

# Event Detection and Analysis from Video Streams

Gérard Medioni, *Senior Member, IEEE*, Isaac Cohen, *Member, IEEE*,  
François Brémond, *Student Member, IEEE*, and  
Ramakant Nevatia, *Fellow, IEEE*

**Abstract**—We present a system which takes as input a video stream obtained from an airborne moving platform and produces an analysis of the behavior of the moving objects in the scene. To achieve this functionality, our system relies on two modular blocks. The first one detects and tracks moving regions in the sequence. It uses a set of features at multiple scales to stabilize the image sequence, that is, to compensate for the motion of the observer, then extracts regions with residual motion and uses an attribute graph representation to infer their trajectories. The second module takes as input these trajectories, together with user-provided information in the form of geospatial context and goal context to instantiate likely scenarios. We present details of the system, together with results on a number of real video sequences and also provide a quantitative analysis of the results.

**Index Terms**—Detection and tracking of moving objects, egomotion estimation, affine stabilization, mosaics, graph representation of objects trajectories, event analysis, geospatial and mission contexts, scenario recognition, finite automaton.

## 1 INTRODUCTION

RECENT development in video acquisition hardware has made possible the acquisition of good quality video streams and increased the scientist's interest in developing video surveillance systems. The development of such systems present several difficulties and one of the most challenging is behavior analysis since it requires the inference of a semantic description of the features (moving regions, trajectories, etc.) extracted from the video stream. The ambitious goal here is to automatically process video streams, acquired in specific situations, in order to characterize the actions taking place and to infer whether they present a threat that should be signaled to a human operator.

Recent efforts have proposed partial solutions for specific video streams, namely, fixed [24], [19], [17], [20], and Pan Tilt Zoom [33] cameras. Systems based on fixed cameras were primarily used for monitoring road traffic scenes [24], where the camera observes the activity of rigid objects in a structured domain. Bobick et al. [2] have proposed a coupled Hidden Markov Model and stochastic grammars for recognizing activities (of rigid and nonrigid objects) and identifying different behavior based on contextual information from a video stream acquired by a static camera. Kanade et al. [32] have developed a multisensor

video surveillance system integrating PTZ and moving cameras. This impressive system has focused so far on the integration of several sensors and on the detection and tracking of moving objects rather than the description of their behavior. The development of such integrated systems remains sparse, but the work done on specific subjects, such as the detection and tracking and action description, is one of the most active in computer vision.

In this paper, we present a generic framework for event detection and behavior analysis. Motion detection is made difficult as both the observer and some elements of the scene may be moving. To handle this situation, we first compensate adjacent frames for the motion field induced by the observer which manifests itself globally. The results of this egomotion estimation are used to register frames, to detect independently moving objects, and to track them. Motion by itself, however, is not a sufficient indication of a threatening or otherwise interesting activity. In most natural scenes, there are a significant number of moving objects and it is the analysis of their trajectories and interaction with the features of the scene which allows us to classify and recognize interesting events. This scenario recognition, or behavior analysis module, is based on a crude model of the site being observed. This image to model correspondence helps in relating the observed motion to relevant features on the ground.

In the following, we first give an overview of our approach in Section 2 and then describe in detail the two modules: Detection and tracking and the graph representation of moving objects are discussed in Section 3. The graph obtained is then interpreted by the behavior analysis module discussed in Section 4. Besides demonstrating the system on real data streams, we characterize the performance of each module.

• G. Medioni, I. Cohen, S. Hongeng, and R. Nevatia are with the Institute for Robotics and Intelligent Systems, University of Southern California, Los Angeles CA 90089-0273.

E-mail: {medioni, icohen, hongeng, nevatia}@iris.usc.edu.

• F. Brémond is with INRIA Sophia-Antipolis, Projet ORION, 2004 route des Lucioles-BP 93, 06902 SOPHIA-ANTIPOLIS Cedex, France.

E-mail: francois.bremond@sophia.inria.fr.

Manuscript received 21 Apr. 1999; revised 20 July 2000; accepted 18 Apr. 2001.

