Expert Systems with Applications 39 (2012) 6982–6993

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

Human activity monitoring by local and global finite state machines

Antonio Fernández-Caballero [a,b,*], José Carlos Castillo [b], José María Rodríguez-Sánchez [b]

[a] Departamento de Sistemas Informáticos, Escuela de Ingenieros Industriales de Albacete, Universidad de Castilla-La Mancha, 02071 Albacete, Spain
[b] Instituto de Investigación en Informática de Albacete (I3A), Universidad de Castilla-La Mancha, 02071 Albacete, Spain

ARTICLE INFO

Keywords:
Video
Moving objects
Human activities
Visual surveillance

ABSTRACT

There are a number of solutions to automate the monotonous task of looking at a monitor to find suspicious behaviors in video surveillance scenarios. Detecting strange objects and intruders, or tracking people and objects, is essential for surveillance and safety in crowded environments. The present work deals with the idea of jointly modeling simple and complex behaviors to report local and global human activities in natural scenes. Modeling human activities with state machines is still common in our days and is the approach offered in this paper. We incorporate knowledge about the problem domain into an expected structure of the activity model. Motion-based image features are linked explicitly to a symbolic notion of hierarchical activity through several layers of more abstract activity descriptions. Atomic actions are detected at a low level and fed to hand-crafted grammars to detect activity patterns of interest. Also, we work with shape and trajectory to indicate the events related to moving objects. In order to validate our proposal we have performed several tests with some CAVIAR test cases.

1. Introduction

Many techniques and methods have been used so far in human activity recognition and understanding (Huang, Lee, Kuo, & Lee, 2010). In human activity understanding, four broad classes can be identified (Dee & Hogg, 2009): learning patterns of activity within a scene (e.g. (Liu & Yuen, 2010; Makris & Ellis, 2005; Stauffer & Grimson, 2000)); modeling the interactions between agents (e.g. (Oliver, Rosario, & Pentland, 2000)); hand-crafting models of particular activities of interest (e.g. (González, Rowe, Varona, & Roca, 2009; Jan, Piccardi, & Hintz, 2002)); and intentional systems (e.g. (Liao, Patterson, Fox, & Kautz, 2007; Bui, Venkatesh, & West, 2002)).

In close relation to finite state machines theory, Ayers and Shah (2001) describe a system which automatically recognizes human actions from video sequences taken of a room. These actions include entering a room, using a computer terminal, opening a cabinet, picking up a phone, etc. The system recognizes these actions by using prior knowledge about the layout of the room. Indeed, action recognition is modeled by a state machine, which consists of 'states' and 'transitions' between states. The transitions from different states can be made based on a position of a person, scene change detection, or an object being tracked. Hongeng, Nevatia, and Bremond (2004) model scenario events from shape and trajectory features using a hierarchical activity representation, where events are organized into several layers of abstraction, providing flexibility and modularity in modeling scheme. An event is considered to be composed of action threads, each thread being executed by a single actor. Amer, Dubois, and Mitiche (2005) propose a real-time system to detect context-independent events in video shots. The authors test it in video surveillance environments with a fixed camera. They assume that objects have been segmented (not necessarily perfectly) and reason with their low-level features, such as shape, and mid-level features, such as trajectory, to infer events related to moving objects. The authors classify events into four types: primitive, action, interaction, and composite. Fernández, Baiget, Roca, and González (2011) present an ontology-based methodology that guides the identification, step-by-step modeling, and generalization of the most relevant events to a specific surveillance domain. Gómez-Romero, Patricio, García, and Molina (2011) propose a computer vision framework aimed at the construction of a symbolic model of the scene by integrating tracking data and contextual information.

Modeling human activities with state machines is still common in our days. A recent system (Arens, Gerber, & Nagel, 2008) aims at an encompassing conceptual description of video sequences. This ultimately culminates in the generation of encompassing natural language textual descriptions of a scene observed by a computer vision system. Also Hamid et al. (2009) investigate modeling activity sequences in terms of their constituent subsequences that we call event n-grams. Exploiting this representation, the authors

* Corresponding author at: Departamento de Sistemas Informáticos, Escuela de Ingenieros Industriales de Albacete, Universidad de Castilla-La Mancha, 02071 Albacete, Spain. Tel.: +34 967 599200; fax: +34 967 599224.
E-mail address: Antonio.Fdez@uclm.es (A. Fernández-Caballero).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2012.01.050

propose a computational framework to automatically discover the various activity-classes taking place in an environment. Our approach is closely related to the works of Ivanov and Bobick (2000) and Hongeng et al. (2004), in the sense that the external knowledge about the problem domain is incorporated into the expected structure of the activity model. Motion-based image features are linked explicitly to a symbolic notion of hierarchical activity through several layers of more abstract activity descriptions. Atomic actions are detected at a low level and fed to hand-crafted grammars to detect activity patterns of interest. Our inspiration also is close to the paper by Amer et al. (2005), as we work with shape and trajectory to indicate the events related to moving objects.

The rest of the paper is organized as follows. Section 2 introduces the theoretical aspects of our proposal based in the description of local and global activities. In Section 3 the specification of simple and complex behaviors are described in detail. Then, Section 4 offers the data and results for several problems with scene analysis taken with surveillance video cameras. Lastly, in Section 5 some conclusions are offered.

2. Description of local and global activities

The last part of the automatic analysis of video sequences is the interpretation of the different behaviors present in the sequence Gascueña and Fernández-Caballero (2011). To do so, we start from the input data generated by the previous analysis processes, namely, segmentation (e.g. Fernández-Caballero, López, & Saiz-Valverde, 2008; López, Fernández-Caballero, Fernández, Mira, & Delgado, 2006; López-Valles, Fernández, & Fernández-Caballero, 2007) and tracking (e.g. (Fernández-Caballero, Gómez, & López-Lápez, 2008; López, Fernández-Caballero, Fernández, Mira, & Delgado, 2007)). Therefore, analyzing a video scene entails two large sections. On the one hand, we have the first phase in object detection. This phase consists of capturing images, analyzing them for shape interpretation and afterwards, recognizing them throughout the scene. On the other hand, we have the part this work focuses on, namely, scene interpretation (context recognition), made up of *basic actions* interpretation, *global behavior* interpretation and finally, interpretation of the *scene on a global scale*. The quality of the result will depend greatly on the quality of the geometric analysis, that is, on the quality of the shape analysis in the video scene. Thus, incorrect input date would hinder good scene interpretation.

These objects are not fixed in the scene and can move over time in each frame. These objects are called dynamic or mobile objects. They perform the actions within the scene, whether individually or as a group. Similarly, other type of passive objects can intervene in a scene. These objects are always in the scene and do not move overtime. These are called, static objects. These objects can also be considered areas within a scene. Many times, dynamic objects interact with static objects, approaching them, placing themselves on them and using them. Some examples of static objects are a seat, a door a wastebasket or a relevant area in the scene in which we wish to analyze whether the dynamic objects are approaching it or placing themselves in it.

In our proposal, the purpose of activity description is to reasonably choose a group of motion words or shout expressions to report activities of moving objects or humans in natural scenes. Generating semantic description is the final goal of human motion analysis (Ji & Liu, 2009).

2.1. Objects of interest

These input data will consist of a series of objects detected in the scene, which should be properly classified. It is necessary to define, from within the possible objects, a series of categories which will help us to identify the activities carried out in a scene. Of course, the objects of interest can be static or dynamic and the latter will perform the actions in the scene. Inspired in ETISEO classification (http://www-sop.inria.fr/orion/ETISEO/index.htm), four categories will be established for dynamic objects and two for static objects. As for the first, we will distinguish between a *person*, a *group of people* (made up of two or more people), a *portable object* (such as a brief case) and other dynamic objects (able to move on its own), classified as *moving object*. As for static objects, we will distinguish between *areas* and *pieces of equipment*. The latter can be labeled as a portable object if a dynamic object, people or group interact with it and it starts to move. We should explain that an area is defined as a non-dynamic object in the scenario where people or dynamic objects are likely to interact. For instance, an automatic cash point has an area people will approach to interact.

