Expert Systems with Applications 38 (2011) 11182–11192

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Intelligent surveillance system with integration of heterogeneous information for intrusion detection

J.L. Castro, M. Delgado, J. Medina, M.D. Ruiz-Lozano *

University of Granada, School of Computer Science, Department of Computer Science and Artificial Intelligent, C/Periodista Daniel Saucedo Aranda s/n E-18071, Granada, Spain

ARTICLE INFO

Keywords:
Intelligent-surveillance
Alarms notification
Multi-sensor system
Handheld devices
Context-sensitive notification

ABSTRACT

Recently, interest about security in public and private spaces has increased in favour of social welfare. Surveillance systems are increasingly needed to provide security for citizens and infrastructures. Currently there are many buildings that are equipped with cameras, sensors or microphones. However, it is difficult to find tools that integrate the information from these sources in a homogeneous system. On the other hand, the intruder detection is increasingly demanded in the corporate, commercial or private sector. For these reasons, we propose a multi-sensor intelligent system that uses information from several sources analysis (video, audio and other sensors) to identify dangerous or interest intrusions. So, we have designed a generic ontology that allows to integrate in a homogeneous way all the input heterogeneous knowledge. To perform the intrusion analysis, we propose a rule-based model, which process all the information obtained from the monitored environment. This model is easily customizable and adjustable, since the rules that define an intrusion in a semantic way can be configured depending on the scenario and circumstances. The system generates an alarm whenever an intrusion is detected. Besides, this alarm is also notified via mobile devices. So, the system reports in real time according to device capabilities, generating a context-sensitive notification.

1. Introduction

Currently, interest in maintaining the security of both people and infrastructure is considerably increasing. New technologies help to complement the monitoring process creating more powerful systems that detects dangerous situations. For this reason, intelligent surveillance systems have a crucial role for security. In this context, the video-surveillance is the area where more work has been done in recent years.

The first video-monitoring systems were the close-circuit television or CCTV systems, which are composed of a network of cameras that send signals to a central point, where the security staff can analyze the events that take place on the environment through visual displays. However, there exist research studies that assert that the performance of a security guard is considerably reduced after 20 min of attention. In this way, the traditional video-surveillance has a number of problems to meet the society demands for security. In other words, these systems should evolve to more intelligent and proactive tools.

Thus, with the aim of reducing the workload of human operators and improving the accuracy of the systems, the analysis of video content (or intelligent video) appears. In the last years, a lot of models and systems about Intelligent Video-Surveillance have been published (Valera & Velastin, 2005). We can find a high number of systems in the literature that use image processing algorithms and Artificial Intelligence techniques, such as VSAM (Collins et al., 2000), W^4 (Haritaoglu, Harwood, & Davis, 2000), and the systems (Bauckhage, Hanheide, Wrede, & Sagerer, 2004; Hudelot & Thonnat, 2003). The most prominent application areas for the development this type of systems are traffic monitoring (Bo et al., 2006; Collins et al., 2000; Fernández-Caballero, Gómez, & López-López, 2008), protection of public transport (Porikli, Ivanov, & Haga, 2008), video-surveillance in indoor environments (Marchesotti, Messina, Marcenaro, & Regazzoni, 2003; Prati, Cucchiara, & Vezzani, 2007). However, it is less common find researches that are focused in the concrete task of intrusion detection. One of these works is Patricio, Carbó, Pérez, García, and Molina (2007), where a multi-agent framework for visual sensor networks is described with the propose of carring out intrusion detection and tracking. Also, in Yuan, Sun, Varol, and Bebis (2003) an intelligent video-based visual monitoring system by detecting certain types of intrusion in dynamic scenes is presented. This system involves object detection and recognition (pedestrians and vehicles) and tracking. The common processing tasks that commercial systems perform are intrusion and motion detection[1] and packages detection[2].

* Corresponding author.
E-mail address: mdruilo@decsai.ugr.es (M.D. Ruiz-Lozano).

[1] http://www.cieffe.com, http://www.neurodynamics.com.
[2] http://www.objectvideo.com, http://www.ipsotek.com, http://www.neurodynamics.com

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.02.165

These systems have constituted the second and the third generation of surveillance systems. Second generation systems combine CCTV technology and IP surveillance with Computer Vision algorithms and Artificial Intelligence, while third generation systems are characterized by their inherently distributed nature and their use on a multisensor environment to provide a good scene understanding and attract the attention of the human operator in real time. A visual surveillance system can follow these stages: model and knowledge acquisition of the monitored environment (Mittal & Nikos, 2004), detection and tracking of moving objects (Behrad, Shahrokni, & Ahmad, 2001), object classification (Fusier et al., 2007) and behavior analysis (Borg et al., 2005).

Systems that analyze the video content by means of Artificial Intelligence techniques offer automatic detection in real-time and identification and analysis of potential risks. However, most of video-surveillance systems are based on image processing algorithms for making the detection and tracking of objects with the goal of performing simple and concrete functions (giving less emphasis on task of behavior analysis). If this analysis is also complemented with the information obtained from other sensors, then the utility of the whole system is considerably increased. Nowadays, studies that analyze audio (Atrey, Maddage, & Kankanhalli, 2006; Clavel, Ehrette, & Richard, 2005) are being developed in order to detect crucial sounds in a security environment. Even so, data obtained after an audio analysis are contextually different to data obtained from a video analysis. Integrating of a wide range of different information involves a very complex process. For this reason, it is very difficult to find video-surveillance systems that perform an analysis of audio with the aim of complementing the visual information.

In this context, our goal is fill this gap. For this, we will integrate information about audio, video or other sensors to analyse complex risk situations. Specifically, we have two main purposes. On the one hand, we want to provide security system with intelligence by fusing all the heterogeneous information obtained from the monitored multi-sensor environment and by performing the analysis of various events that take place in a concrete area. These events are recorded by cameras, sensors or microphones. And, on the other hand, we want focus in the analysis stage of object behavior to detect an intrusion in a building or concrete zone. The number of works in this area is scarce and this phase (behavior analysis) is one of the most useful because the results obtained can be showed to the security guard to help him in his work. So, our proposal will focus in this context to work with high-level information, and to detect more abstract situations, such as the intrusion in a building.

Thus, in this paper we present a multi-sensor system, which uses information from video analysis, sounds detection and sensors activation to detect intrusions in buildings or concrete zones. In the Section 2 is described the proposed system and its architecture. We have created an Ontology, which is defined in the Section 3. This Ontology allows integrate, in only one representation, the input heterogeneous information that is known in the scene.

In order to carry out the intrusion detection, we have developed a rule-based system that is described in the Section 4. This system allows us to know when an intrusion occurs, since an alarm is generated. This information is completed with new knowledge obtained by a Plugin that allows to relate events to the system objects. This plugin is described in the Section 5. In the Section 6, we present the alarm notifier in real time and context sensitive. Next, the system experimental results are showed in the Section 7. Finally in the Section 8, we present the conclusions and future work.

