specifications do not allow a hard real-time strategy (e.g., with interruption handling), but rather a processing at video frame rate (i.e., verifying experimentaly that the processing time of algorithms is less than the frame rate).

The control flexibility necessary for obtaining adaptable systems comes from criteria which are defined by experts. Criteria are a mean for experts to obtain a combination of these different control strategies and to give more importance on a strategy depending on the target application. To deal with the temporal dimension into the reasoning mechanism, we have extended the initial formalism in three points. First, we have created an iteration mechanism which can loop on the root operator (corresponding to the user request) for all images of the input video stream. Second, we have limited the local

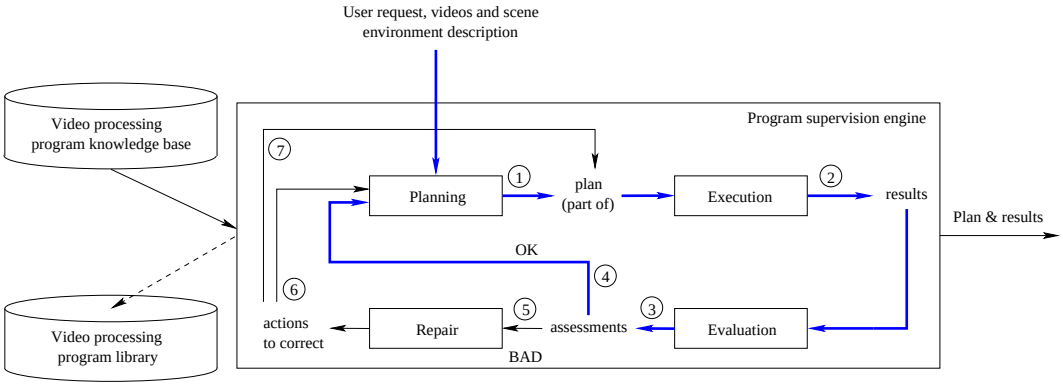


Fig. 7 Main phases of the reasoning engine: *planning* transforms the end-user request into a (partial) plan (1), *execution* calls the video processing programs and produces results (2) that are passed to *evaluation* which returns assessments (3). If the assessments are correct (4), the planning process can go on. If failures have been detected (5), *repair* decides which correcting actions are appropriate (e.g., replanning (6) or re-execution (7)) after tuning some parameters. Blue bold arrows show the main recursive loop. Plain black arrows show the repair loops.

repair mechanism to be only applied on operators which are able to restore their context of execution. These operators can process several times the same input data in the same conditions. Third, we have extended the global repair mechanism which is usually time-expensive to re-execute operators on the next frame rather than on the current frame. Thanks to these modifications, the control mechanism does not add any significant overload to the normal way of processing (at video frame rate).

3.5 Creation of a System

Given a library of video processing programs, a reasoning engine and a knowledge base, we are able to build a video surveillance system. As described in section 3.2, the knowledge base contains all information about the video processing program library, the description of the scene environment and the end-user goals. The main part of the knowledge base which is the knowledge of the video processing programs is generic and has been built in about six months. In counterpart, the knowledge which is more specific to a system (e.g., the scene environment and the end-user goals) can be defined in less than one day. In addition, after having performed an evaluation cycle, it is often necessary to add new generic control rules on video processing programs to take into account some new real-world situations.

The first step consists in parsing all the YAKL description files (i.e., knowledge base) and generating automatically the corresponding C++ code. Then, we compile and link the control, the knowledge base and the library of programs all together. The result of this operation is a binary file representing the current controlled system. This binary code is able to dynamically control the video processing algorithms depending on the input video streams given to the system. Finally, we have to specify a user request. Currently, we can ask for three different requests: *ShortTermTracking*, *SpatioTemporalAnalysis* and *EventRecognition*. Figure 8 illustrates the *EventRecognitionFunctionality* and the *event_recognition1* request which correspond to the detection of a fighting event in a Brussels metro station. In this case, the reasoning engine calls the appropriate recognition operator based on the type of event given in the request.

