Interactive Video Surveillance for Perimeter Control

Javier Ortells, Henry Anaya-Sánchez, Raúl Martín-Félez,
and Ramón A. Mollineda

Abstract. This chapter presents an interactive video-surveillance solution for assisting human operators in the control of movements across a multi-region scenario (perimeter control). It has been conceived as a multi-camera system to detect anomalous trajectory events, such as entering or leaving a region or changing the walking speed, by means of a dynamic collection of decision rules. They relate spatio-temporal patterns and event categories (anomalous, unknown, normal), and are used to assess and classify trajectory events. The interactive paradigm has been adopted as a natural framework to progressively learn and update rules, particularly at early stages of the system operation. The approach of continuously improving system knowledge from user feedback conducts to adaptive, reliable and increasingly automatic systems in a relatively short period of time.

1 Introduction

Video surveillance has become part of our everyday lives. The interest on remote visual systems for security purposes has grown significantly in the last years, particularly for those public spaces where a great number of people pass. Some examples are supermarkets, airports, underground stations, stadiums, shopping centers, etc. Traditionally, these systems have required a huge amount of human supervision both in real-time and in a posterior video analysis to state the truth. However, with the current proliferation of cameras, this exhaustive search has turned into unfeasible.

A major breakthrough of video-surveillance systems is coming about with the development of video analytic techniques. They can be roughly defined as autonomous understanding of events occurring in a scene monitored by multiple video cameras [16]. However, most present-day video-surveillance systems are far from being

Javier Ortells · Henry Anaya-Sánchez · Raúl Martín-Félez · Ramón A. Mollineda
Institute of New Imaging Technologies, Universitat Jaume I of Castelló
Av. Sos Baynat s/n, 12071, Castelló de la Plana, Spain
e-mail: {jortells,henry.anaya,martinr,mollined}@uji.es

fully autonomous, particularly when video analytics should deal with complex and crowded scenes. Furthermore, autonomy is not always an end purpose for some task solutions, which have an intrinsic interactive nature. While some of them can be substantially benefited from human feedback, such as speech transcription and machine translation [18], others require the human involvement in a critical decision making process, for example, in computer-assisted medical image diagnosis.

The interactive paradigm in computer vision applications allows users to progressively enhance priors and constraints by adding their knowledge within an operational loop of the system. This approach has several benefits on learning. Firstly, it exploits human know-how in a natural way (through feedback) for model refinement. Secondly, it leads to solutions that better fit the user demands. Finally, it favors a dynamic adjustment of models when the user needs or the context changes.

A typical video-surveillance application is an intelligent system designed for monitoring one or more regions of a scene, in order to detect and track particular objects or situations according to predefined safety rules. When this kind of system is deployed in simple scenarios, for example, the garden of a house, it is easy to customize the behavior of the system through a few basic and steady rules [8]. However, in more complex and changing environments such as large commercial areas, big factories or public squares, it could be unrealistic to try to foretell every particular situation that requires a specific management. Thus, a prior and hard-to-change collection of safety rules does not appear to be the most suitable strategy. Under such shifting conditions, an interactive framework for supporting the on-line management of unseen events can be a natural alternative to keep a set of meaningful rules updated.

This manuscript proposes an interactive solution to the general problem previously described. A video-surveillance application for interactively controlling movements across a multi-region scenario (perimeter control) is here presented. It is a two-camera prototype to detect anomalous actions on a region model defined in a real-world scene. The analysis of trajectory data is performed by using a collection of dynamic spatio-temporal rules. A suitable graphical user interface allows on-the-fly rule management via emerging windows when a detected event is unknown (rule definition) or when it is classified as anomalous (alarm and event refinement). This feedback mechanism has been adopted for progressively building and keeping updated an effective series of rules.

This chapter is organized as follows. Section 2 summarizes the related state-of-the-art works. The interactive learning approach is explained in Sect. 3. A description of the implemented system is given in Sect. 4, including scope, system architecture, main software modules and resources needed. Finally, Sect. 5 gathers the conclusions of the chapter.

2 Related Work

One of the main applied research areas of video analytic methods focuses on visual event detection, tracking and classification, looking for anomalies in order to alert human operators. A broad range of video-surveillance applications can be envisioned from this general operational context, such as access and perimeter control, human identification at a distance, detection of anomalous behaviors of people or vehicles, etc. An inspiring survey about methods and applications for video surveillance of people and vehicle is given in [10]. It presents a general framework of visual surveillance in dynamic scenes from multiple cameras, and discusses the principles and methods involved in the main parts of the framework. That paper also sheds light upon a number of key research problems and directions, providing a consistent basis for the forthcoming developments. However, although interactive functions are suggested as a key direction for surveillance applications, no discussion is provided about the importance or convenience of using human feedback for managing critical decisions with higher confidence in some tasks.

A complementary study of the state-of-the-art of visual surveillance systems was provided afterwards in [17]. This work focuses in distributed automated surveillance systems, and it reviews a number of real commercial applications. One of them, DETER [15], consists of two main modules, one for detection, recognition and tracking of objects, and a second one for threat assessment and alarm management. The lack of a feedback loop in the alarm module to improve its performance is highlighted in [17]. In spite of this observation, no other claim toward human interaction in surveillance systems appears in this survey.