Recommended for acceptance by R. Collins.

For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 109651.

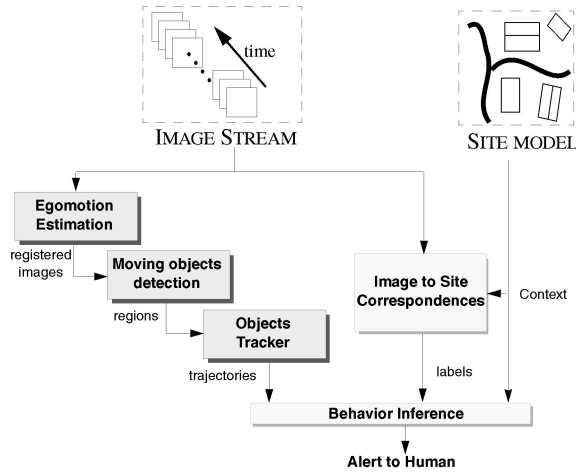


Fig. 1. Overview of the system.

## 2 OVERVIEW

Our approach is based on two modules. The first one achieves detection and tracking of moving objects from the video stream collected by the Unmanned Airborne Vehicle (UAV), while the second takes the inferred trajectory of each object and user-provided contextual information to recognize the behavior of the moving objects among a large number of potential scenarios. An overview of the system is given in Fig. 1. The most important features of each module are briefly described in the following.

Tracking the detected objects over the image sequence amounts to matching these different regions in order to determine the trajectories of the objects. This matching can be done using objects templates, color, or texture. For generality purposes, we propose instead to infer, from the detected regions, a 2D dynamic template of the object [8], [9]. A dynamic template is extracted by using the *temporal coherence* of the object over a number of frames (typically three to five frames). Temporal integration of the detected objects over a number of frames is used to characterize the moving objects by computing their motion, direction, and trajectory. A graph is used to represent moving regions and the way they relate to each other. Each node is a region and each edge represents a possible match between two regions in two different frames. We assign to each edge a cost which is the likelihood that the regions indeed correspond to the same object. This formalism can handle a large number of situations, such as stop and go motion, and allows us to characterize the trajectories of moving objects as an optimal path along each graph's connected component.

The behavior analysis module is based on the definition of a set of scenarios. These scenarios are defined by a combination of spatial and temporal properties. The behavior analysis module uses a set of temporal properties, such as distance of the moving object (the car) to either other interesting mobile objects (already detected vehicles), or a set of reference locations (intersection, buildings, etc.). The reference locations describe the most significant static objects in the scene. Currently, these objects are defined manually and consist of polygonal outline of the objects in the scene. This polygonal description is provided for the first frame of the sequence and propagated to the remaining

frames using the stabilization transforms. Each moving region in the scene, based on its trajectory, generates a set of hypotheses that are used to compute a set of properties. These scalar values are generic, reusable, and are used for identifying the behavior of the object as one of the available scenarios [4]. Moreover, the system relies on a recursive definition of scenarios. Each scenario is defined as a combination of subscenarios. A transition from a subscenario to the next is computed through a finite state automaton. This definition of scenario has the advantage of being flexible since any scenario of any time scale can be represented. Recognition is then achieved by computing a likelihood degree which allows us to state the reliability of a scenario recognition value.

The detection/tracking and behavior analysis modules are implemented to process the frames as they are acquired by the platform. Indeed, the detection of moving regions is achieved using the current and the previous frame. However, to achieve robustness, the tracking and the behavior analysis tasks are done on a buffered set of frames. This buffer is typically made of the last five frames and is used to build the graph representation of the moving objects and infer reliably a template of the moving object and its trajectory. This data is then used by the behavior analysis module for scenario recognition. The advantage of such an approach is that the two modules (tracking and behavior analysis) and the acquisition proceed in a concurrent way. Thus, we are able to characterize the behavior of the moving objects while the action is taking place (with a fixed delay of a few frames).