2. Multi-sensor intelligent surveillance system

The proposed system in this paper consists in a expansion of the system presented by the same authors in Castro, Delgado, Medina, and Ruiz-Lozano (2011). In Castro et al. (2011), a system that detects object collisions from video analysis is described. In this work, the system has been expanded to: (1) gather information from other monitoring sources (microphones and other sensors), (2) detect a new alert, 'the intrusion detection', and (3) notify alarms to mobile devices in real time in a context sensitive way.

Our system has been designed to the surveillance of local scenarios. We define a *scenario* as a monitored environment where occurs several events, so that they can be captured by cameras, microphones and other sensors. Some examples of scenarios are: the entrance to the garage, the hall of a building, etc.

The system will receive results of analysis of video, audio and sensors. And its function will be integrate and analyze all the input information in order to detect the presence of intrusions into the secure environment.

We want to stress that our work is focused on behavior analysis of the observed objects in an monitoring environment. We will use information that has already been processed by Knowledge Extraction Systems from cameras, microphones and other sensors. In other words, we will work with low-level information obtained by video, audio or sensors analysis to generate high-level new knowledge. Therefore, we do not perform detection of moving objects from video, or the classification of sounds from audio. There are many studies that perform these tasks and we will build on its results to carry out our own analysis.

Fig. 1 shows the inputs and outputs of the System and its architecture.

2.1. System inputs

As we have mentioned before, the inputs of our system are the outcomes of several analysis that have been carried by different basic Knowledge Extraction Systems. Specifically, the inputs are:

1. *Information from congnitive video analysis*. Our System receives two incoming streams of video information: on the one hand, information about object detection and 2D-tracking, and on the other hand, the identification of paths that these objects follow.
2. *Information from congnitive audio analysis*. The third stream that our system get as input is the classification of sounds, for example, the identification of a shot, broken glass, etc.
3. *Information from sensors*. The above input information is supplemented by information collected from other sensors existing in the secure environment, such as motion sensors, which are activated when an object is closed. So, our System receives information about the activation and deactivation of sensors.

2.2. System architecture

The architecture of the system is consisted of several *'Translators'* and one '*Processing Unit*'.

2.2.1. Translators

As mentioned above, our system receives as input the outcomes of different analysis carried out by Knowledge Extraction Systems from cameras, microphones or other sensors. For example, the object detection and tracking from video analysis, or a sound detection, or information about the activation of other sensors. As we can see, the input data are heterogeneous and they need

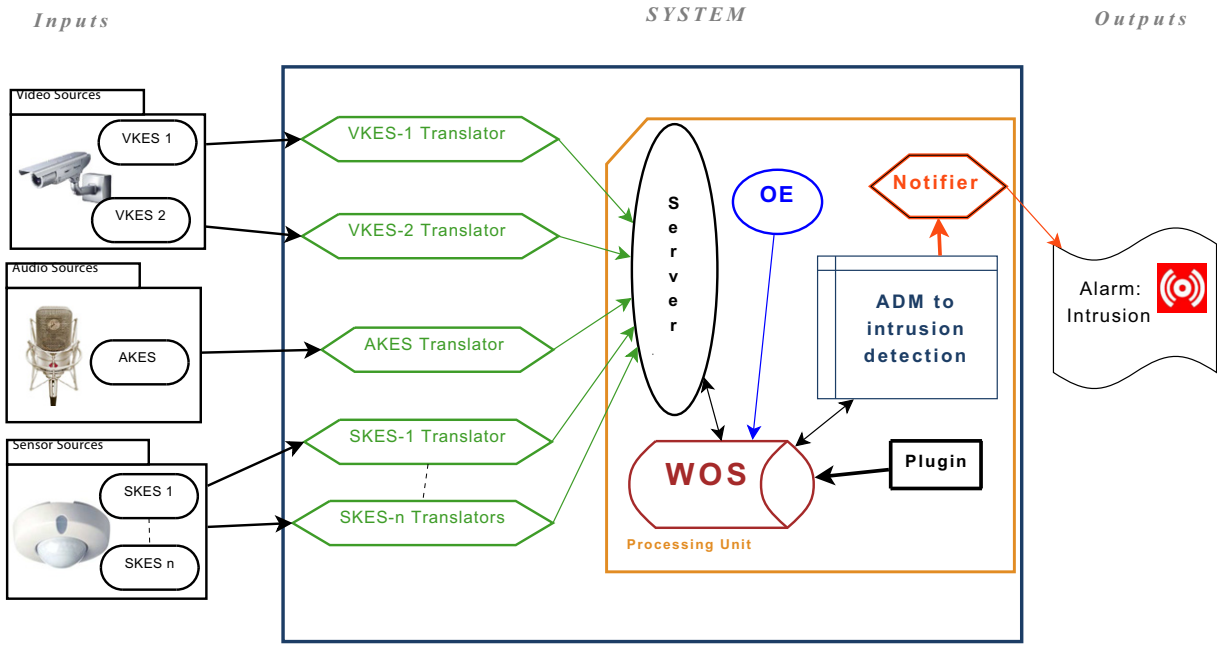

Fig. 1. Architecture of the system.

to be converted into homogeneous information under a domain that our system is able to handle. In order to resolve this fact, it creates the Translators, which are responsible for converting input events into data that are represented under the conceptual framework defined in our System Ontology (see Section 3). The Translators are constituted by a *Listener* that is constantly receiving the input events. There is one Translator for each Input Knowledge Extraction System. In this case, the System has four translators (one for each input stream):

1. Translator for input events about 'detection and 2D-tracking from Video'. This translator is equipped with a geometric procedure and camera calibration process to obtain the real position of the objects (3D positioning). This procedure is described in detail in a previous work presented in Castro et al. (2011).
2. Translator for input events about 'identification of object trajectories'.
3. Translator for input events about 'identification of sounds'.
4. Translator for input events about 'the activation and deactivation of sensors'.

In addition to integrating data in the Ontology, the Translators can contribute new knowledge or highest level data. For example, the 2D-tracking Translator performs a data processing to obtain the 3D location of objects. In this way, if it is possible, it will provide the Translators with additional procedures to obtain new knowledge.

2.2.2. Processing unit

This unit is made up by:

1. The **Server** (*S*), which is responsible for receiving events, which are sent by the Translators, and for updating dynamically the Warehouse of the Objects of the Scenario (creating new Objects or updating the existing ones).
2. The **Warehouse of the Objects of the Scenario** (*WOS*), which stores the different Objects that exist in the scenario in real time. All information that is obtained from the different Input Knowledge Extraction System along with the new information generated by our System, is stored and integrated in this warehouse. A process of mutual exclusion is performed for accessing the WOS with the aim of avoiding inconsistencies. This warehouse is characterized by the existence of fixed objects and dynamic objects, these last called here as moving objects:
   - *Fixed objects*: objects that are always stored in the warehouse. This is the case of the microphones and the sensors that monitor the environment, which are defined and registered in the System as fixed objects of the WOS. These objects are created when the scenario is designed and they can only be deleted when they are not part of the sources that monitor the environment.
   - *Moving objects*: objects that are dynamically created depending on their appearance on the scene. These objects are dynamically deleted if a certain time passes without receiving events on them. This is the case of people and vehicles detected from video analysis, since they appear temporarily in the video. It is also the case of identified sounds.
3. The **Alarm Detection Module** (*ADM*). This module is responsible for analyzing the presence of intrusions (see Section 4).
4. The **Objects Eliminator** (*OE*), which consists of a threat that is released regularly. This process is designed to verify that the objects in WOS are right objects of the real scene, in other words, if these objects are active objects. If it finds objects that have not been updated for some time, these objects will be considered as inactive objects on the scenario and they will be removed by the Object Eliminator. In order to perform this process, each object has an belief associated degree that reflects its activity in the scene. The existence of this procedure is very important because if there is not a good cleaning in the WOS, this fact could lead to detecting false alarms.
5. One **Plugin** that consists in an additional procedure that allows to relate events to the system objects. This plugin is described in the Section 5.
6. One **Alarm Notifier** in real time and context-sensitive. This notifier is described in the Section 6.