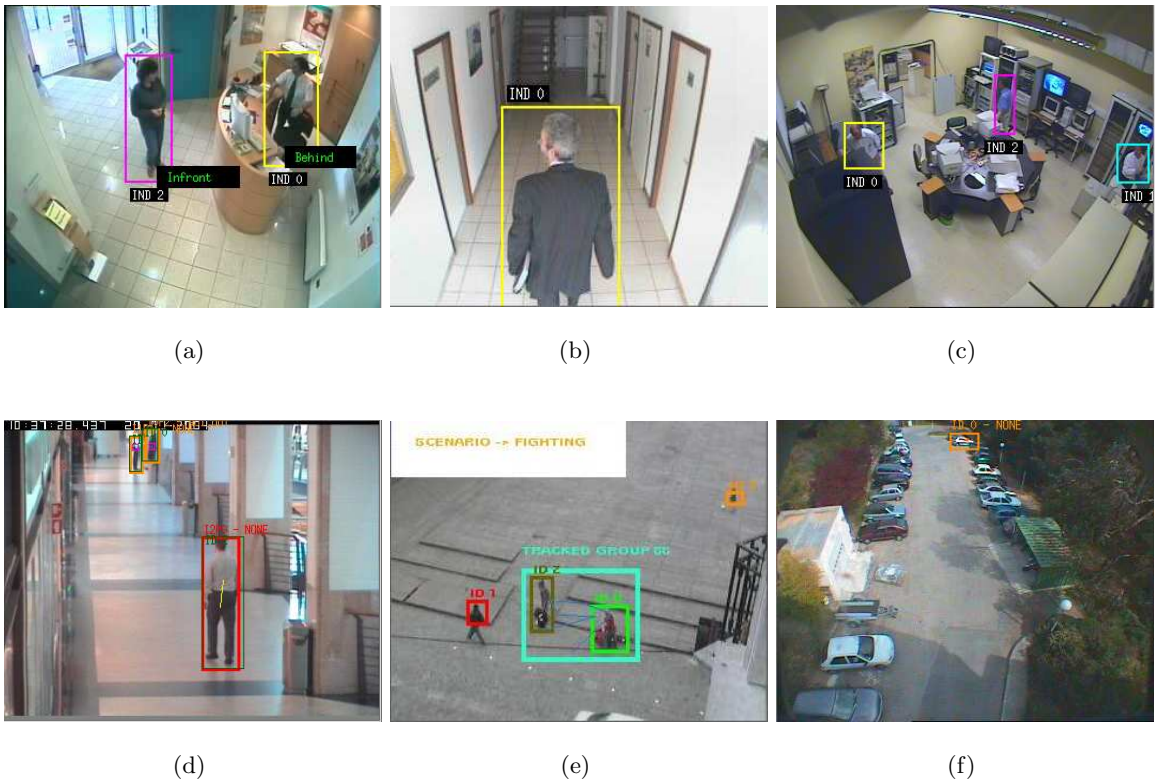
```
Functionality {
   name  EventRecognitionFunctionality
   Achieved by  EventRecognition

   Input Data
      Integer  name  camera_configuration
      Camera   name  camera1
      Camera   name  camera2
      Scene    name  scene
      String   name  event_type
}

Request {
   EventRecognitionFunctionality  name  er1

   Attributes
      camera_configuration  :=  2
      camera1 :=  brussels_metro_camera1
      camera2 :=  brussels_metro_camera2
      scene :=  brussels_metro_scene
      event_type :=  "violence"
}
```

Fig. 8 Instantiation of a request which defines a particular system. The YAKL syntax keywords are highlighted.



(a) (b) (c)

(d) (e) (f)

Fig. 9 This figure shows six systems created with the knowledge-based controlled platform for six applications: bank monitoring (9(a)), directional people counting in a building hall (9(b)), vandalism detection in an office (9(c)), shopping mall monitoring (9(d)), violence detection on a public place (9(e)) and car park monitoring (9(f)).