Before proceeding to list the activities, we should clarify which we are interesting in addressing. Specifically, and once again analyzing ETISEO's property classification, we will focus on those related to kinematics of objects, leaving those related to size and shape for subsequent works. It is also necessary to clarify that for our testing purposes, the type of dynamic objects we are interesting in detecting are people or groups of people.

2.2. Description of local activities

In order to generalize the detection process for complex behavior patterns as much as possible, we will start with small functionalities which, individually, detect simple actions of the active objects in the scene. Using these functions, we will build behavior patterns much more complex and suited for each video surveillance system's aims. These small actions are defined by action indicative queries about the actions performed by an active object. The queries should be enough to describe most of the behavior patterns of the dynamic objects.

Therefore, several types of queries are defined for the detection of the basic actions of the active objects in the scene. Other queries are linked to the interaction between an active object and the scenario and the interaction between the dynamic objects in the scene. Approaches to activity modeling have been mostly relied upon segmentation and tracking of objects in the scene (e.g. (Haritaoglu, Harwood, & Davis, 2000; Stauffer & Grimson, 2000)) after efficient motion detection (e.g. Martínez-Cantos, Carmona, Fernández-Caballero, & López, 2008; Mira, Delgado, Fernández-Caballero, & Fernández, 2004). This is due to the fact that activities have traditionally been considered to be discriminated by the trajectories of object motion, modeled either statically as templates or dynamically as state machines (Xiang & Gong, 2006).

2.2.1. Object-like actions

We first develop the most basic actions of each object, which do not entail environment interaction and which only depend on the object itself to be analyzed. Next, we show some possible basic actions detectable by a video surveillance system:

- *Object speed*. The speed of an object (in displacement units per second, for instance) makes it possible to define if an object is still, walking, running, going at great speeds, etc. In order to determine real object speed in the scene, it is necessary to mitigate possible lens distortion and to interpret the real positions, depending on the distance from the camera. The data taken as input are obtained from a video camera and, therefore, only contain two-dimensional information.
- *Object trajectory*. Apart from speed, we can obtain the direction and moving direction of an object. Let us assume that a road is being analyzed. The speed of a moving vehicle is very useful but

without calculating the trajectory, it is not possible, for instance, to know if the vehicle is going the wrong way. It is also important to detect an abrupt change in direction.

2.2.2. Actions of environment interaction

Basic actions are very important when determining what is happening in a scene. But in order to interpret more complex behaviors, it is also necessary to analyze the interaction between dynamic objects and the scenario. To interpret environment interaction, it is essential to create a small model of the environment. The areas where objects interact with the scenario are defined in each scenario in particular. These areas can be doors, wastebaskets, cash points, and so on. Furthermore, the parameters associated to the interaction objects in the scenario are activated, such as the place where they are located or their size, as well as any information necessary to enable behavior detection in which these areas or static objects intervene. Some possible scenario interaction behaviors might be:

- *Direction*. The system must determine if a person is approaching a specific area of the scenario. By taking the object's speed and trajectory as reference, the object's ultimate goal is inferred.
- *Position*. By knowing the important areas of the scenario, the system is capable of determining the relative position of dynamic objects. This way, it can detect if a person is standing in one of the areas. For example, it is possible to know if a person is sitting on a bench or when analyzing traffic flow, if a car is parked in a pedestrian crossing. We also know the distance from the objects to these areas, thus determining if the object is near these passive elements or areas of the scenario.

2.2.3. Actions involving object interaction

With what we have seen so far, it is possible to identify object behavior in the scene that has to do with the condition of the object itself and its interaction with the scenario; however, it is not possible to interpret interaction between the active objects themselves. To be able to interpret interaction between people, a video surveillance system has to detect behaviors such as:

- *Proximity*. The system must detect the distance between objects. This way, we can determine whether an object is near or far away from the rest. If time is also taken into account, we can determine whether a specific object is getting closer to or moving away from another one.
- *Orientation*. By taking into account the direction of objects, the system determines whether an object is approaching another or whether they are both approaching each other. These data also help to determine whether the objects have the intention of joining or, on the contrary, moving away.
- *Grouping*. The system uses the parameters generated in the two previous points to detect object grouping. By taking into account its proximity and direction, the system can interpret whether an object is with another. For example, it can detect whether two objects are going to the same place together.

2.3. Description of global activities

Interpreting a visual scene is a task which in general resorts to a large body of prior knowledge and experience of the viewer (Neumann & Möller, 2008). Through the actions or queries described in the previous section, we can find out basic patterns, such as whether an object is moving quickly or slowly, whether or not it is going towards a specific place, etc. We usually wish to detect more complex patterns, such as the theft of a purse or of a car in a parking lot. In this type of more complex events, several dynamic objects that interact with each other or with other static objects usually participate. These complex behavior patterns will vary in each scenario according to the type of event you wish to detect. Thus, you will want to detect different behaviors in a parking lot from those in a bank. Therefore, it is essential to define the behavior pattern desired in each situation, by using the basic actions or queries from the previous section. For each specific scene, a state diagram and a set of rules are designed to indicate the patterns.

In many cases, it is not only necessary to detect the individual behavior of an object, but it is also of great interest to discover patterns for the whole set of objects in the scene. More than one object participates in the global behavior patterns. For instance, you might want to detect if a group of people are running around a public square. In this case, a single person running may not be of interest but if one-hundred people are running, something serious has likely happened. To interpret these situations in a scene, the system takes into account the individual behavior of each object, as well as the interaction between objects and between the objects and the scene.

2.4. Alarm system

A great deal of patterns can be detected in a scene but they are not all necessarily dangerous or require special attention. For instance, if a person is walking, behavior "Walking" has to be detected by the scene interpretation application, although it does not require special attention. Detecting behavior patterns will not be useful without a system to select those patterns and rule out the most common and harmless. A smart video surveillance system must consider those patterns by assigning a level of alarm to each of the possible behaviors in the scene. Different alarm thresholds will be used to filter out suspicious patterns. Thus, the system alerts the user when it detects a behavior with an associated alarm level greater than the allowed threshold.

Once we have detected the state of the objects, it is time to find out if any of them deserve special attention on the part of the surveillance system operators. To do this, the guards will have an associated alarm level which weighs out the significance of each likely state of the object and scene. It can have different values (e.g. from *I* to *V*) and it is directly associated to each of the vertices in the state machines for local pattern as well as global pattern detection.

To sum up, the proposed video surveillance system will be able to detect simple actions or queries and adapt to a great deal of situations. Also, it will be configured to detect the behavior patterns necessary in each case and associate an alarm level to each one which will enable them to be filtered and have a priority associated to them.

3. Specification of simple and complex behaviors

3.1. Simple behavior specification

As anticipated, the system should be able to respond to a series of queries intended to find out behavior patterns of objects in the scene. These queries are defined as functions and return a logical value, which will be true if they are fulfilled for a specific object. They are represented in the following format:

$$query(parameter_1, parameter_2, \ldots, parameter_n),$$

where *query* is the name of the action or behavior to be tested and $parameter_i$ are the possible parameters necessary to specify what we want to analyze.

3.1.1. Movement-based queries

Next, we will number the basic queries the system recognizes, the parameters needed, how they were resolved and the data

and hypothesis necessary in each case. To identify them, we use the object speed hypothesis.

- *hasSpeedBetween* (*min*,*max*). It is fulfilled if the object moves at a speed within the range [*min*,*max*].
- *hasSpeedGreaterThan* (*speed*). It is fulfilled if the object moves at a speed greater than that indicated in the parameter *speed*.

3.1.2. Orientation-based direction queries

To analyze these queries, we use the displacement angle of objects hypothesis.