The communication between the modules of the architecture is implemented using event channels. In this case, we speak of events in the usual sense of the network services. In this way, information can be sent asynchronously. Customers or receivers can subscribe

to a channel and kept waiting for news without having to make a request. So, event channels provide an efficient method of change information. For perform this, we use the ZeroC Ice middleware (Henning, 2004).

2.3. System output

All information received is stored in our system and it is represented under the conceptual framework of our Ontology (see Section 3). In addition, these data can be supplemented with new information generated by our System. For example, the system can associate the events detected from audio or sensors analysis with other objects detected from video analysis (see Section 5). Finally, the final output obtained is a value that indicates the level of presence of an intrusion. This alarm level is a degree value that belongs to interval [0,1]. We have divided this interval into subintervals, which we have assigned a label depending on the danger. Thus, the alarm level determines the intrusion risk level in the scenario such as zero, low, medium and high. These states depend on a threshold value, which is a parameter that can be adjusted in the System. The threshold of the alarm also belongs to [0,1]. Besides, we have associated a color for each alarm-level-state.

- *White state: zero presence of intrusion,* when the alarm level is zero. No signs of intrusion.
- *Green state: low presence of intrusion,* when the alarm level belong to (0, threshold/2]. There exists a small possibility that there is an intrusion.
- *Yellow state: medium presence of intrusion,* when the alarm level belong to (threshold/2, threshold). There is an intrusion. When the color is yellow, the System emits a warning sound that attracts the human operator.
- *Red state: high presence of intrusion,* when the alarm level belong to [threshold, 1]. There exists a significant intrusion. In this case, the emitted alarm sound is more intense.

We have designed a desktop application where you can check the status of intrusion detection. In this application, the operator sees the development of alarm state by means of a graphic bar that changes the color and the size according to the level of alert (see Fig. 9). Besides, there exists other application *ad hoc* mobile devices where the intrusion alarm level is notified and is also showed by means of a graphic bar. The mobile application have been developed on *JavaME*,[3] using the configuration CLDC (*Connected Limited Device Configuration*) and it can be installed on any mobile phone with Wi-Fi (even on PDAs). In the Fig. 2 is shown the mobile application.

3. Knowledge representation

We differentiate three levels to the knowledge representation of the system:

1. Inputs representation.
2. Ontology for the homogeneous representation of knowledge.
3. Output representation.

3.1. Inputs representation

Next, we describe the input data structure that our system receives.

3.1.1. Representation of the video information

[3] http://phoneme.dev.java.net/.

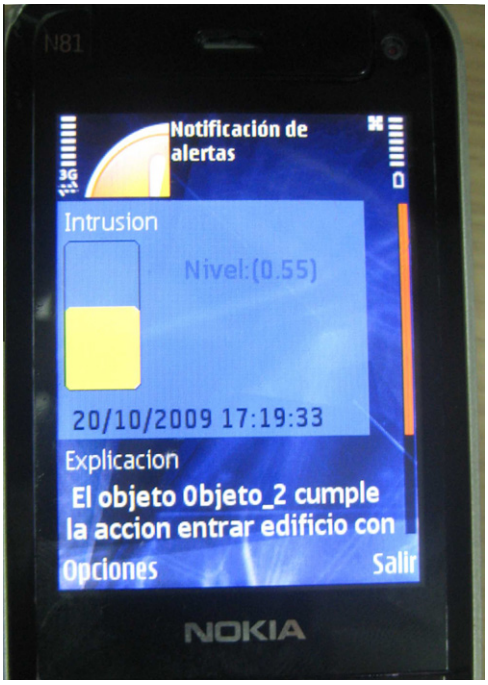

Fig. 2. Mobile application that shows the alarm level.

The stream of video events on the object detection and 2D-tracking consist of the following pattern (ci, f, t, ol) where:

- **ci** denotes the *camera identifier*.
- **f** is the *frame number*.
- **t** is *time*, which is measured in milliseconds.
- **ol** a *list of detected objects* in the current frame. We know of each object: its identifier, its 2D-position, its 2D-speed, its 2D-size and its classification into people and/or vehicles with a degree of belief.

On the other hand, the second stream of events from video analysis corresponds to the identification of object trajectories is represented by the pattern (id, t, pt, rt) where:

- **id** denotes the *object identifier*.
- **t** is the *time*, which is measured in milliseconds.
- **pt** are *the possible trajectories* that the id-object may be performing at the moment t. We know of each trajectories: its identifier and a degree of belief.
- **rt** are *the reconized trajectories* at the moment t, in other words, the trajectories that the object finishes at the time t.

3.1.2. Representation of audio information

The third stream that our system get as input is the classification of sounds. This stream is represented by the pattern (mi, t, s, db, pos) where:

- **mi** denotes the *microphone identifier*.
- **t** is the *time*, which is measured in milliseconds.
- **s** is *identifier of the classification* of the detected sound.
- **db** is *the degree of belief*, which belongs to the interval [0,1] and it is associated with the classification identified.
- **pos** is the *location* where this event is detected.

3.1.3. Representation of sensor information

This input stream on the sensor information is represented by the pattern (si, t, a, db) where:

- **si** denotes the *sensor identifier*.
- **t** is the *time*, which is measured in milliseconds.
- **a** is *identifier of the action* of the sensor: activation or deactivation.
- **db** is *the degree of belief*, which belongs to the interval [0,1] and it is associated with the action identified.