4 Results

In order to validate the approach, we have used the knowledge-based controlled platform to create and configure six video surveillance systems which are illustrated in figure 9. In consequence, three properties of the platform are demonstrated. First, each system has been created in a couple of days, showing the ability of the platform to easily configure systems.

Second, all the systems are running in real-time. Moreover, we show in table 1 that the overload due to the declarative control is limited to 4%. Concerning the regular way of working of the system, the mean time is a pertinent measure to verify that the processing is real-time. However, some repair of more costly repair operations may slown down the overall process. In consequence, controlled systems are likely to work at video frame rate, even if occasionally few frames (e.g., ten per day) are dropped due to the system overload.

Third, these systems exhibit reliable video processing performances. We report here the evaluation results for six systems. Results are presented either for the object detection task or the event recognition task with two indicators: the sensitivity and the precision. Precision is defined as the percentage of good results among all computed results (i.e., the number of true positives over the sum of true positives and false positives). Sensitivity is defined as the percentage of good results among all expected results (i.e., the number of true positives over the sum of true positives and false negatives). The evaluation has been performed by comparing the obtained results with the ground truth. Concerning the object detection task, the comparison has been performed using a pixel comparison based on bounding boxes.

For each application system, the evaluation results are presented for three versions of the system, when available. These versions differ on the way they have been built:

- Minimal controlled system (MCS): The first system has been created with the platform using a first version of the video processing program knowledge base containing 90 initialization criteria, 16 choice criteria, 9 optional criteria, 25 assessment criteria and 7 repair criteria.
- Augmented controlled system (ACS): The second system has been created with the platform after that the first knowledge base has been augmented with 14 additional initialization criteria (8 for the segmentation process, 4 for the merge process and 2 for the ref-

Table 1 Timing comparison between an execution of the system in the controlled mode and in the non-controlled mode. The duration of the video clip is 150 000 frames.

	Controlled system	Non-controlled system
Number of frames	150 000	150 000
Processing time (seconds)	15 600	15 000
Frames/second (mean)	9.6	10

Table 2 Improvement in the precision performances of the object detection task for the bank agency monitoring system, thanks to the definition of new control criteria.

	MCS	ACS
Sensitivity	75%	75%
Precision	70%	71%

erence image updating process). These criteria have been added after a first evaluation cycle.
– Hand-coded system (HCS): The third system has been created without the controlled platform (i.e., an integrated system) and has been manually tuned and modified by the application developer in order to obtain the best results on this particular application. For instance, the application developer was free to add new programs.

Bank agency monitoring system The objective of this application is to recognize a bank attack event involving two people: the robber and the bank employee. For this system we have performed a supervised evaluation of the object detection task on 2 sequences of 500 frames, with the MCS and the ACS system. Results are reported in table 2. We can see that there is a slight improvement in precision, thanks to the eight new initialization criteria which assign a higher value for the colour threshold used during segmentation. These modifications have been realized in two days which correspond to the time needed to perform all the evaluation procedures on the whole test video database.

Directional people counting system The objective of this second system is to count people and to identify their trajectories (i.e., the origin and the destination) in an entrance hall of a company. The goal is to be able to check the comings and goings of visitors in this company. We present here the evaluation results computed at the output of the whole processing chain. This system has been tested on 68139 images divided into 45 video clips containing simultaneous person passage and 102 video clips containing a single person passage. Comparative results are reported in table 3 for the MCS, ACS and the HCS. This table shows that results have significantly improved from MCS to ACS version. However, the ACS system is still less efficient than the HCS system which has been manually tuned and modified during one year by a video processing expert.

Table 3 Event detection performances for the directional counting application with the minimal supervised system (MCS).

	MCS	ACS	HCS
Sensitivity	83%	95%	100%
Precision	38%	46%	57%

Table 4 Improvement in the precision performances of the object detection task for the vandalism detection system, thanks to the definition of new control criteria.