- *hasDirection* (*staticObject*). It is fulfilled if the object is headed towards *staticObject*, being *staticObject* a static object in the scene.
- *isFollowing* (). It is true if a dynamic object is following a non-dynamic object. To know whether an object in the scene is following another one, we use the displacement angle. This way we can calculate if the target is another dynamic object.

3.1.3. Location-based queries

This type of queries is based on the location of objects in the scene. To analyze them, we use the location in each frame obtained directly from the input data.

- *isInsideZone* (*staticObject*). It is true if a dynamic object is on the static object *staticObject*.
- *isCloseTo* (*distance*,*staticObject*). It is fulfilled if the object is closer than distance from the static object *staticObject*.
- *enterInScene* (). It is fulfilled when the object appears in the scene for the first time.

3.2. Complex behavior specification

On the one hand, the behavior patterns at a local level have been defined, which focus on the behavior of objects in the scene at each time instant; and patterns at a global level, used to analyze the scene from a general point of view without focusing on any specific object. This last specification will be used to detect patterns where more than one object intervenes.

3.2.1. Local complex behaviors

Objects in the scene are associated to a state machine that indicates the state they are in (what they are doing at that time instant). This diagram is the same for all objects in the scene and it will depend on the target of the analysis in the video scene. This state machine can be seen as a directed graph where the vertices are the possible states of the object and the edges are the basic functions or queries previously discussed. This graph will be similar for all objects but each object can be found in a different state independently. The source of the objects is a vertex, which will be the initial state (**Initial**). More than one edge can come in and out of each vertex and there is no destination vertex. An object cannot be in several states at the same time.

An edge has at least one associated outcome of the assessment of a query with the parameters previously defined, indicating an action of object, query q_i. These assessments can be true or false. Thus, for some transitions to come about, certain actions will not be fulfilled. For an object to change states, many times one query is not enough. It needs to fulfill several of them to go on to the following state. Therefore, an edge can have more than one query associated to it. For an edge with several actions to be fulfilled, all the associated queries have to be fulfilled. The following type of nested rules can be associated to this type of edges:

$$IF\ (q_1\ AND\ q_2\ AND \ldots q_k),\ THEN\ transit\ to\ vertex_X$$

where $q_1 \ldots q_k$ are the possible queries associated to a vertex and $vertex_X$ where the state transits to.

If a more complex rule is needed, where disjunctions also appear so that an object changes states, as the following:

$$IF\ ((q_1\ AND\ q_2\ AND \ldots q_n)\ OR\ (q_{n+1}\ AND\ q_{n+2}\ AND \ldots q_{n+m})),\ THEN\ transit\ to\ vertex_X$$

the rule must be divided into two edges that come from the same vertex and get to $vertex_X$. The first one will be associated to actions $q_1 \ldots q_n$ and the second one to actions $q_{n+1} \ldots q_{n+m}$. When dividing the rule into two, we get a disjunction, since if one is fulfilled the object goes onto the following state.

Fig. 1 shows, as an example, the local state machine in a simple scene where we attempt to find out if any object is close to a cash point. Notice that the objects in the scene can be walking, standing still, approaching the cash point or be at the cash point. These are the different states the object goes through by means of the state machine. When an object appears in the scene for the first time, it is in the **Initial** state. If "hasSpeedGreaterThan (1)" is fulfilled, i.e., the speed of the object is greater than one meter per second, then it will go on to the **Walking** state. The object changes states according to its behavior, following the same procedure. In the edge that goes from vertex 4 to vertex 3, there is more than one query. As explained above, both actions must be fulfilled, in this case, in order to change states. Later on, we will see more complex cases and how they were solved.

3.2.2. Global complex behaviors

Unlike in local behavior patterns, more than one object intervenes in a global behavior pattern. To detect these patterns, more than just the local state machine from the previous section is needed since only the state of each object in that machine is reflected separately.

To represent this type of patterns, global state machines are used. These state machines are made up of vertices. Each vertex represents a possible state in the scene. Just like in the local state machine, the edges are made up of a series of queries that must be fulfilled at a certain time for the scene to change states. These types of queries do not make reference to a single object but to all of them. Next, we will list the queries used for detecting this type of patterns:

- *isTimeInState* (*state*,*time*). It is fulfilled when an object remains in the same *state* longer than what is suggested in *time*.
- *areObjectsInState* (*state*,*min*,*max*). It is fulfilled if the number of objects in *state* is between *min* and *max*.
- *areMoreObjectsInState* (*state*,*number*). It is true if the number of objects is greater than *number* in the *state*.

Just like in local behaviors, queries can be used using negation. Next (see Fig. 2), we will see a simple global state machine that detects if someone approaches someone else while he/she is at a cashpoint. If it is necessary to detect several global behavior patterns simultaneously in a video scene, we must create several specialized state machines to detect each of those patterns. Thus, a scene could have several global state diagrams, each one focusing on detecting one type of behavior pattern.

4. Data and results

In the following section, we will present several problems with scene analysis taken with surveillance video cameras and their resolution with the proposed system. We have opted for working with

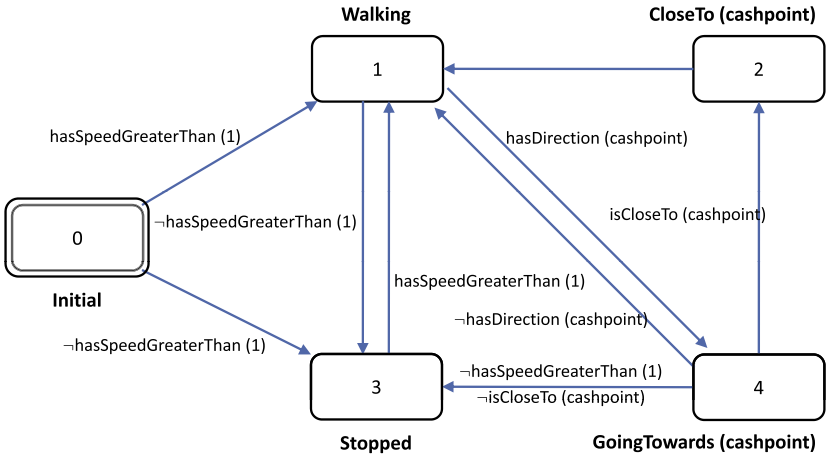

Fig. 1. Local state diagram.

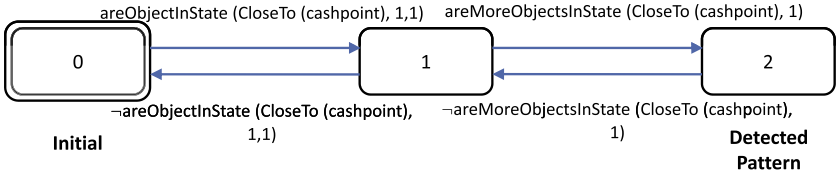

Fig. 2. Global state diagram.

the ground truth of manually labeled images, frame by frame, whereby errors in data entry will be null. Therefore, we have used the test cases that CAVIAR (coming from the EC Funded CAVIAR project/IST 2001 37540, found at URL: http://homepages.inf.ed.ac.uk/rbf/CAVIAR/) makes available for researchers. This is a usual approach (e.g. (Blunsden & Fisher, 2009), where the detection and classification of fighting and pre and post fighting events when viewed from a video camera is investigated). For the tests, we have used specific videos recorded for project CAVIAR recorded with a wide angle camera in the lobby of INRIA Labs in Grenoble, France. In these scenes, there are different people interacting with the environment and with each other. They are videos recorded with video surveillance cameras supported with manual labeling of the objects in each frame and with estimations about the state of each one of them. These input data are in XML format and they include the objects that makeup the frames in the video scene, their position, size and a label to find them in subsequent frames.

Thus, the application obtains a series of objects in each frame in the XML file. However, the static objects are not included in the CAVIAR XML file. Moreover, they will be different in each scene so they will have to be specified in each case, indicating their position and size, just like for dynamic objects. On the other hand, with the data read from the XML in CAVIAR, we have information about the objects in each scene, their position and size; however, to carry out the analysis and meet the goals set, we will need other data not included. These are speed and direction of objects. The system takes the initial data and infers the new data geometrically.