## 3 DETECTION AND TRACKING OF MOVING OBJECTS

### 3.1 Detection of Moving Objects

Most available techniques for detecting moving objects have been designed for scenes acquired by a stationary camera. These methods allow us to segment each image into a set of regions representing the moving objects by using a background differencing algorithm [17], [27], [20]. More recently, Grimson et al. [19] have proposed a local modeling of the background using a mixture of K-Gaussians allowing us to process video streams with time varying background. These methods give satisfactory results and can be implemented for real time processing without a dedicated hardware.

The availability of video sensors, at low cost, with Pan-Tilt and Zoom capabilities or video streams acquired by moving platforms have focused the attention of researchers on the detection of moving objects in a video streams acquired by a moving platform. In this case, background differencing techniques cannot be used. A stabilization "preprocessing" step has to be performed in order to cancel the camera motion. We propose integrating the detection into the stabilization algorithm by locating regions of image where a residual motion occurs. These regions are detected using the normal component of the optical flow field.

Normal flow is derived from image spatiotemporal gradients of the stabilized image sequence. Each frame of this image sequence is obtained by mapping the original frame to the selected reference frame. Indeed, let  $T_{ij}$  denote

the warping of the image  $i$  to the reference frame  $j$ . The mapping function is defined by the following equation:

$$\mathcal{T}_{ij} = \prod_{k=i, \dots, j+1} \mathcal{T}_{k, k-1} \quad (1)$$

and the stabilized image sequence is defined by  $\mathcal{I}_i = I_i(\mathcal{T}_{ij})$ . Here,  $I_i(\mathcal{T}_{ij})$  denotes the image  $I_i$  warped onto the reference frame  $I_j$  by the transform  $\mathcal{T}_{ij}$ . The estimation of the mapping function amounts to estimating the egomotion, based on the camera model, which relates 3D points to their projection in the image plane. The approach we use models the image induced flow instead of the 3D parameters of the general perspective transform [30]. The parameters of the model are estimated by tracking a small set of feature points  $(x_i, y_i)$  in the sequence. Given a reference image  $I_0$  and a target image  $I_1$ , image stabilization consists of registering the two images and computing the geometric transformation  $\mathcal{T}$  that warps the image  $I_1$  such that it aligns with the reference image  $I_0$ . The parameter estimation of the geometric transform  $\mathcal{T}$  is done by minimizing the least-square criterion:

$$E = D \sum_i \{I_0(x_i, y_i) - I_1(\mathcal{T}(x_i, y_i))\}^2, \quad (2)$$

where outliers are detected and removed through an iterative process. We choose an affine model which approximates well the general perspective projection in the case of video streams acquired by an airborne platform while having a low numerical complexity. Furthermore, a spatial hierarchy, in the form of a pyramid, is used to track selected feature points. The pyramid consists of at least three levels and an iterative affine parameter estimation produces accurate results.

### 3.1.1 Image Sequence Stabilization

Given a reference image  $I_r$  and a target image  $I_t$ , image stabilization consists of registering the two images and computing the geometric transformation  $\mathcal{T}$  that warps the image  $I_t$  such that it aligns with the reference image  $I_r$ . Several authors have proposed different approaches to solve this problem. The most common models involve affine [29], [28], [34] or quadratic approximation of the motion [29], [28]. General perspective models were also considered for generating full mosaics of a scene from a collection of pictures [45]. This induces a larger numerical complexity and was addressed in applications where the computation time is not an important issue. Among these methods, we can distinguish two types of approaches: intensity or feature-based approaches. The intensity approaches rely on *all* image pixels to recover the set of parameters, while the feature-based approach extracts a set of relevant points in the frames and derives the parameters according to the matching of these points. This second approach, selected in this study, is less time consuming and allows fast and accurate registration of the frames.

Recovering the parameters of the geometric transformation amounts to minimizing the least-square criterion given by (2).