	MCS	ACS
Sensitivity	66%	64%
Precision	89%	94%

In this case, two new criteria have been added to enhance the 3D localization of people (and so to ensure a correct classification) in a close-up view situation.

Vandalism detection system The third system is intended to detect a vandalism act against an equipment. This type of event involves 3 persons: a vandal, his/her accomplice and a third independent person. This event model describes the following temporal sequence of actions: the vandal comes near an equipment to damage it, then go away from it when his/her accomplice alert him/her that a third person arrives. An alert must be raised when the vandal comes close to the equipment more than twice. Two experiments have been done: one to validate the event recognition task and another one to validate the object detection task. For the first experiment, we have been able to recognize the unique vandalism event occurring in a video sequence of 620 frames, with no false alarm. For the second experiment, we report in table 4 evaluation results for the detection task on these 620 frames, showing the improvement of the ACS system mainly in terms of precision.

Shopping mall monitoring system For a purpose of benchmarking video processing performances with other video understanding systems, we have created a shopping mall monitoring system to be run on the CAVIAR [6] test sequences. We report here the evaluation results for the object detection task on the 26 sequences of the CAVIAR dataset which has a large part of artificial lighting. Again, we have performed the same kind of experiments by measuring the improvement between the MCS and ACS ver-

Table 5 Improvement in the performance of the object detection task for the shopping mall monitoring system, thanks to the definition of new control criteria.

	MCS	ACS
Sensitivity	83%	83%
Precision	70%	75%

Table 6 Comparison between the MCS, ACS and HCS violence detection system for the event recognition task.

	MCS	ACS	HCS
Sensitivity	11%	22%	56%
Precision	29%	44%	91%

sion of the system. We have been able to verify that the initialization criteria added for the bank agency monitoring system are also valid in this new environment. The results are reported in table 5.

The additional criteria take into account the high perspective of the camera with a side-view position which was generating problems in the detection of people. Considering the merge procedure, false positive pixels may appear in case two blobs are merged due to the perspective effect. In such a situation, a big blob contains both people but also a lot of wrong pixels in between. Unfortunately, it has been difficult to compare these evaluation results with other systems in the literature, due to the none availability of evaluation results. Some recent articles have produced results on other CAVIAR video sequences or globally on different applications, preventing us to perform an objective comparison. In addition, there is no agreement on the metrics. With no other objective in mind than giving an order of magnitude of current published results [13], the sensitivity of detection algorithms is less than 50%, using a different metric based on the number of boxes having a sufficient overlap with the ground truth. We believe that this metric is less accurate than the one we have used.

Violence detection system This system has been evaluated for the event recognition task with an outdoor camera viewing a square mixing pedestrians and vehicules on 22600 frames mixing actor played sequences and normal life situations. The current *AndOrTree* operator can recognize two types of violence actions: an erratic trajectory for a group of persons and a fallen person which is close to a group. The results are reported in table 6. We want to point out the complexity of this application. This is due to the fact there are a large variety of violence events that must be modelled. In addition, a single non-detected person may prevent to recognize the event, because the group of persons is not detected and created. The results presented here must be put in relation to the time that was spent to create these systems. On one hand, the MCS system has been created in one day and the first

Table 7 Performances for the object detection task for the car park monitoring ACS system, using the learnt parameter values. The first column has been obtained with a unique cluster decomposition, the second column has been obtained with a 2 cluster decomposition.

	MCS	ACS
Sensitivity	45%	74%
Precision	74%	74%

evaluation cycle needed to define new criteria and to obtain the ACS system has been realized in two days. In this situation, the new criteria are intended to increase the number of links created by the tracker to allow the creation of groups of people who are close to each other.

On the other hand, the HCS system has been designed during six months by the application developer. This developer has particularly focused on handling the vehicles passing by or in front the groups of persons. In fact, the vehicles generate wrong interactions with the groups, resulting in the wrong detection of erratic trajectories due to the switch of the tracked group identifier from one vehicle to another one.