We begin by showing the configuration common to all case studies to avoid repeating it in each section. Next, the case studies carried out will be listed, beginning with the basic cases where the running and usefulness of each query is shown individually and separately and continuing with the more complex cases where two or more queries are required.

4.1. Image preprocessing

We select the first frame in any scene as backdrop image to make the placement of control points and fixed objects easier.

4.1.1. Creation of point maps

Ideally, placing control points is done manually, by directly measuring the scenario where the camera is placed but to speed up the process, the data in the CAVIAR web site are used and the points that do not appear are approximated. Table 1 shows the data provided in the CAVIAR web site:

The rest of the positions are interpolated using these data as a guide. Enough points have been placed to cancel image distortion caused by the perspective and camera lens. Fig. 3a shows where these position points have been placed.

4.1.2. Specification of fixed objects in the scene

After creating the point map, we point out the fixed objects in the scene that will interact with the dynamic objects (Fig. 3b). As we can see, the entrances to the scenario and the most relevant objects, such as a cashpoint, wastebasket or seats have been tagged in the image. A list of objects tagged in the scene can be seen in Table 2, along with their positions and sizes.

Once the static objects in the scene and the control point maps are stored in the configuration file, the next step is the creation of state diagrams to complete the configuration in the scene.

4.2. Motion detection

We design a configuration capable of detecting when someone starts walking and he/she stops. With this first case, we show the performance of the queries "hasSpeedBetween" and "hasSpeedGreaterThan", which make reference to the movement of people. In this case, we do not wish to detect a pattern for which it is

Table 1
Data provided in the CAVIAR web site.

Point	(column, row) in pixels	(x,y) in cm
1	(64,88)	(0,671.5)
2	(211,40)	(1116,670)
3	(349,184)	(1545,190)
4	(39,187)	(0,0)

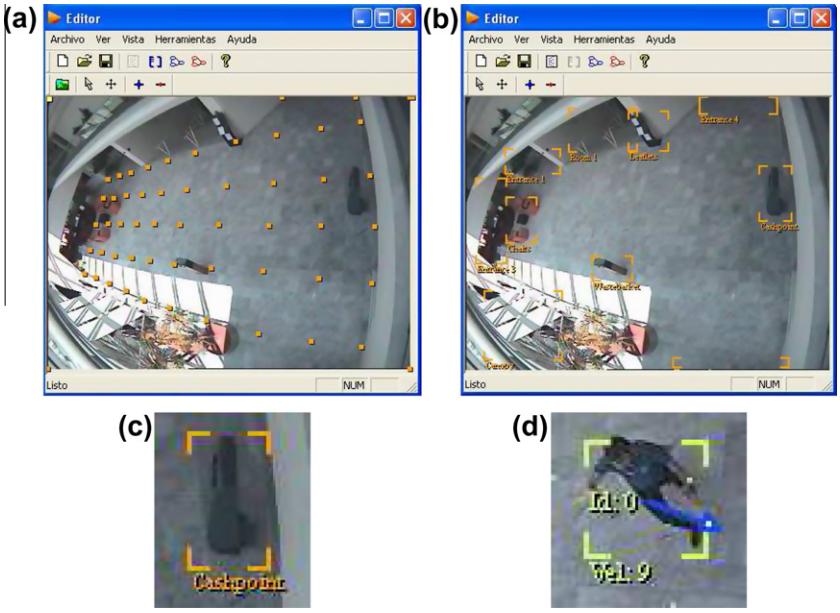

Fig. 3. Test environment for CAVIAR (a) position point maps. (b) Fixed objects in the scene (c) Labeling of a static object in the scene (d) Labeling of a dynamic object in the scene.

Table 2
Fixed objects.

Object	(column, row) in pixels	(width, height) in pixels
Cashpoint	(327,102)	(32,57)
Entry 1	(71,68)	(57,25)
Entry 2	(280,280)	(120,9)
Entry 3	(27,131)	(30,89)
Entry 4	(288,9)	(80,16)
Wastebasket	(155,181)	(40,25)
Seats	(59,130)	(30,46)
Awnings	(61,241)	(80,70)
Room 1	(145,35)	(70,44)
Leaflets	(193,36)	(40,40)

necessary to check the global state in the scene. We can see how detecting a person in a scene creates a state machine with an initial state in the **Initial** state. If the system detects that a person is stopped the transition is to vertex **Stopped**. If it detects motion, it will enter **Walking** state. The object will change states depending on whether it starts moving or it stops. Table 3 shows all the details of the automaton.

For queries "hasSpeedBetween" and "hasSpeedGreaterThan", the input parameters indicate how many kilometers per hour the object travels. To consider a person to be in motion, his speed must be greater than 1 km per hour and to consider him stopped, his speed must be 0. This has been designed this way to avoid false positives that can take place when a person opens his arms or makes a gesture with his body, which can be interpreted as motion.

If we analyze the first few frames in scene `Walk1` (series *Walking*, test case *One person walking - straight line*) we get the results shown in Table 4.

4.3. Speed detection

In this case, we do not just detect if a person is in motion or not. We also detect if the person starts running or moves slowly. To do this, we use the same queries as in the previous case, "hasSpeedBetween" and "hasSpeedGreaterThan", but a more complex local state diagram is necessary.

When a person appears on the scene, the system enters **Initial** state. From here, he/she can move straight into any other state. However, if the person is stopped, he/she will first have to go through **Wandering**, before going into any other state. Likewise, if he/she is running, he/she will first have to walk slowly before stopping. This way, we avoid illogical and unlikely leaps, such as a person going from stopped to running and vice versa. Table 5 shows every detail of the automaton. In this case, we use a higher alarm level for the **Running** state. Thanks to this, the data provided to the user can be filtered so that it will only signal when somebody starts running.

Table 5 shows that a person is considered to be stopped if his speed is 0, to be walking slowly if his speed is between 1 and 2 kilometers per hour, to be walking at a normal pace if his speed is between 4 and 10 km per hour and to be running if his speed is over 10 km per hour. As in the previous case, to change states we have kept a margin to prevent getting too many false positives.

When adjusting the alarm level to *I* and analyzing scene `Browse2` (series *Browsing*, case *Person browsing and reading for a while*), we get the output shown in Table 6.

If the alarm level is adjusted to *II* and scene `Fight_RunAway1` (series *Two people fighting*, test case *Two people meet, fight and run away*) is analyzed, we get Table 7 as output. As shown, the application has detected the time the two people started running.

Table 3
Local state diagram, motion detection.

Action	Origin vertex	Destination vertex	Associated verbs	Alarm level
Stationary	Initial	Stopped	hasSpeedBetween (0,0)	I
Walks	Initial	Walking	hasSpeedGreaterThan (1)	I
Stops	Walking	Stopped	hasSpeedBetween (0,0)	I
Starts	Stopped	Walking	hasSpeedGreaterThan (1)	I

Table 4
Results of motion detection in scene "Walk1".

Time	Object	State	Alarm
0:00:01	0	Walking	I
	4	Stopped	
0:00:02	5	Stopped	I
0:00:03	4	Walking	I
0:00:04	0	Stopped	I
	5	Walking	
0:00:05	0	Walking	I
0:00:06	4	Stopped	I
	5	Stopped	
	5	Walking	
0:00:07	5	Stopped	I
0:00:09	1	Stopped	I
0:00:10	1	Walking	I
0:00:11	4	Walking	I
0:00:12	5	Walking	I
	4	Stopped	
	5	Stopped	
0:00:17	5	Walking	I
0:00:18	5	Stopped	I
	5	Walking	
0:00:19	1	Stopped	I
	5	Stopped	

Table 5
Local state diagram, speed detection.