Focusing on specific surveillance solutions, it is compulsory to mention two early prominent proposals [5, 9]. As a part of the Video Surveillance and Monitoring (VSAM) project (1997-1999), an ambitious initiative of the DARPA, the Robotics Institute at Carnegie Mellon University (CMU) and the Sarnoff Corporation developed a system for autonomous video surveillance and monitoring [5]. This system included robust methods for detecting and tracking moving objects by multiple cooperative video sensors, and for classifying these objects into semantic categories such as human, human group, car and truck. Besides, people activity was also classified as walking or running. The system GUI had a module for managing Regions Of Interest (ROIs), that allowed the interactive creation of a polygonal ROI from a collection of boundary points. The user could also link object types (e.g. human, vehicles) to ROIs, providing the system with a mechanism for triggering events like "enter", "pass through", "stop in", etc. The detected activities from all sensors were transmitted to a central unit, where they were displayed and analyzed.

In [9], a real-time visual surveillance system (W^4) for detecting and tracking people in an outdoor environment is described. From gray-scale and infrared video imagery, W^4 is able to locate and track people and their parts (head, hands, feet, torso) using models of people's appearance. W^4 can also detect and segment objects that are being carried by people, so interactions among people and objects can be described, such as depositing, removing and exchanging objects. However, despite of the relevance of the above two cited papers, none of them documents any human feedback function for model improvement.

A methodological work is presented in [7], where a number of critical issues related to video-surveillance requirements are addressed. For instance, it describes several low-level image and video processing techniques such as change detection for fixed and mobile cameras, background updating, multi-camera view registration, etc.

This paper provides valuable guidelines and resources to new researchers in video surveillance, to design low-level vision modules. High-level video analysis functions for scene understanding are beyond the scope of this paper.

Recent progress in embedded sensor technology is encouraging the development of distributed sensor networks. In particular, smart cameras comprise sensing, processing and communication functions, all built in within the same device. Two papers about distributed networks of embedded smart cameras are [3, 4]. In [3], a distributed network of smart cameras for surveillance purposes is introduced, along with the low-level video processing algorithms developed for particular nodes. Some of the low-level routines included are segmentation, labeling, tracking and classification of detected objects. On the other hand, in [4], a traffic surveillance application is proposed. A scalable smart camera hardware architecture is designed as an open technology to develop distributed intelligent video-surveillance systems. Each smart camera is assigned to a particular region, and some motion vectors are defined to represent the spatial relationship among the cameras. These vectors allow to check whether an object is moving in the correct direction. A multi-camera object-tracking application is implemented, where a tracking agent migrates from one camera to another in no more than one second. This parameter imposes conditions on both maximum vehicle speed and minimum camera distance.

In the literature reviewed, taking advantage of the user's feedback for a continuous improvement of the models is not a target. We believe that this practice can help to build adaptive, reliable and efficient systems in a short period of time, which is especially useful when critical decisions must be made. For example, in the case of distributed smart camera networks, the inter-node spatial relationships could be modeled by an interactive approach, in which the network would propose spatial hypotheses to a human operator for their refinement and validation. These validated hypotheses (knowledge) could help to infer new and more accurate hypotheses, and so on. This is the cornerstone of interactive learning: each piece of human feedback information must contribute to increase the system robustness and, usually, to decrease the human-machine interaction effort.

3 Interactive Learning Strategy

As discussed previously, the singularity of the proposed visual surveillance solution is the interactive strategy to learn and update safety rules as context conditions change. Object trajectories are monitored, and those events relevant to the system end-purpose are detected and classified by a collection of rules, which can be interactively defined or modified either on-line or off-line.

From a conceptual perspective, the implemented prototype embodies the following two main interactive learning requirements:

- System knowledge increase, by allowing on-the-fly creation of new rules to model unseen trajectory events.
- System knowledge adapting, by allowing on-the-fly updating of existing rules, when they are triggered, to model new user needs or a changing environment.

A simple rule-based approach has been implemented to manage these requirements. It is able to encapsulate human knowledge through a collection of “if then” declarative statements, which is an easy way for human beings to understand the know-how description. Considering our goals, this approach has two main strengths: 1) it makes quick decisions in a repeatable form (deterministic procedure), and 2) it admits an agile adaptation when a new piece of knowledge is added or an existing one is updated. These useful qualities are due to the absence of hard constraints on the order of evaluating the rules. Generally, a conflict-resolution policy is needed to decide which rule to fire if some of them match with a given problem state. The simplest approach is possibly to define a rule order for choosing the first triggered rule. Thus, when a new rule is added, an ordered insertion is enough to adapt the collection of rules for dealing with an extended problem scope.