Car park monitoring system The evaluation of the car park monitoring system aims at validating the values obtained by learning for the segmentation operator. The table 7 describes the performances of the object detection task on the testing video sequences in two cases. These two cases correspond to the two cluster decompositions we have applied on the range of video sequences recorded at different moment a same day: 1 cluster (the whole range) and 2 clusters (approximately half the range for each cluster). In the first case, all the testing sequences are processed with a unique set of parameter values corresponding to the unique cluster decomposition. In the second case, each testing sequence is processed with the set of parameter values corresponding to the closest cluster (i.e., this cluster is determined by the smallest distance between the cluster histogram representative and the histogram computed at the beginning of the test video sequence). These results have to be interpreted with care. Indeed, more experiments are needed to thoroughly validate the impact of the learnt values. However, these results demonstrate the expected trend: the overall performances increase when the system use learnt values from a cluster representing as closely as possible the environmental conditions. In other words, the more clusters we define, the higher the performances we expect to obtain.

All the created systems have shown better results using the additional criteria. They demonstrate the genericity of the criteria and the effectiveness of the evaluation cycle. For instance, we have visually found out that the 4 systems working in an indoor environment were generating a lot of false positives (i.e., small noisy blobs). This has been confirmed by the evaluation. The com-

mon explanation is that the artificial lighting condition makes the colour information less reliable, and thus not justified compared to the little improvement in sensitivity brought by this colour information. In consequence, it has been decided to assign a higher value on the 4 colour thresholds used during the segmentation process.

Finally, in order to demonstrate the effectiveness of the automatic control mechanism, we illustrate in figure 10 the dynamic system adaptation for a camera which is installed inside a safe in a bank agency. This room is usually dark unless a person comes into it and turns on the light. A usual background adaptation process is not sufficient to handle a fast transition. In addition, during a light switching, the sensor undergoes a transient period during which images are of very poor quality. In this case, the intelligent control enables to skip these frames in order to recover a normal processing after this transient period. The key idea is the assesment-repair action triggered by a strong illumination variation or a too large variation of the detected blob area. The repair action consists in generating a new background image from the previous image. After a transient period of about 15 frames, the system is able to detect the incoming person and then run with a correct background corresponding to the new environmental conditions. On the opposite, the non-controlled version of the system cannot recover from a wrong background.

```
Composite Operator { name Segmentation
   ...
   Assessment Criteria
   Rule {
      name eval_area_of_blobs
      Let repair a RepairHandler
      If {{Segmentation::getDataInteger("diffRelativeArea")}} >= 20
      Then repair.change_background := true ,
           assess_operator background_problem repair }
   ...
   Repair Criteria
   Rule {
      name repair_background
      If assess_operator? Segmentation background_problem
      Then send_up change_background }
    ...
}
Composite Operator { name ObjectDetectionAndClassification
   ...
   Repair Criteria
   Rule {
      name repair_background
      Let time a TimeHandler
      If assess_operator? ObjectDetectionAndClassification change_background
      Then time.current_time_step := time.current_time_step +1,
           re_execute
   ...
   Optional Criteria
   Rule {
      name optional_temporal_repair
      Let repair a RepairHandler
      If repair.change_background == true
      Then repair.change_background := false ,
           use_optional_operator ReferenceImageGeneration }
   ...
}
```

Fig. 10 Global (temporal) repair mechanism to generate a new reference image. The detected problem is transmitted up from the *Segmentation* to the *ObjectDetectionAndClassification* operator, which triggers a re-execution of itself on a new frame. This re-execution triggers the optional operator for creating a new reference image. The YAKL syntax keywords are highlighted.

Therefore, these six systems are reliable enough and were built in convenient conditions. These main properties are directly related to the explicit use of knowledge in the platform (e.g., knowledge of control of video processing programs) and its formalization.

5 Conclusion and Future Works

In this article, we have presented a knowledge-based controlled video understanding platform for easily building of reliable video surveillance systems. This platform uses a combination of tools (e.g., formalism, learning tool) and of dedicated ontologies (e.g., video processing data and functionalities). This platform is built on a program supervision architecture and contains three main components: a library of video processing programs, a knowledge base and a reasoning engine. In order to validate the proposed approach, we have created six video surveillance systems: bank agency monitoring, directional people counting, vandalism detection, shopping mall monitoring, violence detection and car park monitoring.