Action	Origin vertex	Destination vertex	Associated verbs	Alarm level
Stationary	Initial	Stopped	hasSpeedBetween (0,0)	I
Wanders	Initial	Wandering	hasSpeedBetween (1,3)	I
Walks	Initial	Walking	hasSpeedBetween (4,10)	I
Runs	Initial	Running	hasSpeedGreaterThan (10)	II
Starts	Stopped	Wandering	hasSpeedGreaterThan (1)	I
Accelerates and walks	Wandering	Walking	hasSpeedGreaterThan (3)	I
Accelerates and runs	Walking	Running	hasSpeedGreaterThan (10)	II
Decelerates and walks	Running	Walking	SpeedLessThan (8)	I
Decelerates and wanders	Walking	Wandering	SpeedLessThan (2)	I
Stops	Wandering	Stopped	hasSpeedBetween (0,0)	I

4.4. Direction detection

Continuing with simple case studies, in this section, a configuration is designed to detect when someone goes to the cashpoint. To do this, we use query "hasDirection" which is able to detect when a dynamic object approaches a static object. The designed graph is very simple, having only two vertices and two edges. This diagram could get more complex if we were to add motion detection because for someone to go to a place, he/she first has to be moving. This situation will be analyzed in the section of complex case studies, where the combinations of different types of queries will be studied. The details of the automaton are described in Table 8 where the name of the static object referred to is used as the parameter for query "hasDirection".

The test is carried out with scene `Browse2`, obtaining the output shown in Table 9, where the application has detected when an object goes towards a cashpoint. Furthermore, Fig. 4 shows the video output where the hypothesis of a person's direction has been tagged with a red line.[1]

Table 6
Results of speed detection in scene "Browse2".

Time	Object	State	Alarm
0:00:00	0	Stopped	I
	1	Stopped	
	1	Wandering	
0:00:01	1	Walking	I
0:00:04	1	Wandering	I
0:00:05	1	Stopped	I
	1	Wandering	
0:00:06	1	Walking	I
0:00:07	2	Walking	I
	2	Wandering	
	2	Stopped	
0:00:09	1	Wandering	I
	2	Wandering	
0:00:12	3	Walking	I
	3	Wandering	
0:00:13	3	Walking	I
0:00:14	3	Wandering	I
0:00:15	3	Walking	I
0:00:21	3	Wandering	I
0:00:22	3	Stopped	I
0:00:30	3	Wandering	I
	3	Walking	
0:00:33	3	Wandering	I
	3	Stopped	

4.5. Position detection

When detecting object position in an image, it is necessary to design a configuration with which to analyze a person's relative position to the static objects in the scenario. To do this, queries "isInsideZone" and "isCloseTo" are used. The configuration designed is able to detect when a person is in the seats area, when he/she is in the awning area, when he/she is close to the leaflets or when he/she is near the cashpoint. Obviously, the automaton can be widened to cover all areas and static objects in the scenario. Table 10 shows the details of this automaton.

As shown in Table 10, in this case, we have used query "isInsideZone" to locate people inside certain areas (seats) and query "isCloseTo" to detect if the person is next to a static object in the scenario, such as the cashpoint, the leaflets and the wastebasket. The numerical parameter of "isCloseTo" is the maximum distance in pixels to be considered close. Somewhat of a safety margin has been kept to avoid false positives when considering whether an object is close or not to the cashpoint and the leaflets.

Next, three different scenes will be analyzed to check the results from position detection. When analyzing scene `Browse2` we get Table 11 as output. Scene `Browse3` (series *Browsing*, test case *Person browsing and reading with back turned*) was also analyzed and yielded Table 12. Finally, after analyzing scene `Rest_InChair` (series *Resting, slumping or fainting*, test case *Person resting in chair*) we get Table 13.

The results show how the application has detected when a person has placed himself in the different position selected in the configuration.

4.6. Following detection

Using query "isFollowing", it is possible to detect whether a person is following another one (see Table 14). To detect a tracking

[1] For interpretation of colour in Fig. 4, the reader is referred to the web version of this article.

Table 7
Results of speed detection in scene "Fight_RunAway1".

Time	Object	State	Alarm
0:00:15	7	Running	II
0:00:16	6	Running	II

Table 8
Local state diagram, direction detection.

Action	Origin vertex	Destination vertex	Associated verbs	Alarm level
Walks to the cashpoint	Initial	GoingTowards (cashpoint)	hasDirection (cashpoint)	I
Wanders	Initial	Wandering	hasSpeedBetween (1,3)	I
Does not walk to the cashpoint	GoingTowards (cashpoint)	Initial	¬hasDirection (cashpoint)	I

Table 9
Results of direction detection in scene "Browse2".

Time	Object	State	Alarm
0:00:16	3	GoingTowards (cashpoint)	I

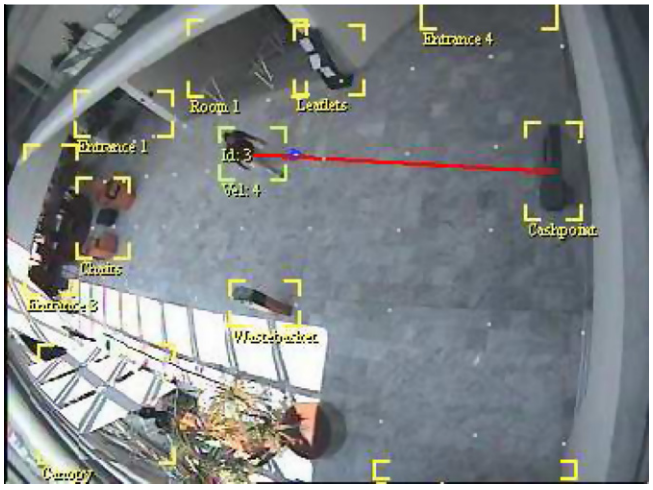

Fig. 4. Detected target.

situation, it would be ideal to be able to specify who is participating; however, since they are dynamic and not static objects, we have opted for only indicating whether there is tracking or not in the scene. Thus, the designed configuration indicates if someone is headed towards another person. This is useful when detecting interaction between people.

Scene `Fight_RunAway1` was used for the tests and the alarm level was set on *III*. The results in Table 15 show that when two people are about to meet, they walk towards each other.

4.7. Appearance detection

In this case, we have designed a configuration able to detect when an object appears on the scene by using query "enterInScene". In this case, only one query is used, although as we will see in the compound case section, by using this query along with others (forming more complex diagrams), we can detect the appearance of suspicious objects in the scene. The state machine

Table 10
Local state diagram, position detection.

Action	Origin vertex	Destination vertex	Associated verbs	Alarm level
Is close to the seats	Initial	InsideZone (seats)	isInsideZone (seats)	I
Is not close to the seats	InsideZone (seats)	Initial	¬isInsideZone (seats)	I
Is close to the wastebasket	Initial	CloseTo (wastebasket)	isInsideZone (wastebasket)	I
Is not close to the wastebasket	CloseTo (wastebasket)	Initial	¬isInsideZone (wastebasket)	I
Is close to the cashpoint	Initial	CloseTo (cashpoint)	isCloseTo (25, cashpoint)	I
Is not close to the cashpoint	CloseTo (cashpoint)	Initial	¬isCloseTo (30, cashpoint)	I
Is close to the leaflets	Initial	CloseTo (leaflets)	isCloseTo (25, leaflets)	I
Is not close to the leaflets	CloseTo (leaflets)	Initial	¬isCloseTo (30, leaflets)	I

Table 11
Results of position detection in scene "Browse2".

Time	Object	State	Alarm
0:00:21	3	CloseTo (cashpoint)	I

Table 12
Results of position detection in scene "Browse3".

Time	Object	State	Alarm
0:00:20	1	CloseTo (leaflets)	I

Table 13
Results of position detection in scene "Rest_InChair".

Time	Object	State	Alarm
0:00:16	1	InsideZone (seats)	I
0:00:31	1	InsideZone (seats)	I

Table 14
Local state diagram, following detection.