3.2. Ontology for the homogeneous representation of knowledge

We have designed an Ontology in order to define input heterogeneous information in a homogeneous way. The concept 'System Object' is formally defined in this Ontology. All information that we know (since it is captured by microphones, cameras or sensors) on the same moving object in the monitored scene is represented and integrated in a single object of the system using the following scheme:

An 'System Object' is defined as the pattern *(i, db, li, t, loc, q, a)*, where:

- **i** is the object identifier.
- **db** is a degree of belief, which varies between 0 and 1 and indicates the activity level of the object in the scene.
- **li** is a list with other possible identifiers of i-object.
- **t** is the time of last update of the object in the System (measured in milliseconds).
- **loc** represents the set of object locations within the scenario. A *location* is defined with a pattern (p, v, s, t, i) where: p is the position, v denotes the speed; s represents the object real size; t is the time when the object has this position;i indicates the increase of time since the last object location.
- **q** represents the object qualities. An *quality* is an attribute or property of an object. It represents as a triple (c, v, d), where: c denotes the class or type of quality; v is the quality value; d is the degree of belief of the quality (between 0 and 1). Examples: *("type", "vehicle", 0.7), ("type", "person", 0.2), ("sound", "loud", 0.9)*.
- **a** represents the object actions. An *action* is defined by a pattern (i, d, ti, tf) where: i denotes the action identifier; d is a degree of belief; ti represents the time when the action began; tf represents the time when the action finished. If the tf value is 0, it means that the id-action have not finished. Examples: *("to come in the garage",0.9,10:00 10/09/08, 10:01 10/09/08), ("sound of breaking glass",0.6,10:00 10/09/08, 10:00 10/09/08), ("motion activation",1.0,23:03 12/09/08, 0),*.

3.3. Outputs representation

Another important concept that is also defined is the concept of alarm. An alarm is represented as a pattern (i, db, u, t, eo, et) where:

- **i** is the alarm identifier.
- **db** is the degree of belief of the alarm. It varies between 0 and 1 and indicates the alarm level.
- **u** is the threshold. It belongs to the interval [0,1]. We consider that an alarm is activated when its degree of belief exceeds this threshold.
- **t** is the time of last update of the alarm (measured in milliseconds).
- **eo** is a structure that summarizes the explanation of the alarm activation. It consists in a peculiar sequence of system objects. This sequence consists only of those objects that affect the state of the presence of alarm, and besides, these objects have only those actions or qualities involved in that change in the level of alarm.
- **et** is the explanation of the alarm activation in text format.

4. Rule-based system for intrusion detection

In order to perform the analysis of intrusion detection, we have developed a rule-based system. This system is characterized as configurable because it is not a fixed system with fixed rules, but is an adaptive system where rules can be defined and adjusted according to circumstances. The need to develop an adaptable system arises because the detection of an intrusion is entirely context-dependent. It is that depending on the scenario, the intrusion is defined in one way or another. Even on the same scenario, there will be circumstances that make intrusion detection varies, for example, the definition of intrusion may change if it is a weekday or a holiday etc.

The model proposed here allows to create a set of rules, which describe what is considered intrusion in a particular scenario. These rules are defined by a expert from the desktop application (see Fig. 9). It is important to emphasize that the rules will be evaluated using as reference the objects participating in the current scene. These objects are the WOS objects, which keep up the conceptual scheme described in the Ontology. With each rule we can define what type of objects will increase the alarm level with a certain degree of relevance if they perform a particular action. These rules follow the pattern or logic that is shown below:

IF obj *is a* 'Object_Type' **AND** obj *performs* 'The_X_action' *taking into account a time value* 'ReferenceTime', **THEN** alarm_level *is increased by* a 'percentage'.

Where 'object_type', 'the_X_action', 'ReferenceTime' and 'percentage' are four variables that must be defined for each rule:

- *Object_Type* is a parameter that indicates the type of objects that are affected by the rule, for example, in our case can be: vehicles, people or sensors (considering the microphones and other sensors as 'sensors'). When this parameter is empty, it is considered that the rule affects all types of objects.
- *The_X_action* is a parameter that indicates the identifier of the action evaluated by the rule.
- *ReferenceTime* is a reference value, which is measured in milliseconds and it is used to obtain a degree of importance about the relevance of the action of study when is evaluated by an object. If 'The_X_action' is being currently performed by the object 'obj', then the relevance of this action is 1. However, if 'The_X_action' has been performed some time ago by the object 'obj', then this relevancy depends on the time elapsed since the action finished. In this case, the function shown in Fig. 3 is put into practice. This function obtains a degree of belief about 'How important an action is depending on time spent since

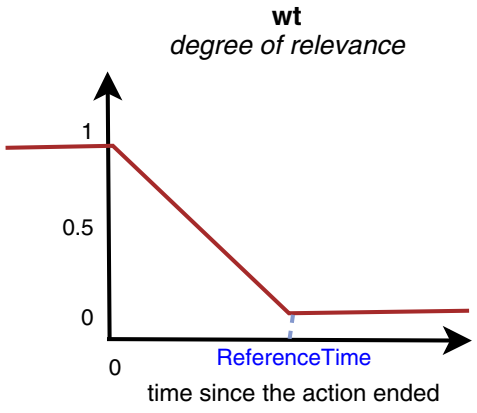

Fig. 3. Membership Function to the fuzzy concept 'How important an action is depending on time spent since the action finished'.

the action finished'. The value 'ReferenceTime' is used in this function to indicate the action is recent or not. If 'ReferenceTime' o more milliseconds pass from the action finished, then it is considerated that the action is not recent and the relevance is 0. This process is described in more detail in the Section 4.1.

- *Percentage.* It is the parameter introduced in the consequent of the rule. When a system object is an 'Object_Type', and further, it is performing or has recently performed the action with identifier 'The_X_action', then the alert level is increased according to a percentage value. In this case, this value is the parameter 'Percentage', which belongs to the interval [0,100].

Henceforth, we refer to *"obj is a 'Object_Type'"* as the first condition of the rule and *"obj performs 'The_X_action' taking into account a time value 'ReferenceTime'"* will be the second condition.

4.1. Rule assessment

The evaluation of the rule-based system is carried out each time that a change in the actions of the objects in the WOS is detected. The rules are evaluated one by one. In this subsection we explain how one rule is assessed by the WOS objects. For each rule, all objects in the WOS are checked in order to find how many of them assert the rule and change the alarm level.

When a rule is evaluated by a concrete object, we check that the two rule conditions are true for this object (*condition_1 ANDcondition_2*):

- First, we assess that the object has the same type as the type specified in the rule. The type of the object is stored as a *quality* (c, v, d) with the parameter c equals to *'type'*, the parameter v equals to the type classification (person, vehicle or sensor) and the parameter d equals to the degree of belief. In addition, the objects of the WOS can have different types associated. For example, the qualities of an object with several types associated can be: ("*type*", "*vehicle*", 0.2), ("*type*", "*person*", 0.8). So, we check that the object contains the same type specified in the rule, returning the belief degree as the fuzzy value of condition fulfilling (w0). If the type is not included in the object, w0 is zero.
- Next, we evaluate the second condition, which involves to check that the action specified in the rule is performing or has recently been performed by the object. In this process, all the actions with the same identifier that the identifier of the action specified in the rule are studied, since the object may have performed several times an action.
  In each action, we study the accuracy of two parameters: *a.degreeOfBelief* **AND** w_t. First, we consider the degree of belief with which the action has been identified (*a.degreeOfBelief*), and second, we studied the importance of action according to the time elapsed since it ended. If the action is currently being developed, the importance is 1, whereas if the action is already completed, the degree of relevance (w_t) is obtained depending on elapsed time and a reference value, *ReferenceTime*. 'How important an action is depending on time spent since the action finished' is a fuzzy concept. So, we use a function that allows us to obtain this importance degree, as this function gives us the action membership of this concept. In this function (see Fig. 3), the x variable represents the time that have passed from the action finished.