This criterion leads to a global nonlinear minimization scheme which could be handled by a Levenberg-Marquardt method [44]. An alternative solution can be obtained iteratively, using all image points and a coarse to fine approach

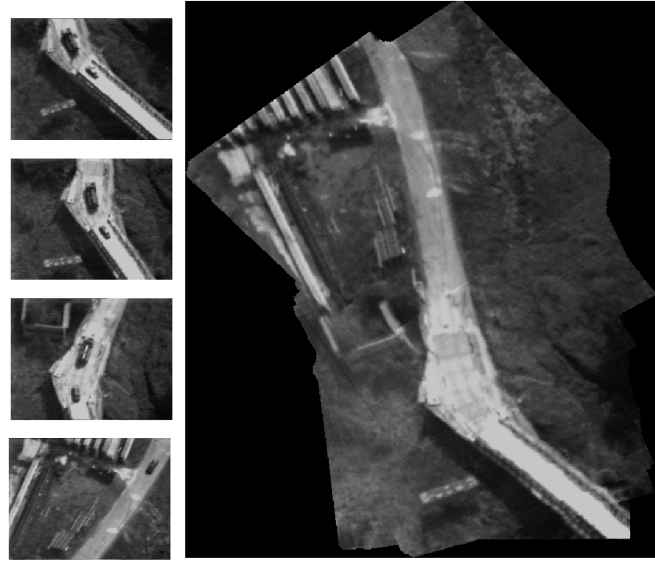


Fig. 2. Mosaic obtained by the hierarchical feature-based approach.

[38]. Both approaches are computationally expensive and, therefore, not suitable for a video surveillance application.

Our approach is based on multigrid matching of feature points extracted from the reference and target frames. There are several ways to define a feature point and each definition is context dependent [41]. Corners or high curvature points are commonly used as feature points in matching algorithms. The extraction of corner points can be done through the use of a *corner model* [48] or a rough estimation can be obtained through image partial derivatives [21]. In this paper, we extract the feature points by considering a partition of the image into a regular grid and extract in each cell the point which maximizes  $|\nabla_x I| + |\nabla_y I|$ . The feature points are selected at the coarsest pyramid resolutions of the reference image  $(x_r, y_r)$  and the target image  $(x_t, y_t)$  and are matched using local correlation. This gives us a set of pair of points from which we can derive the parameters of the motion model by minimizing the criterion given by (2). Using the affine model, the minimum for  $E$  in (2) is obtained by solving the set of linear equations. The system of linear equations is solved at each level of the pyramid by propagating the feature points and the parameters obtained at the previous level in order to refine the estimated parameters. Since this approach is based on a least-square, the obtained parameters may be biased by erroneous matched pairs of feature points or feature points belonging to an independently moving object in the scene. These outliers can be processed using a RANSAC [16] technique by selecting the best three pairs of points; however, this approach is computationally expensive as it requires the evaluation of  $E$  (2) for each possible combination. An alternate approach consists of discarding the point with the largest error and reestimating the parameters of the affine transform. This is repeated while the error measure is decreasing. This approach is more efficient than scaling the coefficients of the motion equations inversely proportional to the temporal derivatives [44], [30] since it does not require a warping of the images. Fig. 2 illustrates a byproduct of the processing, a mosaic obtained from a video stream.

### 3.1.2 Detection of Moving Objects

The moving objects in the scene are detected using normal flow field. However, the reference frame and the warped one do not, in general, have the same metric since, in most cases, the mapping function  $\mathcal{T}_{ij}$  is not a translation but a true affine transform and, therefore, it influences the computation of image gradients for moving object detection. Most of the approaches based on normal flow field rely first on warping the processed frame into the reference frame and then estimate a motion measure [31], [35] or the normal flow field [7] for detecting moving objects based on residual flow.