Action	Origin vertex	Destination vertex	Associated verbs	Alarm level
Pursues	Initial	Following	isFollowing ()	III
Does not pursue	Following	Initial	¬isFollowing ()	I

is provided in Table 16. In this case, only one edge was used because the objects can only appear in the scene once.

Scene `Walk1` was used for the tests, obtaining Table 17 as output, which shows the time every object appears on the scene.

4.8. Number of objects in the same state

It is possible to use the system to detect whether more than one person is walking or is stopped in the scene. To do this, we use rules with queries "areObjectsInState" and "areMoreObjectsInState" to analyze the number of objects found in a certain state in the scene. Up to now, configurations that make reference to the individual state of the objects participating in the scene have been designed, but in this case, a more global vision to analyze

Table 15
Results of following detection in scene "Fight_RunAway1".

Time	Object	State	Alarm
0:00:14	6	Following	III
0:00:11	7	Following	III

Table 16
Local state diagram, appearance detection.

Action	Origin vertex	Destination vertex	Associated verbs	Alarm level
Appears	Initial	NewObject	enterInScene ()	I

Table 17
Results of appearance detection in scene "Walk1".

Time	Object	State	Alarm
0:00:00	0	NewObject	I
0:00:01	4	NewObject	I
0:00:02	5	NewObject	I
0:00:09	1	NewObject	I

objects in the scene from a higher level is necessary. To detect whether more than one person is walking in the scene, we have the configuration provided in Table 18.

We have used query "areMoreObjectsInState (Walking, 1)", which is fulfilled when there is more than one object in state **Walking** of the local state diagram, and query "¬areMoreObjectsInState (Walking, 2)", which is fulfilled if there are fewer than 2 objects in state **Walking**.

Analogously, it is possible to detect whether there are more than one object in state **Stopped** of the local state diagram by using query "areMoreObjectsInState (Stopped, 1)". Query "¬areMoreObjectsInState (Stopped, 2)", which is fulfilled if there are fewer than 2 objects in state **Stopped**, takes you back to state **Initial**. A level *II* of alarm was assigned to the vertices of the two previous graphs to filter out the alarms of the local state diagram and be able to listen only to the new added verifications. The configuration provided in Table 19 was used to detect whether there is more than one person stopped in the scene.

When analyzing scene `Browse_WhileWaiting1` (series *Browsing*, test case *Person browsing while waiting short*), using both automata simultaneously, we get the output shown in Table 20.

4.9. Position and direction analysis

A configuration was designed for these purposes, which is able not only to analyze the position of people in a scene, but also to predict if someone is headed towards a specific position. Queries "isInsideZone" and "isCloseTo" are used to detect position and query "hasDirection" in order to generate a direction hypothesis. This type of configuration is useful to detect things such as someone entering a private area, since the behavior can be detected before it happens. If it does happen, we can trigger an alarm with more priority which will, for example, make the security officers intervene. Table 21 shows the automaton that detects if someone is headed towards or is at the wastebasket, the leaflets, the seats or the cashpoint. As shown, it is logical that in order for someone to be in a place, he/she has to head towards it first. Thus, we create several loops of the type:

Initial → GoingTowards (X) → InsideZone (X) → Initial

or

Initial → GoingTowards (X) → CloseTo (X) → Initial

A greater level of alarm was assigned to the vertices related to position than to those related to direction to be able to filter them when you only want to know when someone is situated somewhere. Also, notice that to consider that an object is headed towards an area, it is not enough for query "hasDirection" to be fulfilled, but also for "hasSpeedGreaterThan (1)" to be fulfilled; that is, for its speed to be greater than 1 km per hour. This guarantees that for a person to go to a place, he/she first has to be moving.

Tests on three different scenes have been run: from the analysis of scene `Rest_InChair`, we get the results found in Table 22; from the analysis of scene `Browse2` we get Table 23; and after analyzing scene `Browse3` we get the output shown in Table 24.

There are false positives in the last two tests. They are in the 14th second of test case `Browse2` and in the 15th second of test case `Browse3`. Indeed, object 3 was not going to the wastebasket but the direction of the object at that time made it seem like it could be going there. To avoid this, we could add another rule to edge direction to avoid predicting a possible target if the object is too far away. We could add an "isCloseTo" query to act along with queries "hasDirection" and "hasSpeedGreaterThan".

4.10. Possible cashpoint holdup detection

It is also possible to design configurations able to detect suspicious behaviors. Here is an example pertaining to a cashpoint. First, a local state diagram is created to detect the different ways of getting to the cashpoint. With this graph, we will be able to know is someone is going to the cashpoint, how fast he/she is going and if he/she is already next to the cashpoint. In Table 25, more importance has been attached to the fact that a person is walking slowly or running to the cashpoint than when he/she is walking at a normal pace. This is so in order to confront more suspicious behaviors.

Table 18
Global state diagram, objects in motion.

Action	Origin vertex	Destination vertex	Associated verbs	Alarm level
Two or more objects are moving	Initial	TwoOrMoreObjectsMoving	areMoreObjectsInState (Walking, 1)	II
Less than two objects are moving	TwoOrMoreObjectsMoving	Initial	¬areMoreObjectsInState (Walking, 2)	I

Table 19
Global state diagram, stopped objects.

Action	Origin vertex	Destination vertex	Associated verbs	Alarm level
Two or more objects are still	Initial	TwoOrMoreObjectsStill	areMoreObjectsInState (Stopped,1)	II
Less than two objects are still	TwoOrMoreObjectsStill	Initial	¬areMoreObjectsInState (Stopped,2)	I

Table 20
Results of objects in the same state in scene "Browse_WhileWaiting1".

Time	Object	State	Alarm
0:00:06	Scene	TwoOrMoreObjectsMoving	II
0:00:09	Scene	TwoOrMoreObjectsStill	II
0:00:11	Scene	TwoOrMoreObjectsMoving	II
0:00:18	Scene	TwoOrMoreObjectsStill	II
0:00:20	Scene	TwoOrMoreObjectsMoving	II
0:00:22	Scene	TwoOrMoreObjectsStill	II
0:00:24	Scene	TwoOrMoreObjectsStill	II
0:00:25	Scene	TwoOrMoreObjectsMoving	II

Once the local state diagram has been created, we go onto behavior pattern specification at global level in the scene. We want to detect suspicious behaviors, such as when there is someone at the cashpoint and someone else approaches him slowly. It can also detect if there is someone at the cashpoint and one or more people run towards him. Lastly, it can detect possible vandalism at the cashpoint. It will detect if one or more people run to the cashpoint and there is no one using it. We have decided to use high alarm levels because of the relevance of those patterns (see Table 26). Depending on what you want to analyze, the alarm level displayed will be adjusted to *II*, for example, to see all changes of state or to *V* to see only the alarms related to possible assaults to the cashpoint.

4.11. Suspicious objects

This section shows the design of a configuration able to detect the appearance of potentially suspicious objects in the scene. It detects when someone leaves something on the floor and later picks it up (the objects stops being suspicious). These types of patterns can be used, for example, to warn security officers that someone might have left an explosive device in a public place. The local state diagram described through Table 27 will detect if an object is suspicious (by having been placed near the cashpoint or next to the wastebasket), if it is a bag on the floor (because it has not been moved since it was placed there) or if it is simply someone who was motionless.

There are no loops in Table 27. Once an object gets to vertex **MovingObject** it stops drawing attention because it is not a static object but a person. To consider an object to be very dangerous, it cannot move after appearing. If the object appears next to the cashpoint or to the wastebasket, it will automatically be considered suspicious, and very suspicious, if it remains there for more than 30 seconds. If the object appears anywhere else, it will have to remain still for 30 s to be considered suspicious and 60 to be considered very suspicious (see Table 27).

When analyzing scene `LeftBag` (series *Leaving bags behind*, test case *Person leaving bag by wall*), with an alarm filter *IV*, we get the output shown in Table 28. Notice that in the 38th second, a suspicious object has been detected. After 15 seconds, in the 53rd second, the suspicious object has become very suspicious.