$$w_t = f(x) = \begin{cases} 1 \text{ if } x \leqslant 0 \\ \left(\frac{-1}{ReferenceTime} \cdot x\right) + 1 \quad if \quad 0 < x < ReferenceTime \\ 0 \text{ if } x \geqslant ReferenceTime \end{cases}$$

Algorithm 1: isPerformedTheAction (*obj*,*Action*)

Require: *obj*, *Action*, *ReferenceTime*
{*obj* is the object of study; *Action* is the action to check and *ReferenceTime* is a time used as reference value to obtain a degree of belief about how relevance the action of study is, according to the time elapsed since the action finished. Other parameters: w_t is the degree of relevance that indicates the importance of the action studied to an object according to this action is being performed by the object at the moment or it has been performed recently. w is the final degree of relevance of an action and indicates the relevance of the action according to both the degree of belief of the action and the time elapsed since the action finished, *wMax* is the maximum degree of relevance of all the degrees of relevance w that the evaluated actions have, *ElapsedTime* is the time elapsed since the action ended}
for $a = 1$ to *size*(*getActions*(*obj*)) **do**
  $w_t = 0$
  if $a \cdot id = Action \cdot id$ **then**
    if $a \cdot isFinished()$ **then**
      $ElapsedTime = CurrentTime - a \cdot getEndTime()$
      $w_t = getFuzzyRelevance\ Value(ElapsedTime, ReferenceTime)$(see Fig. 3)
    else
      $w_t = 0$
    end if
  end if
  w = MIN (w_t,a.degreeOfBelief)
  wMax = MAX (w,wMax)
end for
return wMax

When we obtain *a.degreeOfBelief* and w_t, we apply the AND operator using the MIN function. In this way, each object action is evaluated (see Fig. 4). After we evaluate all the object actions, we return the MAX value of them as the degree of belief with which the second condition rule is fulfilled, (see Algorithm 1).

After verifying that both conditions are true for a concrete object *obj*, we apply the AND operator using the MIN function. So, we know the degree of belief associated with the rule for the

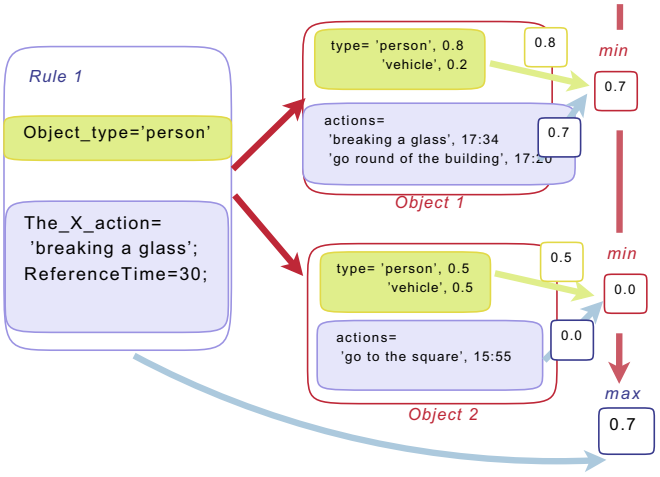

Fig. 4. Assessment of one rule with a concrete object.

evaluation of a specific object (*wR*). However, this rule is evaluated for each one of the objects in the WOS. Thus, we keep the maximum of the values obtained for the different objects (OR operator). This maximum value is the final value of belief associated with the rule (*wMax*), which represents the riskest object that assert the rule.

4.2. Increase of alarm level

Finally, once it is known that the rule is fulfilled with a concrete value *wMax*, the system increases the alarm level. This increase is depending on *wMax* and *percentage*, which is the parameter introduced in the consequent of the rule:

$$AlarmLevel = AlarmLevel + increase(wRMax, r.Percentage)$$

where,

$$increase(wRMax, r.Percentage) = (1 - AlarmLevel) \cdot wMax \cdot Percentage$$

All rules change the alarm level in a independent way (see Fig. 5). The rule-based model follows the algorithm shown in Algorithm 2.

Algorithm 2: AlgorithmforRuleAssessment (*WOS*, *R*)

Require: *WOS*, *R*
{*WOS* is the set of objects belonging to the Scenario-Object-Warehouse and *R* is the Rule-set. Other parameters: *w*0 is the degree of belief associated with fulfilling the first condition of the rule when is assessed by a concrete object, *w*1 is the degree of belief associated with fulfilling the second condition of the rule for a concrete object, *wR* is the degree of belief associated with fulfilling the rule for a concrete object, *wRMax* is the maximun of all the degrees of belief that the rule takes when is evaluated with different objects,*AlarmLevel* is the level of presence of the intrusion alarm}
AlarmLevel = 0
for *r* = 1 to *size*(*R*) **do**
wRMax = 0
for *obj* = 1 to *size*(*WOS*) **do**
*w*0 = *isFulfilledTheTypeOfObject*(*obj*,*r.Object_Type*)
*w*1 = *isPerformedTheAction*(*obj*,*r.The_X_action*) (see Algorithm 1)
wR = AND (w0,w1)
wRMax = OR (wR,wRMax)
end for
AlarmLevel = *AlarmLevel* + *increase*(*wRMax*,*r.Percentage*)
end for

Thanks to the value *ReferenceTime* the system automatically decreases the level of alert when a situation changes. The duration of active alarm depends on this value, and may be adjusted according to needs. When much time elapses since a dangerous action ends, this action ceases to be relevant. This means that the second condition of the rule is not satisfied, and therefore the value of rule fulfilling is very low and does not affect the level of alarm.

Another important aspect is that the parameter *Percentage* can be positive or negative. If it is positive, the alert level is increased. However, if it is negative, the alert level is decremented. The latter case corresponds to actions that can soften a dangerous situation.

One of the great advantages of our rule-based system is that it generates automatically an explanation through the actions that have changed the alert level. Thus, the system generates a description, which is understandable to humans.

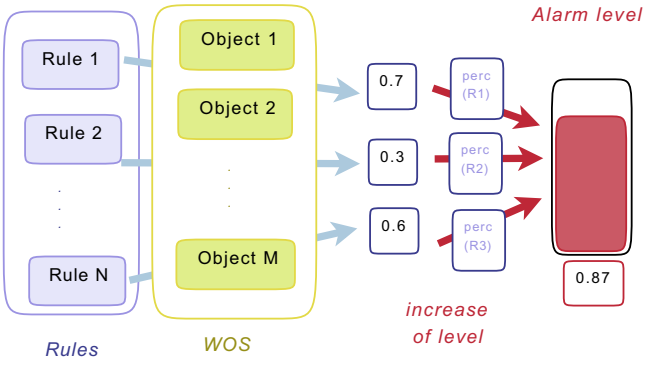

Fig. 5. Assessment of the rules and update the alarm level.

5. Plugin to relate events to system objects

For further information from one or more Objects, we have defined the Plugins. They allow to associate new knowledge dynamically depending on the objects that exist in the WOS.