An alternative would have been a rule-based decision tree, a structure that can be induced from a set of rules [2]. Its main merit is that it can potentially organize rules in a concise and efficient way to take the best decision readily. However, when the collection of rules changes, the decision tree must be completely re-built in order to keep it consistent. The complexity of this process depends on the number of rules, the attributes involved in rule premises, the intersection level between rules, and the heuristic criteria used to create the tree. In the case of an interactive solution that starts with no knowledge of the task to be solved, which is also expected to change frequently, a hard-to-adapt model is not certainly the most suitable choice.

The video-surveillance system here introduced should monitor, describe and classify (as anomalous, unseen or normal) the trajectory points of moving object across a scene view. Given a trajectory point description, a rule-based reasoning process tests the condition of each decision rule against the trajectory point data (known fact), regarding a predetermined evaluation order. This process searches for a first matching rule to produce an output. On success, the trajectory event is classified as anomalous or normal, according to the action associated to the triggered rule. Otherwise (when no rule is applicable), it is considered as unknown.

The interactive strategy adopted allows a progressive learning and updating of decision rules, particularly at early stages of the system operation. This approach leads to the following knowledge life cycle:

Starting Point: The system starts with no knowledge about the surveillance task to be solved. That is, there are no rules at the beginning, so every trajectory event is initially classified as unseen.

Learning a New Rule: When an unseen trajectory event is detected, an emerging GUI asks the user to infer and define a decision rule from this particular event[1]. The new rule is then inserted in an orderly manner into the rule set, in such a way that rules which involve more specific conditions are prioritized. To this end, an algorithm computes the generality/specificity relation between the new rule and each existing rule, using two criteria (in the order they appear): 1) the higher the number of attributes in a rule condition is, the more specific the rule is, and 2)

[1] It is also possible to create rules using the standard GUI, out of the main system operation loop.

when the conditions of two rules are expressed in terms of the same attributes, an analysis about their intersection is performed. Three possible results are considered from this analysis: a) if there is no intersection between their attribute values, both rules are considered as not related; b) if the attribute values that satisfy one rule are subsets of the corresponding attribute values of the other rule, the former is considered more specific than the latter (the first one can be deemed as an exception of the second one); and c) when other types of intersections are found out, the system warns the user about a possible contradiction between both rules, and no insertion is carried out. Note that the binary relation defined, "more specific than", is antisymmetric and transitive, what avoids critical ambiguities in deciding where to insert. When no generality/specificity relation exists between two rules, their order of appearance determines the rule order.

Updating a Rule: When an alarm rule is triggered, a related emerging GUI allows the user to update the rule to fit a new requirement of the task. For example, the user might modify its parameters or deactive the rule, *i.e.* to transform the alarm into a normality rule.

This procedure of continuously improving system knowledge from user feedback should conduct to an adaptive, reliable and increasingly automatic solution.

4 Prototype Description

In this section, we provide details about the functional scope, system architecture and the main application modules of the proposed video-surveillance solution.

4.1 Functional Scope

The implemented prototype has been devised as a multi-camera system for perimeter control over a region map defined for a real-world scene. A multi-camera object tracking process allows trajectory monitoring and assessment through a collection of rules, which can be interactively defined either on-line or off-line.

Apart from the interactive learning capacities (see Sect. 3), the prototype provides a number of user functions for camera configuration, scene region definition, region projection over distinct scene views, rule definition, alarm management, real-time display of the monitored scene views, video recording for off-line analysis, among others. These end-user functions are supported by some other operations such as multi-camera acquisition, view-dependent object detection, view-dependent object tracking, object matching between different camera views, trajectory-based event detection, rule-based event classification and real-time information display.

Next sections provide details of this operational context.

4.2 System Architecture

Figure 1 illustrates the high-level system architecture. Five main modules can be identified in the diagram:

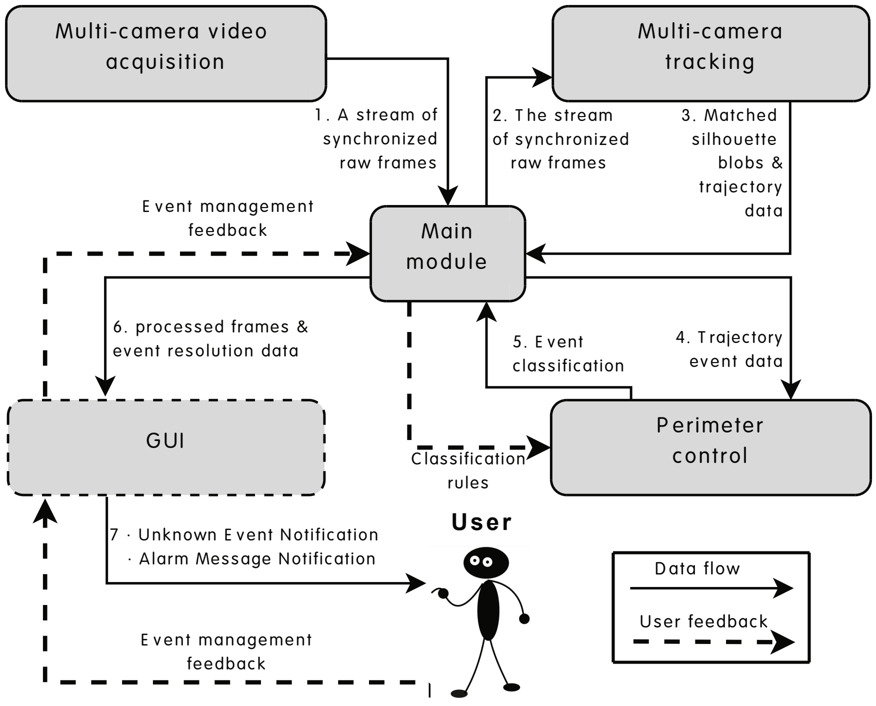
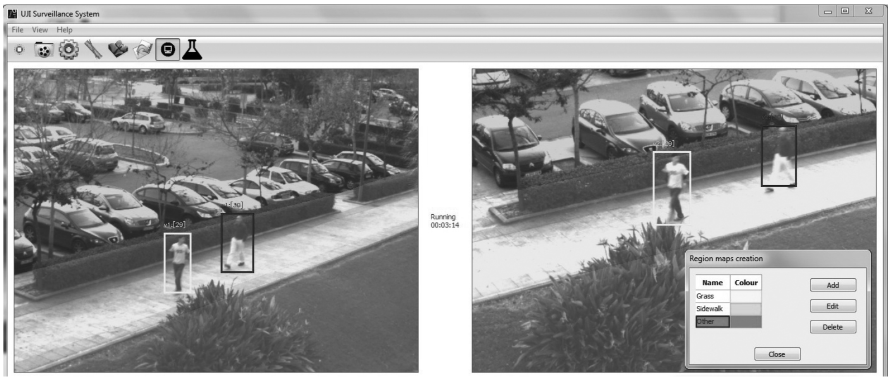

Multi-camera video acquisition
Multi-camera tracking
1. A stream of synchronized raw frames
2. The stream of synchronized raw frames
3. Matched silhouette blobs & trajectory data
Event management feedback
Main module
6. processed frames & event resolution data
5. Event classification
4. Trajectory event data
GUI
Perimeter control
Classification rules
User
7 · Unknown Event Notification
· Alarm Message Notification
Data flow
User feedback
Event management feedback

Fig. 1 High-level architecture of the video-surveillance prototype for perimeter control

Multi-camera Video Acquisition: It consists of multi-port firewire cards where one or more video cameras are plugged into, low-level software drivers to gain access to the cards, and high-level software components to make easier the interaction between the system and the cameras. The output of this module is a stream of synchronized raw frames for each camera, which feeds the tracking module.

Multi-camera Tracking: The inputs to this module are the streams of synchronized raw frames coming from different cameras and provided by the Video acquisition module. Object tracking is separately performed on each stream by a particular tracking thread. Then, a collection of synchronized tracking results (blobs) from all the tracking threads is given to the matching algorithm. It assesses all possible correspondences between blobs tracked on different cameras, looking for multiple views of the same object/subject. When two blobs are matched, they are tagged with a same code, thus uniquely identifying the corresponding real-world object/subject in the system. Figure 2 illustrates this process. The two blobs of a same subject are linked by a common colour of their bounding box edges, and also by an identical numeric label located on the upper edge of the bounding box. The output of this module is a collection of matched blobs, which model the 1 : n relationships between scene real objects and their camera projections.



Name	Colour
Grass	▨
Sidewalk	▨
Other	▨

Fig. 2 An example of two-camera object tracking. The two silhouettes of a same subject share a common colour in their bounding box edges and an identical numeric label.

Perimeter Control: The input to this module is trajectory data of tracked objects. Given a multi-region model defined on a real-world scene and projected onto the different camera views (view region maps), a collection of rules classify trajectory points obtained from tracking objects across the regions. The use of multiple cameras covering different viewpoints of a scene should lead to a more accurate detection of anomalous movements.

GUI: In human-machine interaction, the graphical user interface (GUI) is a crucial part of the system. On the one hand, the GUI shows a real-time view of each camera with the bounding boxes of the detected objects overprinted on them. On the other hand, it allows the user to provide feedback to the system for performance improvement, what mostly occurs as a part of a work session. An intuitive, simple and easy-to-use GUI has been designed and implemented. In Sect. 4.2.3, the most important functions provided by the GUI are described.

Main Module: This module coordinates all data flows across the system. As more than one video stream must be concurrently managed, a multi-threaded design is set to allow a synchronized video acquisition, tracking and analysis. It also keeps the main data structures updated, including the collection of decision rules (see Sect. 3) and the estimated trajectory data obtained from blob positions such as the movement speed, crossed regions, and so on.

In the following subsections, the three most relevant architecture modules to the end-purpose of the system are described in detail.

4.2.1 Multi-camera Tracking

As stated above, the term *multi-camera tracking* is used to refer the process of assigning a unique label to multiple camera projections of a same real object/subject. In this paper, this function consists of two main steps: view-dependent foreground segmentation, and a matching algorithm to find the proper correspondences between blobs detected on different cameras. These two steps are explained below.