If looking at scene `LeftBag_PickedUp` (series *Leaving bags behind*, test case *Person leaving bag but then pick it up again*), where someone leaves a bag next to the cashpoint but later picks it up again is analyzed, we get the output shown in Table 29. When the bag is left on the floor but not by the cashpoint or the wastebasket, it is not considered a dangerous object. However, after being still for more than 15 s, it becomes suspicious at the 35th second. The person that left the object there picks it up again before being considered very dangerous.

Table 22
Results of position and direction analysis in scene "Rest_InChair".

Time	Object	State	Alarm
0:00:13	1	GoingTowards (seats)	II
0:00:16	1	InsideZone (seats)	III

Table 23
Results of position and direction analysis in scene "Browse2".

Time	Object	State	Alarm
0:00:14	3	GoingTowards (wastebasket)	II
0:00:16	3	GoingTowards (cashpoint)	II
0:00:21	3	CloseTo (cashpoint)	III

Table 24
Results of position and direction analysis in scene "Browse3".

Time	Object	State	Alarm
0:00:15	3	GoingTowards (cashpoint)	II
0:00:20	3	GoingTowards (leaflets)	II
0:00:20	3	CloseTo (leaflets)	III

Table 21
Global state diagram, position and direction analysis.

Action	Origin vertex	Destination vertex	Associated verbs	Alarm level
Walks to the seats	Initial	GoingTowards (seats)	hasDirection (seats)	
			hasSpeedGreaterThan (1)	II
Does not walk to the seats	GoingTowards (seats)	Initial	¬hasDirection (seats)	I
Is in the zone of seats	GoingTowards (seats)	InsideZone (seats)	isInsideZone (seats)	III
Is not in the zone of seats	InsideZone (seats)	Initial	¬isInsideZone (seats)	I
Walks to the wastebasket	Initial	GoingTowards (wastebasket)	hasDirection (wastebasket)	
			hasSpeedGreaterThan (1)	II
Does not walk to the wastebasket	GoingTowards (wastebasket)	Initial	¬hasDirection (wastebasket)	I
Is close to the wastebasket	GoingTowards (wastebasket)	CloseTo (wastebasket)	isCloseTo (25,wastebasket)	III
Is not close to the wastebasket	CloseTo (wastebasket)	Initial	¬isCloseTo (30,wastebasket)	I
Walks to the cashpoint	Initial	GoingTowards (cashpoint)	hasDirection (cashpoint)	
			hasSpeedGreaterThan (1)	II
Does not walk to the cashpoint	GoingTowards (cashpoint)	Initial	¬hasDirection (cashpoint)	I
Is close to the cashpoint	GoingTowards (cashpoint)	CloseTo (cashpoint)	isCloseTo (25,cashpoint)	IV
Is not close to the cashpoint	CloseTo (cashpoint)	Initial	¬isCloseTo (30,cashpoint)	I
Walks to the leaflets	Initial	GoingTowards (leaflets)	hasDirection (leaflets)	
			hasSpeedGreaterThan (1)	II
Does not walk to the leaflets	GoingTowards (leaflets)	Initial	¬hasDirection (leaflets)	I
Is close to the leaflets	GoingTowards (leaflets)	CloseTo (leaflets)	isCloseTo (25,leaflets)	III
Is not close to the leaflets	CloseTo (leaflets)	Initial	¬isCloseTo (30,leaflets)	I

Table 25
Local state diagram, holdup at a cashpoint.

Action	Origin vertex	Destination vertex	Associated verbs	Alarm level
Wanders to the cashpoint	Initial	GoingTowardsSlowly (cashpoint)	hasDirection (cashpoint) hasSpeedBetween (1,3)	III
Does not wander to the cashpoint	GoingTowardsSlowly (cashpoint)	Initial	¬hasDirection (cashpoint)	I
Walks to the cashpoint	Initial	GoingTowards (cashpoint)	hasDirection (cashpoint) hasSpeedBetween (4,10)	II
Does not walk to the cashpoint	GoingTowards (cashpoint)	Initial	¬hasDirection (cashpoint)	I
Runs to the cashpoint	Initial	GoingTowardsQuickly (cashpoint)	hasDirection (cashpoint) hasSpeedGreaterThan (10)	III
Does not run to the cashpoint	GoingTowardsQuickly (cashpoint)	Initial	¬hasDirection (cashpoint)	I
Accelerates and walks	GoingTowardsSlowly (cashpoint)	GoingTowards (cashpoint)	hasSpeedGreaterThan (3)	II
Decelerates and wanders	GoingTowards (cashpoint)	GoingTowardsSlowly (cashpoint)	SpeedLessThan (2)	III
Accelerates and runs	GoingTowards (cashpoint)	GoingTowardsQuickly (cashpoint)	hasSpeedGreaterThan (10)	III
Decelerates and walks	GoingTowardsQuickly (cashpoint)	GoingTowards (cashpoint)	SpeedLessThan (8)	II
Arrives to the cashpoint wandering	GoingTowardsSlowly (cashpoint)	CloseTo (cashpoint)	isCloseTo (25,cashpoint)	IV
Arrives to the cashpoint walking	GoingTowards (cashpoint)	CloseTo (cashpoint)	isCloseTo (25,cashpoint)	IV
Arrives to the cashpoint running	GoingTowardsQuickly (cashpoint)	CloseTo (cashpoint)	isCloseTo (25,cashpoint)	IV
Is not close to the cashpoint	CloseTo (cashpoint)	Initial	¬isCloseTo (30,cashpoint)	I

Table 26
Global state diagram, holdup at a cashpoint.

Action	Origin vertex	Destination vertex	Associated verbs	Alarm level
Vandalism action	Initial	Vandalism (cashpoint)	areMoreObjectsInState (GoingTowardsQuickly (cashpoint),0) areObjectsInState (CloseTo (cashpoint),0)	V
Nobody runs to cashpoint	Vandalism (cashpoint)	Initial	areObjectsInState (GoingTowardsQuickly (cashpoint),0)	I
Assault	Initial	Assault (cashpoint)	areMoreObjectsInState (GoingTowardsQuickly (cashpoint),0) areObjectsInState (CloseTo (cashpoint),1)	V
No assault	Assault (cashpoint)	Initial	areObjectsInState (GoingTowardsQuickly (cashpoint),0) areObjectsInState (CloseTo (cashpoint),0)	V
Possible holdup	Initial	Holdup (cashpoint)	areObjectsInState (GoingTowardsSlowly (cashpoint),1) areObjectsInState (CloseTo (cashpoint),1)	V
Nobody is at the cashpoint	Holdup (cashpoint)	Initial	areObjectsInState (CloseTo (cashpoint),0)	I

Table 27
Local state diagram, suspicious objects.

Action	Origin vertex	Destination vertex	Associated verbs	Alarm level
Stationary	Initial	StillObject	hasSpeedBetween (0,0) ¬isCloseTo (20, cashpoint) ¬isCloseTo (20, wastebasket)	I
Appears close to the cashpoint	Initial	SuspiciousObject	enterInScene () isCloseTo (20, cashpoint)	IV
Appears close to the wastebasket	Initial	SuspiciousObject	enterInScene () isCloseTo (20, wastebasket)	III
Starts	StillObject	MovingObject	hasSpeedGreaterThan (1)	I
Starts	SuspiciousObject	MovingObject	hasSpeedGreaterThan (1)	I
Is stopped for a long time	StillObject	SuspiciousObject	isTimeInState (Stopped, 30)	IV
Is much time suspicious	SuspiciousObject	VerySuspiciousObject	isTimeInState (SuspiciousObject, 30)	V

Table 28
Results of suspicious objects in scene "LeftBag".

Time	Object	State	Alarm
0:00:38	4	SuspiciousObject	IV
0:00:53	3	VerySuspiciousObject	V

Table 29
Results of suspicious objects in scene "LeftBag_PickedUp".