In this case, we have defined a Plugin, which is an additional process able to relate sound events or sensor events to people or vehicles. The procedure is easy: both when a sound is detected and identified and a sensor detects something, it is checked if there exists some moving object in the system that is close to the position where the event has been detected.

First, is calculated the dist between the event detected and the all the current moving objects. Next, it is checked if the obtained dist indicates that the object and the event are near. 'To be near' is a fuzzy concept. So, we use a function that indicates the degree of membership (*w*) of *dist* with respect to the concept 'near', (see Fig. 6). We rely on a reference distance *d*. With this function, we assume that if distance between two positions is less than $\frac{d}{2}$, they are nearby. In contrast, if the distance exceeds the amount of $\frac{d+d}{2}$, probably they will not be nearby. In intermediate cases, it will be decreasing the certainty that they are close.

After the below function is put into practice, the action corresponding to the event detected is associated to the objects that are considered as closed objects to this event. Thus, our System allows to know a possible list of suspicious, which may be the cause of the event detected. For example, if a breakage glass is detected when is captured by the microphone, then the System analyse if there are people o vehicles that are near the position where this event has been identified. In the affirmative case, these objects have a new action associated: 'the breakage of glass'.

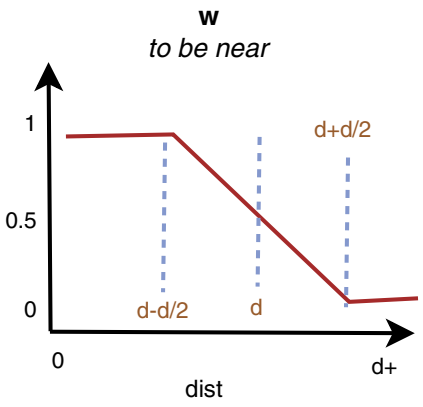

Fig. 6. Membership Function to the fuzzy concept 'near'.

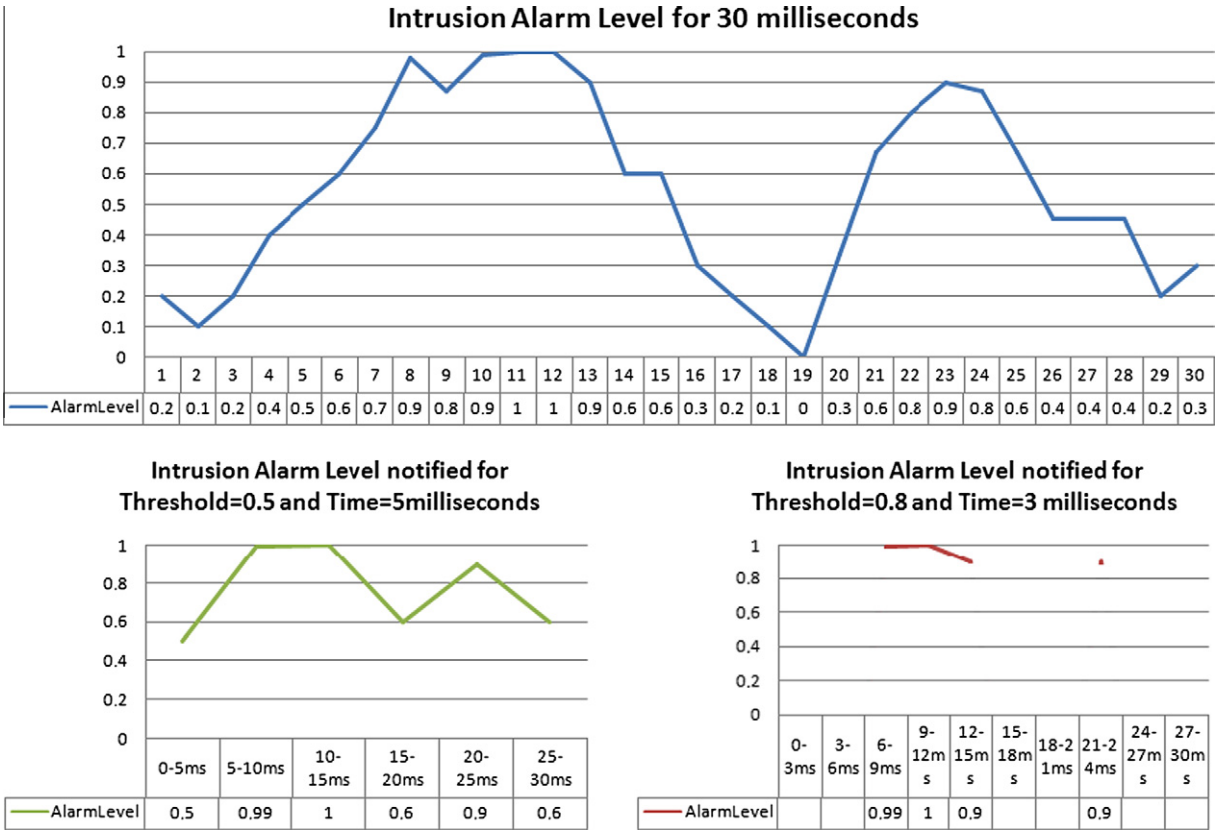

Fig. 7. Alarm notification: two examples on different contexts.

Fig. 8. Test scenario.

6. Alarm notifier in real time and context sensitive

As we detailed above, the system generates an alarm when an intrusion is detected. In order to several receptors know the alarm state, there is a process called *Alarm Notifier* that is responsible for publishing the alarm in real time.

The subscription is a prerequisite for a client can receive notifications. This process is performed using the *ZeroC Ice middleware* (Henning, 2004). In the system, there is a server listening to subscriptions and when it receives one, the server notifies the client of the port where he must listen. In a subscription, the subscriber must specify three parameters:

- *IdAlarm*: is the identifier of the alarm to which he subscribes.
- *Threshold*: is a value between 0 and 1. This data means that the user wants to receive information only when the alarm level exceeds this threshold.
- *Time*: is the number of milliseconds that indicates how often the user want to be informed. If this parameter is greater than 0, the notification will contain the maximum level of alarm in that time period.