These systems have been built in a short amount of time (i.e., a couple of days per system). They have abilities to dynamically adapt themselves to the environment and they work at video frame rate. They are reliable enough to reach the end-user requirements, even if they are not as efficient as hand-coded systems. The reason is that there is a lot of knowledge to acquire. This knowledge acquisition needs many experiments and is a long process, especially while getting a deep and precise insights of video processing programs. Nevertheless, these systems show the effectiveness of the approach.

Finally, we propose future research directions which should be investigated in order to improve the proposed platform and the associated tools. Currently, the learning tool is able to learn the relationship existing between a set of parameters and a characteristic of the scene environment. However, this tool should be extended and generalized. For instance, we would like to learn typical control strategies or typical sequences of operators in a given situation. This way, the reasoning engine would be able to learn from experiences and would be able to take more efficient decisions when encountering the same or similar situations.

References

1. Aggarwal, J., Cai, Q.: Human motion analysis: a review. In: Proceedings of the IEEE Workshop on Motion of Non-

Rigid and Articulated Objects, pp. 90–102. Puerto Rico, USA (1997)
2. Avanzi, A., Brémond, F., Thonnat, M.: Tracking multiple individuals for video communication. In: Proceedings of the International Conference on Image Processing (ICIP'01), pp. 379–382. Tessaloniki, Greece (2001)
3. Avanzi, A., Brémond, F., Tornieri, C., Thonnat, M.: Design and assessment of an intelligent activity monitoring platform. EURASIP Journal on Applied Signal Processing, special issue on Advances in Intelligent Vision Systems: Methods and Applications **2005**(14), 2359–2374 (2005)
4. Bins, J., List, T., Fisher, R., Tweed, D.: An intelligent and task-independent controller for video sequence analysis. In: Proceedings of the IEEE International Workshop on Computer Architecture for Machine Perception (CAMP05), pp. 172–177. Palermo, Italy (2005)
5. Caporossi, A., Crowley, J., Hall, D., Reignier, P.: Robust visual tracking from dynamic control of processing. In: J. Ferryman (ed.) Proceedings of the 6th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS'04), pp. 23–31. Prague (2004)
6. Caviar: (2004). //homepages.inf.ed.ac.uk/rbf/CAVIAR/
7. Crubézy, M.: Pilotage de programmes pour le traitement d'images mdicales. Thse d'informatique de l'cole doctorale sciences de l'ingnieur, Universit de Nice Sophia-Antipolis, France (1999)
8. Cupillard, F., Brémond, F., Thonnat, M.: Behaviour recognition for individuals, groups of people and crowd. In: Proceedings of the 1st Workshop on Intelligent Distributed Surveillance Systems (IDSS'03). London, United Kingdom (2003)
9. Desurmont, X., Chaudy, C., Bastide, A., Delaigle, J.F., Macq, B.: A seamless modular image analysis architecture for surveillance systems. In: Proceedings of the 2th Workshop on Intelligent Distributed Surveillance Systems (IDSS'04), pp. 66–70. IEE London, London, United Kingdom (2004)
10. Georis, B.: Program supervision techniques for easy configuration of video understanding systems. Electrical engineering department, Universit Catholique de Louvain, Belgium (2005)
11. Georis, B., Brémond, F., Thonnat, M., Macq, B.: Use of an evaluation and diagnosis method to improve tracking performances. In: M. Hamza (ed.) Proceedings of the 3rd IASTED International Conference on Visualization, Imaging and Image Processing (VIIP'03), pp. 827–832. Acta press, Benalmadera, Spain (2003)