Foreground Segmentation – In the proposed system, background modeling is combined with post-processing techniques to retrieve the foreground from each video frame. The background modeling relies on the adaptive approach introduced in [11]. The aim is to make the system robust enough to deal with dynamic changes that can appear in the scene views (e.g. global illumination changes produced from day-night transitions or long-term background updates corresponding to events such as parking a car in front of a building).

In the background model of a scene view, each color pixel i is represented by three non-stationary Gaussian distributions $\left\{N_c^{(i)}\left(\mu_c^{(i)}(t), \sigma_c^{(i)}(t)\right)\right\}$, $c \in \{R, G, B\}$, where each distribution models a RGB-component of the pixel at time t. The methodology for segmenting the foreground proceeds as follows.

Initially, the learning of the background model of the scene view comprises a sequence of video frames given as training. Then, to retrieve the foreground of an incoming frame, its pixels are classified into either background or foreground. A pixel i is classified into the foreground class if at least one of its three measured RGB values are out of a confidence region of their corresponding Gaussian distributions. Otherwise, the pixel is considered as background, and its Gaussian representation is updated using the equations (1) and (2):

$$\mu_c^{(i)}(t+1) = \alpha\mu_c^{(i)}(t) + (1-\alpha)z_c^{(i)}(t) \tag{1}$$

$$\begin{aligned}\left(\sigma_c^{(i)}(t+1)\right)^2 = \alpha\left((\sigma_c^{(i)}(t))^2 + \left(\mu_c^{(i)}(t+1) - \mu_c^{(i)}(t)\right)^2\right) + \\ + (1-\alpha)\left(z_c^{(i)}(t) - \mu_c^{(i)}(t+1)\right)^2\end{aligned} \tag{2}$$

where $z_c^{(i)}(t)$ is the observed value for the RGB-component c of pixel i in the frame, and the parameter α is the adaptation rate $(0 < \alpha < 1)$. Finally, the foreground is represented by a binary pixel map, according to the classification of the pixels.

Additionally, post-processing techniques are applied on the binary pixel map since it frequently contains noise due to motion of small background objects or shadows. This post-process includes morphological operations and shadow filtering.

Matching Algorithm – In this second step, the multi-camera tracking process is completed by combining the single-camera tracking corresponding to each scene view with a matching procedure operating on 2D-scene regions.

Broadly, the single-camera tracking is based on both 1) an appearance model for each individual in the scene view and 2) a blob overlapping criterion to estimate the regions occupied by a tracked individual through the video stream frames.

In the matching procedure 2D-scene regions are modeled as Gaussian distributions, each one being parameterized by both a mean vector and a covariance matrix. That is, a given region is represented by the maximum likelihood estimated Gaussian distribution that describes the locations of its pixels in the view, and the distance between regions is calculated in terms of the Bhattacharyya distance.

From these settings, the matching procedure is build upon a stochastic model of pairs of regions that represents the correspondence between 2D-regions in the different views according to the real-world 3D-region they represent. This correspondence model between regions in the different scene views is represented by a finite set of of categories $\mathcal{C} = \{C_1, \dots, C_k\}$; where each category C_i $(1 \leq i \leq k)$ is centered at a pair of 2D-regions $\langle u_i, v_i \rangle$ (in different scene views) with a high likelihood of representing the same real-world region. Each category C_i is also provided with deviation parameters σ_{i_1} and σ_{i_2}.

In this way, given a pair of 2D-regions $\langle u, v \rangle$ in different views, the matching procedure estimates a measure of the confidence this pair of regions is generated from the model $\mathcal{C}$ as showed in Eq. 3:

$$\mathrm{p}\left(\langle u,v \rangle | \mathcal{C}\right) = \frac{1}{k} \sum_{i=1}^{k} \frac{1}{\sigma_{i_1} \sigma_{i_2}} \mathcal{K}\left(\frac{d(u,u_i)}{\sigma_{i_1}}, \frac{d(v,v_i)}{\sigma_{i_2}}\right) \tag{3}$$

where d represents the distance function between regions and $\mathcal{K}$ is the Gaussian of the Eq. 4:

$$\mathcal{K}\left(x,y\right) = e^{-\frac{1}{2}(x^2+y^2)} \tag{4}$$

From the estimated value $\mathrm{p}\left(\langle u,v \rangle | \mathcal{C}\right)$, the matching procedure makes a decision on whether the regions $\langle u, v \rangle$ represents the same real-world scene or not by relying on an empirical threshold. In the system, the set of categories $\mathcal{C}$ is learned from a synchronized training sequence of video frames from the scene views.

4.2.2 Perimeter Control

The aim of the Perimeter control module is to evaluate and classify trajectory events using a collection of rules. This section describes how this module works based on two important data structures: the *view region map* and the *rule*.

The term *view region map* is used to refer the 2D projection of a 3D region model onto a particular camera view. It provides spatial information to the trajectory evaluation process. Each view region map is coded by a logic matrix whose size matches the view resolution. Thereby each camera pixel corresponds to a matrix cell where the identifier of the pixel region is stored. This data structure makes easy to check which region a pixel belongs to.