In this paper, we propose incorporating the change in metric into the optical flow equation associated to the image sequence  $\mathcal{I}_i$  (warped image). This allows us to derive the normal flow component which is based on the estimated affine transform and the original images in order to detect more efficiently the moving objects. Indeed, the optical flow associated to the image sequence  $\mathcal{I}$  is:

$$\nabla \mathcal{I}_i \nabla^T \mathcal{I}_i w = -\nabla \mathcal{I}_i \frac{d\mathcal{I}_i}{dt}, \quad (3)$$

where  $w = (u, v)^T$  is the optical flow. Expanding the previous equation, we obtain:

$$\begin{aligned} \nabla \mathcal{T}_{ij} \nabla \mathcal{I}_i(\mathcal{T}_{ij}) \nabla^T \mathcal{I}_i(\mathcal{T}_{ij}) \nabla^T \mathcal{T}_{ij} w = \\ -\nabla \mathcal{T}_{ij} \nabla \mathcal{I}_i(\mathcal{T}_{ij})(I_{i+1}(\mathcal{T}_{i+1,j}) - I_i(\mathcal{T}_{i,j})) \end{aligned} \quad (4)$$

and, therefore, the normal flow  $w_{\perp}$  is characterized by:

$$w_{\perp} = -\frac{(I_{i+1}(\mathcal{T}_{i+1,j}) - I_i(\mathcal{T}_{i,j})) \cdot \nabla \mathcal{T}_{ij} \nabla \mathcal{I}_i(\mathcal{T}_{ij})}{\|\nabla \mathcal{T}_{ij} \nabla \mathcal{I}_i(\mathcal{T}_{ij})\| \cdot \|\nabla \mathcal{T}_{ij} \nabla \mathcal{I}_i(\mathcal{T}_{ij})\|}. \quad (5)$$

Although  $w_{\perp}$  does not always characterize image motion due to the aperture problem, it allows detection moving regions. Indeed, the amplitude of  $w_{\perp}$  is large near moving regions and becomes null near stationary regions. A threshold allowing the detection of a displacement of one pixel per frame is then used to extract regions of moving objects.

The above definition of the normal flow component is based on the parametric model used for compensating for the camera's motion. It allows us to take into account the recovered affine transform for detecting moving objects without an explicit warping of the frames into the reference frame. Consequently, this approach increase the accuracy and the efficiency of the detection of moving objects since no linear interpolation of gray level is used. Indeed, methods based on the computation of residual flow after warping the images rely on the linear interpolation scheme used by the warping algorithm. Fig. 3 illustrates the detection of moving vehicles in a video stream taken from an airborne platform. It displays the bounding boxes (in white) of the detected moving objects.

## 3.2 Graph Representation of Moving Objects

The detection of moving objects in the image sequence gives us a set of regions which represent the locations where a motion was detected. The normal component given by (5) allows, given a pair of frames, detection of points of the image where a motion occur. These points are then aggregated into regions by considering a thresholded value of the normal component of the optical flow and then labeled using a 4-connectivity scheme. Each of these

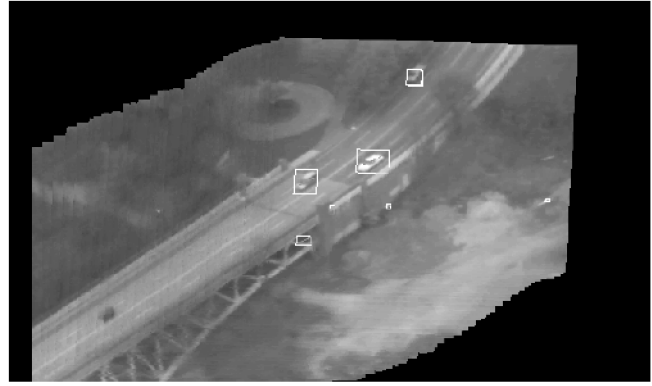


Fig. 3. Mosaic and detection of several vehicles in a video stream acquired by an airborne platform. The bounding boxes of the moving objects are displayed on the constructed mosaic.

connected components represents a region of the image where a motion was detected.