If the subscriber wants to be informed of all changes, the Threshold and Time parameters (specified above) must be 0. These parameters are adjusted depending on client device capability or

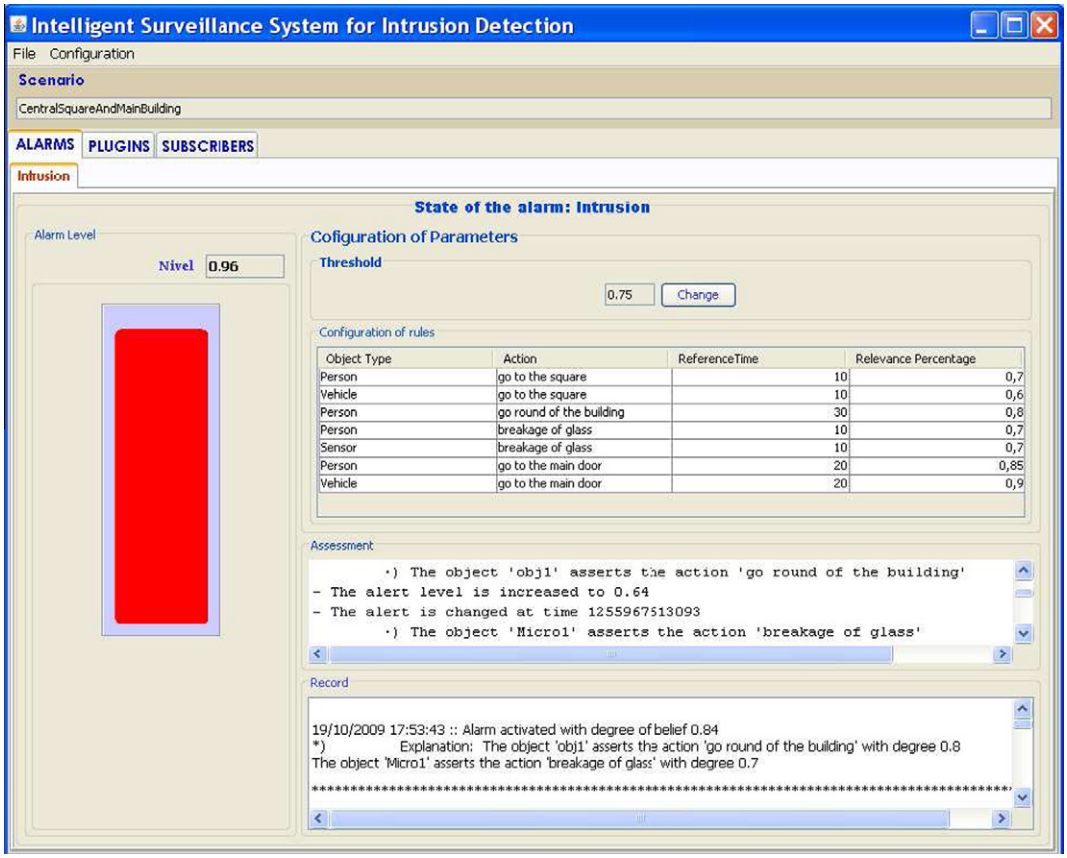

Fig. 9. Desktop application.

Table 1
Variables of the rules.

	Object_type	The_X_action	ReferenceTime	Percentage
Rule 1	Person	Go to the square	10	0.7
Rule 2	Vehicle	Go to the square	10	0.6
Rule 3	Person	Go round of the building	30	0.8
Rule 4	Sensor	Breakage of glass	30	0.8
Rule 5	Person	Breakage of glass	30	0.8
Rule 6	Person	Go to the main door	20	0.85
Rule 7		leave the Square	5	−0.80

the circumstances of the user. In this way, we can observe that the notification process will be context sensitive (in accordance with the needs and context of the subscriber). In the Fig. 7 is shown an example about the alarm level that the System notifies two subscribers. One of them wants to receive information every 5 ms and only when the alarm level is higher than 0.5. The other subscriber is informed every 3 milliseconds as long as the alarm level exceeds 0.8.

In the alarm notifier there is a thread that handles the different times of the subscribers and it wakes up when it is need to notify the alarm status to a client. The notification process is by means of *User Datagram Protocol* (*UDP*) (Postel, 1980), that is, the server will notify the client of the alarm level using UDP datagrams. We use UDP because offers a transmission speed better than TCP (*Transmission Control Protocol*).

UDP is an Internet Protocol Suite that sends messages without implicit hand-shaking dialogues for guaranteeing reliability, ordering, or data integrity. Time-sensitive applications often use UDP because dropping packets is preferable to using delayed packets. UDP is located in the transport layer. In turn, RTP was defined to deliver audio and video over Internet and it is part of application layer. This protocol provides payload-type identification, sequence numbering, time stamping or delivery monitoring. Both are complementary, because to solve the UDP lacks, the RTP is located over UDP providing order and check.

7. Experimental results

The system has been tested on a specific scenario. It aims to identify intrusions on a holiday where the staff of the company do not work on campus but other people or vehicles may access, since in one of the buildings there is a museum can be visited on holidays. It is only necessary to monitor a portion of the enclosure where there is a square that gives access to the main building of the headquarters. This is the local scenario that our system will analyze (see Fig. 8). The test scenario is monitored by a surveillance camera, a microphone and a motion sensor, strategically distributed in the environment.

In this case, it is considered intrusion:

- Any person or vehicle that has access to the central square
- The loitering of any person that goes round the building
- Identification of movement at the front door
- Identification of a gunshot or an explosion of glass

The rules that describe this situation are given in Table 1. The rules allow to define when an intrusion is more important than another, for example, it is not the same as a vehicle going to the square and parking by mistake, than if a person prowls the building and breaks a window to go inside.

Table 2
Experimental results.

Example number	Intrusion	Maximum alarm level detected by our system
1	Yes	0.8
2	Yes	0.9
3	Yes	1.0
4	Yes	0.7
5	Yes	0.83
6	Yes	0.8
7	Yes	0.76
8	Yes	0.32
9	Yes	1.0
10	Yes	0.8
11	Yes	0.8
12	Yes	0.9
13	Not	0.3
14	Not	0.4
15	Not	0.0
16	Not	0.0
17	Not	0.2
18	Not	0.0
19	Not	0.0
20	Not	0.0

We have simulated 20 examples of situations that can occur in our scenario selected. Between these examples, there are 12 intrusions and 8 no intrusions. For do this, we have simulated annotated-video events, sound events and motion-sensor events as system inputs. The defined rules have been sufficient to detect intrusions in the vast cases, which shows a high performance of the System (see Table 2). As it occurs in the real life, a high number of data known make that the detection of presence of intrusions is more notable. In the Fig. 9 is shown an execution system for an example.

We have also developed a study on the system run times. To do this, we used the previous proof examples. The computer used is a Pentium (R) Dual-Core CPU E5200@2.50 GHz 2.51 GHz, 2.75 GB RAM. We have carried out three experiments.

First, we measured the time that the middleware spends since an input event is sent until it is captured by the translators of our system. The maximum elapsed time detected is 16 ms. The average is 0.62 ms, with a standard deviation of 3.04 ms. With these data, the middleware can process until 62.5 events per seconds without causing delays.

Second, we measured processing times of the system, from an event is collected by the translators until the system obtains the level of alert generated by this event. The maximum time spent in processing is 188 ms. The average is 36.11 ms, with a standard deviation of 28.60 ms. We observe that the delay is short, indicating that the system is efficient.

Finally, we have carried out another study about the times obtained if the number of rules varies. These results are shown in the Fig. 10. We can see that the average times are increased with increasing the number of rules. However, we observe that the data obtained by changing the number of rules are not significant. As we can see, increasing the number of rules affects in a few milliseconds processing time of the system. This means that the system is scalable, since the introduction of new rules has an little effect on evaluation time of the system.