On the other hand, a *rule* can be formally defined as a condition that can be formulated in terms of one or more of the following context parameters:

Time Range: This parameter can adopt three different formats, depending of the nature of the time period used:

i If the rule is intended to control events within the same time period every day, the starting and finishing times are required.

ii If the rule is intended to control weekly events within a time period between two week days (e.g. from every Friday at 22:00 h. to next Monday at 6:00 h.), these two week days and the starting and finishing times must be provided.

iii If the rule is intended to control events within a time period between two calendar days (e.g. from July 31st at 22:00 h. to September 1st at 6:00 h.), these two calendar days and the starting and finishing times must be provided.

Current Region: It identifies the region where the object is physically located.

Previous Region: This parameter identifies the region, if any, on which the object was before entering into the current region.

Movement Speed: This parameter sets a maximum speed limit, in such a way that higher speeds are considered anomalous.

In addition, the rule must be defined/labeled as either normality or alarm, which will determine the *action* taken by the system. The fulfillment of a normality rule does not cause any visible result. On the contrary, when an alarm rule is triggered, a warning message alerts the operator, who could also interact with the system to modify the involved rule.

4.2.3 GUI Capabilities

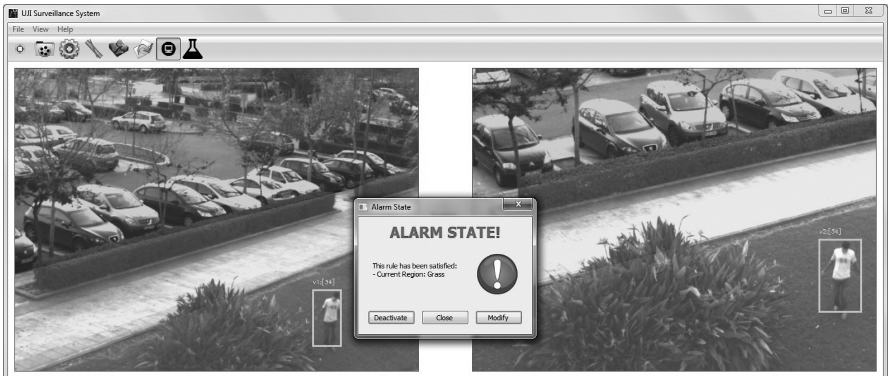
This section provides an overview of the main functional capabilities of the GUI of the implemented video-surveillance prototype.

Camera Configuration: At the beginning of a work session, the user can choose which cameras will be used among those plugged into the multi-port firewire cards. Besides, the user can configure the cameras by choosing values for some operating parameters such as the resolution and the sampling frequency.

Region Definition and Region Map Outlining: Regions are defined in a real-world scene, but they are manually drawn (projected) over each camera view. This user function supports region definition by a meaningful name and a specific color, and the creation of view region maps. A drawing tool to outline piecewise linear closed contours on fixed images was implemented. It is used to draw the view region map on some key video frame with suitable background information of each camera view. The tool also supports the visual superposition of a region map on the video frames as a semi-transparent layer when the system is running.

Interactive Event Management:

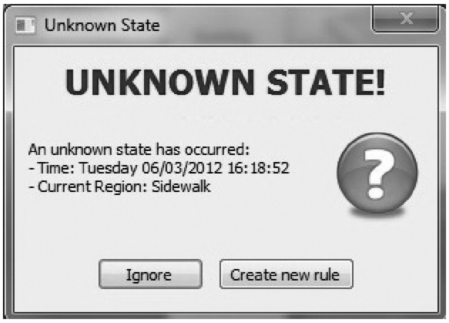
- *Alarm notification*. When an alarm rule is fulfilled, a pop-up window is triggered (see Fig. 3). It allows the user to deactivate the rule, *i.e.* to transform the alarm into a normality rule, or to modify its parameters.
- *On-line rule definition*. When a trajectory event is classified as unknown, a new window emerges (see Fig. 4) describing the current state and allowing the user to transform it into an alarm or a normality rule by the GUI shown in Fig. 5. Some rule parameters are filled in with the collected information. It is also possible to ignore an unknown event when normality and alarm criteria are not clear (see the Ignore button in Fig. 4). When this decision is made, a short-term normality rule for a parameterized period of time is automatically



ALARM STATE!

This rule has been satisfied:
- Current Region: Grass

Fig. 3 An alarm rule is satisfied when a subject enters to the grass area



UNKNOWN STATE!

An unknown state has occurred:
- Time: Tuesday 06/03/2012 16:18:52
- Current Region: Sidewalk

Fig. 4 An example of an unknown state detection that requires user interaction

created. That means, for a limited amount of time, those situations similar to the one ignored will be considered as normal. When this time ends, the provisional rule is removed and similar events will be unknown again.