Time	Object	State	Alarm
0:00:35	4	SuspiciousObject	IV

5. Conclusions

In this paper, an approach to human activities detection in complex scenarios has been presented. The approach describes two levels in which activities should be considered: local activities are necessary to generalize the detection process; and global activities are used to detect behavior patterns that involve not only a single object, but also groups of objects (or even the whole set of objects) in the scene. Some parameters must be inferred from the objects in the scene, such as speed or direction. The system takes the initial segmentation to calculate these parameters. Next, a set of queries are proposed in order to specify simple behaviors (to detect movement, orientation and location of the objects), and complex behaviors (where one or several objects intervenes).

The results obtained so far are promising and we are currently engaged in performing tests after segmenting videos taken from different scenarios with our proper biologically motivated algorithms, namely accumulative computation and lateral inhibition (Delgado, López, & Fernández-Caballero, 2010; Moreno-Garcia, Rodriguez-Benitez, Fernández-Caballero, & López, 2010; Fernández-Caballero, López, Castillo, & Maldonado-Bascón, 2009).

Acknowledgements

This work was partially supported by the Spanish Ministerio de Economía y Competitividad / FEDER under project TIN2010-20845-C03-01, Spanish Ministerio de Industria, Energía y Turismo / FEDER under project TSI-020100-2010-261, and by the Spanish Junta de Comunidades de Castilla-La Mancha under projects PII2I09-0069-0994 and PEII09-0054-9581.

References

Amer, A., Dubois, E., & Mitiche, A. (2005). Rule-based real-time detection of context-independent events in video shots. *Real-Time Imaging, 11*(33), 244–256.

Arens, M., Gerber, R., & Nagel, H.-H. (2008). Conceptual representations between video signals and natural language descriptions. *Image and Vision Computing, 26*(1), 53–66.

Ayers, D., & Shah, M. (2001). Monitoring human behavior from video taken in an office environment. *Image and Vision Computing, 19*(12), 833–846.

Blunsden, S., & Fisher, R. B., 2009. Pre-fight detection – Classification of fighting situations using hierarchical AdaBoost. In *The fourth international conference on computer vision theory and applications* (Vol. 2, pp. 303–308).

Bui, H., Venkatesh, S., & West, G. (2002). Policy recognition in the abstract hidden Markov model. *Journal of Artificial Intelligence Research, 17*(45), 451–499.

Dee, H. M., & Hogg, D. C. (2009). Navigational strategies in behaviour modeling. *Artificial Intelligence, 173*(2), 329–342.

Delgado, A. E., López, M. T., & Fernández-Caballero, A. (2010). Real-time motion detection by lateral inhibition in accumulative computation. *Engineering Applications of Artificial Intelligence, 23*(1), 129–139.

Fernández, C., Baiget, P., Roca, F. X., & González, J. (2011). Determining the best suited semantic events for cognitive surveillance. *Expert Systems with Applications, 38*(4), 4068–4079.

Fernández-Caballero, A., Gómez, F. J., & López-Lápez, J. (2008). Road-traffic monitoring by knowledge-driven static and dynamic image analysis. *Expert Systems with Applications, 35*(3), 701–719.

Fernández-Caballero, A., López, M. T., Castillo, J. C., & Maldonado-Bascón, S. (2009). Real-time accumulative computation motion detectors. *Sensors, 9*(12), 10044–10065.

Fernández-Caballero, A., López, M. T., & Saiz-Valverde, S. (2008). Dynamic stereoscopic selective visual attention (DSSVA): Integrating motion and shape with depth in video segmentation. *Expert Systems with Applications, 34*(2), 1394–1402.

Gascueña, J. M., & Fernández-Caballero, A. (2011). Agent-oriented modeling and development of a person-following mobile robot. *Expert Systems with Applications, 38*(4), 4280–4290.

Gómez-Romero, J., Patricio, M. A., García, J., & Molina, J. M. (2011). Ontology-based context representation and reasoning for object tracking and scene interpretation in video. *Expert Systems with Applications, 38*(6), 7494–7510.

González, J., Rowe, D., Varona, J., & Roca, F. X. (2009). Understanding dynamic scenes based on human sequence evaluation. *Image and Vision Computing, 27*(10), 1433–1444.

Hamid, R., Maddi, S., Johnson, A., Bobick, A., Essa, I., & Isbell, C. (2009). A novel sequence representation for unsupervised analysis of human activities. *Artificial Intelligence, 173*(14), 1221–1244.

Haritaoglu, I., Harwood, D., & Davis, L. S. (2000). W4: Real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22*(8), 809–830.

Hongeng, S., Nevatia, R., & Bremond, F. (2004). Video-based event recognition: activity representation and probabilistic recognition methods. *Computer Vision and Image Understanding, 96*(2), 129–162.

Huang, P. C., Lee, S. S., Kuo, Y. H., & Lee, K. R. (2010). A flexible sequence alignment approach on pattern mining and matching for human activity recognition. *Expert Systems with Applications, 37*(1), 298–306.

Ivanov, Y. A., & Bobick, A. F. (2000). Recognition of visual activities and interactions by stochastic parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22*(8), 852–872.

Jan, T., Piccardi, M., Hintz, T., (2002). Detection of suspicious pedestrian behavior using modified probabilistic neural network. In *Proceedings of Image and Vision Computing* (pp. 237–241).

Ji, X., & Liu, H. (2009). Advances in view-invariant human motion analysis: a review. *IEEE Transactions on Systems, Man, and Cybernetics, Part C, 40*(1), 13–24.

Liao, L., Patterson, D., Fox, D., & Kautz, H. (2007). Learning and inferring transportation routines. *Artificial Intelligence, 171*(5-6), 311–331.

Liu, C., & Yuen, P. C. (2010). Human action recognition using boosted EigenActions. *Image and Vision Computing, 28*(5), 825–835.

López, M. T., Fernández-Caballero, A., Fernández, M. A., Mira, J., & Delgado, A. E. (2006). Motion features to enhance scene segmentation in active visual attention. *Pattern Recognition Letters, 27*(5), 469–478.

López, M. T., Fernández-Caballero, A., Fernández, M. A., Mira, J., & Delgado, A. E. (2007). Dynamic visual attention model in image sequences. *Image and Vision Computing, 25*(5), 597–613.

López-Valles, J. M., Fernández, M. A., & Fernández-Caballero, A. (2007). Stereovision depth analysis by two-dimensional motion charge memories. *Pattern Recognition Letters, 28*(1), 20–30.

Makris, D., & Ellis, T. (2005). Learning semantic scene models from observing activity in visual surveillance. *IEEE Transactions on Systems, Man and Cybernetics, 35*(3), 397–408.

Martínez-Cantos, J., Carmona, E., Fernández-Caballero, A., & López, M. T. (2008). Parametric improvement of lateral interaction in accumulative computation in motion-based segmentation. *Neurocomputing, 71*(4–6), 776–786.

Mira, J., Delgado, A. E., Fernández-Caballero, A., & Fernández, M. A. (2004). Knowledge modelling for the motion detection task: The lateral inhibition method. *Expert Systems with Applications, 7*(2), 169–185.

Moreno-Garcia, J., Rodriguez-Benitez, L., Fernández-Caballero, A., & López, M. T. (2010). Video sequence motion tracking by fuzzification techniques. *Applied Soft Computing, 10*(1), 318–331.

Neumann, B., & Möller, R. (2008). On scene interpretation with description logics. *Image and Vision Computing, 26*(1), 82–101.

Oliver, N. M., Rosario, B., & Pentland, A. P. (2000). A Bayesian computer system for modeling human interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22*(8), 831–843.

Stauffer, C., & Grimson, E. (2000). Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22*(8), 747–757.

Xiang, T., & Gong, S. (2006). Beyond tracking: Modeling activity and understanding behaviour. *International Journal of Computer Vision, 67*(1), 21–51.