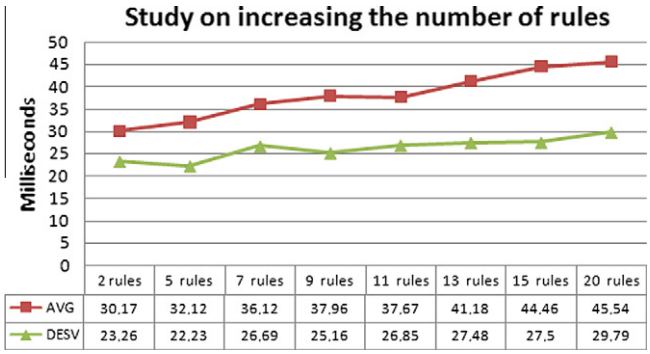

Fig. 10. Data obtained by changing the number of rules.

8. Conclusions and future work

In this paper, we propose an intelligent surveillance system, which integrates heterogeneous information from different sources using different technologies: fuzzy logic, ontologies, information notifiers sensitive-context in real time and handheld devices. The strong point of the system is the combination of video-audio-sensors analysis. In this way, it carries out an integration process of information and it builds a more powerful surveillance system because there is more available information at the decision making process. We propose an Ontology for the Knowledge Representation. This Ontology allows to represent the input heterogeneous information in a homogeneous way and integrates all information known (from the different sources) about one scene object in only one system object.

Thus, this system is a good tool for improving the security of the people and the infrastructures, since the system alerts in real time of the presence an intrusion in a building or a concrete zone. In addition to reporting alarms to a desktop application, where the operator can check the presence of the situation, the alarms will also be notified via mobile devices. In this way, the operator can be informed without being in front of the command post. Besides, when an intrusion is detected, the system offers an explication about the events involved in this detection.

To analyze the presence of intrusions, we propose a model based on rules, which are characterized by being easily customizable and adjustable. This model lets you configure rules based on the scenario and circumstances. The rules define an intrusion in a semantic way (according to the actions of the objects). Moreover, this system has been designed to be robust to fuzzy information. Through the use of fuzzy logic the model output, in this case the detection of intrusions, is gradual.

The alarm notification is done in real time. Therefore, we have developed a sensitive-context alarm notifier, which is able to transmit the notifications according to the subscriber needs. In this way, the subscriber can decide when he wants to be informed and from what alert level.

If it is intended that the proposed system can be used in a wide-area, then the global environment will be divided into monitored scenarios. And the System will be instantiated as many times as there are scenarios. The rules will be described in a different way in each scenario, that is, to each system instance. Thus, currently various intrusion detections in different scenarios are independent, although in some cases are the same intrusion, in other words, the individuals who create the intrusion are the sames. As future work we want to achieve the tracking of an intrusion at different scenarios. In addition, we will develop new Alarms Detection Modules to carry out identification of new risk situations.

Acknowledgments

The presented research has been realized within the HESPERIA project, http://www.proyecto-hesperia.org, financed by the Centre for the Industrial Technological Development (CDTI) across programmes CENIT and for the companies: Indra, Union Fenosa, Tecnobit, Visual-Tools, BrainStorm, SAC and TechnoSafe.

References

Atrey, P. K., Maddage, N. C., & Kankanhalli, M. S. (2006). Audio based event detection for multimedia surveillance. In *ICASSP06*.

Bauckhage, C., Hanheide, M., Wrede, S., & Sagerer, G. (2004). A cognitive vision system for action recognition in office environments. In *Proceedings of the IEEE computer society conference on computer vision and pattern recognition*.

Behrad, A., Shahrokni, A., & Ahmad, M. (2001). A robust vision-based moving target detection and tracking system. In *Proceeding of image and vision computing conference*.

Bo, L., Qimei C., & Fan, G. (2006). Freeway auto-surveillance from traffic video. In *6th international conference on ITS telecommunications proceedings* (pp. 167–170).

Borg, M., Thirde, D., Ferryman, J., Fusier, F., Valentin, V., Bremond, F., et al. (2005). Video surveillance for aircraft activity monitoring. In *Proceedings of IEEE conference on AVSS* (Vol. 1, pp. 16–21).

Castro, J. L., Delgado, M., Medina, J., & Ruiz-Lozano, M. D. (2011). An expert fuzzy system for predicting object collisions. Its application for avoiding pedestrian accidents. *Expert Systems with Applications, 38*(1), 486–494.

Clavel, C., Ehrette, T., & Richard, G. (2005). Events detection for an audio-based surveillance system. In *IEEE international conference on multimedia and expo* (Vol. 0, pp. 1306–1309).

Collins, R., Lipton, A., Kanade, T., Fujiyoshi, H., Duggins, D., Tsin, Y., et al. (2000). A system for video surveillance and monitoring. Technical report CMU-RI-TR-00-12, Robotics Institute, Carnegie Mellon University.

Fernández-Caballero, A., Gómez, F. J., & López-López, J. (2008). Road-traffic monitoring by knowledge-driven static and dynamic image analysis. *Expert Systems with Applications, 35*, 701–719.

Fusier, F., Valentin, V., Brmond, F., Thonnat, M., Borg, M., Thirde, D., et al. (2007). Video understanding for complex activity recognition. *Machine Vision and Applications, 18*(3), 167–188.

Haritaoglu, I., Harwood, D., & Davis, L. S. (2000). W^4: Real-time surveillance of people and their activities. *Patter Analysis and Machine Intelligence, 22*(8), 809–830.

Henning, M. (2004). A new approach to object-oriented middleware. *IEEE Internet Computing, 8*, 6675.

Hudelot, C., & Thonnat, M. (2003). A cognitive vision platform for automatic recognition of natural complex objects. In *International conference on tools with artificial intelligence, ICTAI, Sacramento* (pp. 398–405).

Marchesotti, L., Messina, A., Marcenaro, L., & Regazzoni, C. S. (2003). A cooperative multisensor system for face detection in video surveillance applications. *Acta Automatica Sinica, 29*(3), 423–433.

Mittal, A., & Paragios, N. (2004). Motion-based background subtraction using adaptive kernel density estimation. In *Proceedings of the IEEE computer society conference on computer vision and pattern recognition* (Vol. 2, pp. II302–II309).

Patricio, M. A., Carbó, J., Pérez, O., García, J., & Molina, J. M. (2007). Multi-agent framework in visual sensor networks. *EURASIP Journal on Advances in Signal Processing, 2007*, 21. doi:10.1155/2007/98639.

Porikli, F., Ivanov, Y., & Haga, T. (2008). Robust abandoned object detection using dual foregrounds. *EURASIP Journal on Advances in Signal Processing*.

Postel, J. (1980) User Datagram Protocol. RFC from Internet Society. RFC 768.

Prati, A., Cucchiara, R., & Vezzani, R. (2007). A multi-camera vision system for fall detection and alarm generation. *Expert Systems, 24*(5), 334–345.

Valera, M., & Velastin, S. A. (2005). Intelligent distributed surveillance systems: A review. In *IEEE proceedings vision, image and signal processing, iee proceedings* (Vol. 152, supplement 2, pp. 192–204).

Yuan, X., Sun, Z., Varol, Y., & Bebis, G. (2003). A distributed visual surveillance system. In *Proceedings of IEEE conference on advanced video and signal based surveillance* (pp. 199–205).