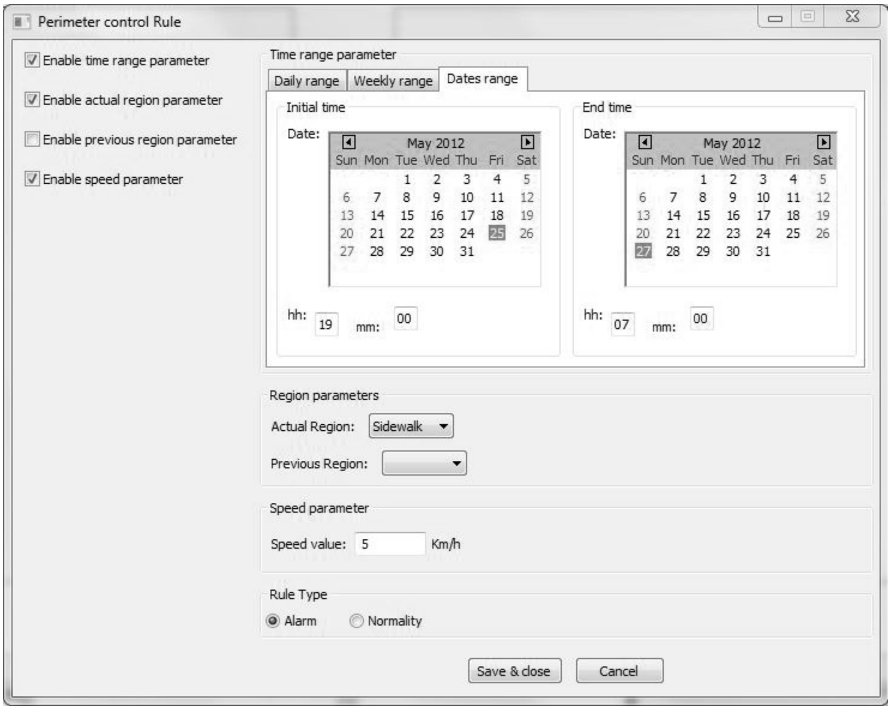
- *Off-line rule definition*. A surveillance rule can also be defined at any time from the application menu. Figure 5 illustrates an example of the definition of a new alarm rule using time, region and speed parameters.

4.3 Hardware and Software Resources

The implemented prototype has been the result of using a number of specific hardware and software resources. This section summarizes the most relevant ones.

Hardware Resources:

- Two firewire 'B' AVT Stingray F-080^{TM} cameras. The prototype was also tested using other AVT cameras such as Guppy F-036^{TM} (Firewire 'A').
- A firewire 'B' card with two independent buses to connect Stingray cameras. The firewire 'A' cameras were plugged into two other suitable cards.



Perimeter control Rule

[✓] Enable time range parameter

[✓] Enable actual region parameter

[ ] Enable previous region parameter

[✓] Enable speed parameter

Time range parameter

Daily range | Weekly range | Dates range

Initial time

Date:

May 2012

Sun	Mon	Tue	Wed	Thu	Fri	Sat
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

hh: 19 mm: 00

End time

Date:

May 2012

Sun	Mon	Tue	Wed	Thu	Fri	Sat
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

hh: 07 mm: 00

Region parameters

Actual Region: Sidewalk

Previous Region:

Speed parameter

Speed value: 5 Km/h

Rule Type

[✓ Alarm] Normality

Save & close | Cancel

Fig. 5 Example of defining a new alarm rule using time, region and speed parameters

- A Dell XPS 730X^{TM} desktop computer to develop and test the prototype. Its main features are: Intel Core^{TM} i7 920 (2.67GHz) CPU, nVidia GeForce GTX 285^{TM} GPU, 6GB RAM, Windows 7 Professional^{TM} 64 bits.

Software Resources:

- Microsoft Visual Studio 9^{TM} [12] was used as development environment.
- AVT Universal Package, which is a programming API to easily operate with AVT (Allied Vision Technologies^{TM} [1]) cameras.
- wxWidgets cross-platform GUI library [19], which is a C++ library that supports the development of graphic interfaces. It also offers useful methods and data structures for general purpose programming.
- OpenCV library [13] provides some image processing functions that were used with the video frames.
- CvBlobslib library [6] was used to perform some tracking and perimeter control tasks. Some pieces of its source code were modified to fit our purposes.

5 Conclusions and Future Work

This work proposes an interactive video-surveillance solution for assisting human operators in the control of movements on a multi-region scenario (perimeter control). Trajectory monitoring is performed by a multi-camera tracking algorithm, while trajectory event assessment is supported by a dynamic collection of spatio-temporal rules. An interactive approach has been implemented for allowing human operators to progressively extend, adapt and refine the system knowledge (rules) by providing feedback within an operational loop of the system. When an anomalous or an unknown event is detected, suitable emerging user interfaces allow on-the-fly rule management for defining or updating rules. This kind of interaction leads to increasingly automatic operating modes, because the more feedback the user provides, the less future interactions are expected to be demanded by the system and more reliable decisions can be autonomously made.