12. Hall, D.: Automatic parameter regulation for a tracking system with an auto-critical function. In: Proceedings of the IEEE International Workshop on Computer Architecture for Machine Perception (CAMP05), pp. 39–45. Palermo, Italy (2005)
13. Hall, D., Nascimento, J., Ribeiro, P., Andrade, E., Moreno, P., Pesnel, S., List, T., Emonet, R., Fisher, R., Victor, J.S., Crowley, J.: Comparison of target detection algorithms using adaptive background models. In: The Second Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS 2005). Beijing, China (2005)
14. Khoudour, L., Hindmarsh, J., Aubert, D., Velastin, S., Heath, C.: Enhancing security management in public transport using automatic incident detection. In: L. Sucharov, C. Brebbia (eds.) Urban Transport VII: Proceedings of the 7th International Conference on Urban Transport and the Environment for the 21st Century, pp. 619–628. WIT Press, Southampton, United Kingdom (2001)
15. List, T., Bins, J., Fisher, R., Tweed, D.: A plug-and-play architecture for cognitive video stream analysis. In: Proceedings of the IEEE International Workshop on Computer Architecture for Machine Perception (CAMP05), pp. 67–72. Palermo, Italy (2005)
16. Lux, A.: The imalab method for vision systems. Machine Vision and Applications **16**(1), 21–26 (2004)
17. Moenne-Locoz, N., Brémond, F., Thonnat, M.: Recurrent bayesian network for the recognition of human behaviors from video. In: J. Crowley, J. Piater, M. Vincze, L. Paletta (eds.) Proceedings of the 3rd International Conference on Computer Vision Systems (ICVS'03), Lecture Notes in Computer Science, pp. 68–77. Springer-Verlag, Graz, Austria (2003)
18. Moisan, S.: Program supervision: YAKL and PEGASE+ reference and user manual. Research Report RR-5066, ORION team, INRIA Sophia-Antipolis, France (2003)
19. Moisan, S., Thonnat, M.: What can program supervision do for program reuse. IEE Proceedings - Software **147**(5), 179–185 (2000)
20. Paletta, L., Greindl, C.: Context-based object detection from video. In: J. Crowley, J. Piater, M. Vincze, L. Paletta (eds.) Proceedings of the 3rd International Conference on Computer Vision Systems (ICVS'03), Lecture Notes in Computer Science, pp. 502–512. Springer-Verlag, Graz, Austria (2003)
21. Piater, J., Richetto, S., Crowley, J.: Event-based activity analysis in live video using a generic object tracker. In: J. Ferryman (ed.) Proceedings of the 3nd IEEE Workshop on Performance Evaluation of Tracking and Surveillance (PETS'02). Copenhagen, Denmark (2002)
22. Thonnat, M.: Knowledge-based techniques for image processing and for image understanding. J. Phys. IV France EDP Science, Les Ulis **12**(1), 189–236 (2002)
23. Thonnat, M., Moisan, S., Crubézy, M.: Experience in integrating image processing programs. In: H. Christensen (ed.) Proceedings of the 1st International Conference on Vision Systems, Lecture Notes in Computer Science, pp. 200–215. Springer-Verlag, Las Palmas, Gran Canaria, Spain (1998)
24. Tornieri, C., Brémond, F., Thonnat, M.: Updating of the reference image for visual surveillance systems. In: Proceedings of the 1st Workshop on Intelligent Distributed Surveillance Systems (IDSS'03). London, United Kingdom (2003)
25. Turner, R.: Context-mediated behavior for intelligent agents. International Journal of Human-Computer Studies Special Issue on Context **48**(3), 307–330 (1998)
26. Vu, T., Brémond, F., Thonnat, M.: Automatic video interpretation: a recognition algorithm for temporal scenarios based on pre-compiled scenario models. In: J. Crowley, J. Piater, M. Vincze, L. Paletta (eds.) Proceedings of the 3rd International Conference on Computer Vision Systems (ICVS'03), Lecture Notes in Computer Science, pp. 523–533. Springer-Verlag, Graz, Austria (2003)
27. Wang, L., Hu, W., Tan, T.: Recent developments in human motion analysis. Pattern Recognition **36**(3), 585–601 (2003)