Future developments could involve the integration of non-intrusive biometric functions such as those based on face, gait or even on a combination of them under a multi-modal biometric system. Other related applications might be re-identification of people over different cameras do not sharing the same scene, detection of abandoned luggage, and so on. Furthermore, due to the system capability of storing trajectory data, other parameters such as the covered path distance, the scene entry and exit points, and the exposure time can be straightforwardly estimated, which might be useful for instance in a control access application.

Acknowledgements. The work documented in this paper has been partially funded by the five-year Multimodal Interaction in Pattern Recognition and Computer Vision (MIPRCV) research project (2007-2012) with code CSD2007-00018. Other projects that have also contributed to support this work are TIN2009-14205-C04-04 from the Spanish Ministry of Innovation and Science, P1-1B2009-04 from Fundació Bancaixa, and PREDOC/2008/04 grant from Universitat Jaume I.

References

1. Allied Vision Technologies^{TM} website, `http://www.alliedvisiontec.com`
2. Abdelhalim, A., Traore, I.: Converting Declarative Rules into Decision Trees. In: Proc. of the World Congress on Engineering and Computer Science (WCECS), San Francisco, USA (2009)
3. Benet, G., Sim, J.E., Andreu-Garcia, G., Rosell, J., Sanchez, J.: Embedded low-level video processing for surveillance purposes. In: Proc. of 3rd Conference on Human System Interactions (HSI), Rzeszow, Poland, pp. 779–786 (2010)
4. Bramberger, M., Doblander, A., Maier, A., Rinner, B.: Distributed embedded smart cameras for surveillance applications. Computer 39(2), 68–75 (2006)
5. Collins, R.T., Lipton, A.J., Kanade, T., Fujiyoshi, H., Duggins, D., Tsin, Y., Tolliver, D., Enomoto, N., Hasegawa, O., Burt, P., Wixson, L.: A System for Video Surveillance and Monitoring. Technical report CMU-RI-TR-00-12, Carnegie Mellon University, Pittsburgh, and The Sarnoff Corporation, Princeton, NJ (2000)
6. cvBlobslib library website, `http://opencv.willowgarage.com/wiki/cvBlobsLib`
7. Foresti, G.L., Micheloni, C., Snidaro, L., Remagnino, P., Ellis, T.: Active video-based surveillance system: the low-level image and video processing techniques needed for implementation. Signal Processing Magazine 22(2), 25–37 (2005)

8. Frejlichowski, D., Forczmański, P., Nowosielski, A., Gościewska, K., Hofman, R.: SmartMonitor: An Approach to Simple, Intelligent and Affordable Visual Surveillance System. In: Bolc, L., Tadeusiewicz, R., Chmielewski, L.J., Wojciechowski, K. (eds.) ICCVG 2012. LNCS, vol. 7594, pp. 726–734. Springer, Heidelberg (2012)
9. Haritaoglu, I., Harwood, D., Davis, L.S.: W^4: Real-Time Surveillance of People and Their Activities. IEEE Trans. on Pattern Analysis and Machine Intelligence 22(8), 809–830 (2000)
10. Hu, W., Tan, T., Wang, L., Maybank, S.: A survey on visual surveillance of object motion and behaviors. IEEE Trans. on Systems, Man, and Cybernetics - Part C 34(3), 334–352 (2004)
11. McKenna, S.J., Jabri, S., Duric, Z., Rosenfeld, A., Wechsler, H.: Tracking Groups of People. Computer Vision and Image Understanding 80(1), 42–56 (2000)
12. Microsoft Visual Studio 9$^{\text{TM}}$ (2008), `http://www.microsoft.com/visualstudio/en-gb/products/2008-editions`
13. OpenCV (Open Source Computer Vision) library website, `http://opencv.willowgarage.com`
14. Ortells, J., Anaya-Sánchez, H., Mollineda, R.A.: A demo of an interactive video-surveillance prototype for perimeter control (2012), `http://miprcv.iti.upv.es/index.php?option=com%5fcontent&task=view&id=220&Itemid=205`
15. Paulidis, I., Morellas, V.: Two examples of indoor and outdoor surveillance systems. In: Remagnino, P., Jones, G.A., Paragios, N., Regazzoni, C.S. (eds.) Video-based Surveillance Systems, pp. 39–51. Kluwer Academic Publishers, Boston (2002)
16. Regazzoni, C.S., Cavallaro, A., Wu, Y., Konrad, J., Hampapur, A.: Video Analytics for Surveillance: Theory and Practice. IEEE Signal Processing Magazine 27(5), 16–17 (2010)
17. Valera, M., Velastin, S.A.: Intelligent distributed surveillance systems: a review. IEE Proc. - Vision, Image and Signal Processing 152(2), 192–204 (2005)
18. Vidal, E., Rodríguez, L., Casacuberta, F., García-Varea, I.: Interactive Pattern Recognition. In: Popescu-Belis, A., Renals, S., Bourlard, H. (eds.) MLMI 2007. LNCS, vol. 4892, pp. 60–71. Springer, Heidelberg (2008)
19. wxWidgets cross-platform GUI library website, `http://www.wxwidgets.org`