The purpose of detecting moving objects in video stream is to be able to track these objects over time and derive a set of properties from their trajectory such as their behavior. Commonly used approaches for tracking are token-based, when a geometric description of the object is available [15], or intensity-based (optical flow, correlation, etc.). These techniques are not appropriate for blob tracking since a reliable geometric description of the blobs cannot be inferred. On the other hand, intensity-based techniques ignore the geometric description of the blob. Our approach combines both techniques by incorporating in the representation of the moving objects both spatial and temporal information. Such a representation is provided by an attributed *graph* structure where nodes represent the detected moving regions and edges represent the relationship between two moving regions detected in two separate frames. Each newly processed frame generates a set of regions corresponding to the detected moving objects. We search for possible similarities between the newly and previously detected objects. Establishing such connections can be done through different approaches such as template matching [25] or correlation [47]. However, in video surveillance, little information about the moving object is available since the observed objects are of various types (car, humans, etc.) and at different scales. Also, objects of small size (humans in airborne imagery) or large changes of object size are frequent and, therefore, unsuitable for template matching approaches. We propose deriving a dynamic template of each detected moving object using the graph description of moving objects.

Each pair of frames gives us a set of regions where residual motion was detected (see Fig. 4). These regions can be related to the previously detected one by measuring the gray-level similarity between a region at time  $t$  and a set of regions at time  $t + 1$  located in its neighborhood. The size of this neighborhood is estimated from the objects motion amplitude. In this defined neighborhood, a region may have multiple matches and, therefore, a node in the graph representation may have several parent nodes and/or several children. In Fig. 4, we show a partial view of the graph representation associated to the detected blob in frame 123. As we can see, due to aperture problems, in frame 125, instead of detecting on single region representing the moving vehicle, we locate three regions. These

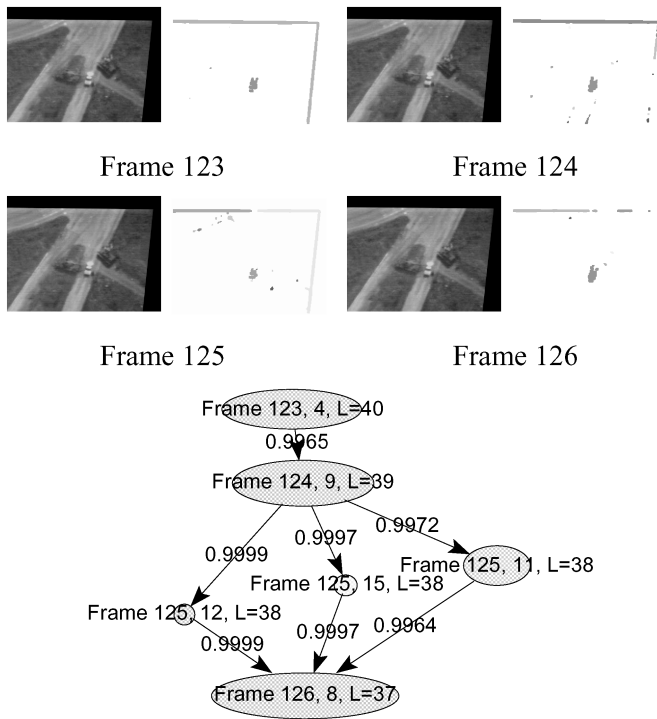


Fig. 4. Detected regions and associated graph. Each node in the graph represents a detected moving region. It is represented using an ellipsoid reflecting the principal radius (inverse of the eigenvalues of the autocorrelation matrix) of the region. The graph displayed here corresponds to a connected component of the graph associated to the moving car.

regions are related to the one detected in frames 124 and 126 through the graph representation. Each node is a region represented by an ellipsoid derived from the principal directions of the detected blob and the associated eigenvalues. Also, a set of attributes is associated to each node, as illustrated in Fig. 5. We assign to each edge a cost which is the likelihood that the regions correspond to the same object. In our case, the likelihood function is the image gray-level correlation between a pair of regions.

### 3.3 Dynamic Template Inference

The graph representation gives an exhaustive description of the regions where a motion was detected and the way these regions relate one to another. This description is appropriate for handling situations where a single moving object is detected as a set of small regions. Such a situation happens when, locally, the normal component of the optical flow is null (aperture problem) and, consequently, instead of detecting one region, we have a set of small regions. Usually, clustering techniques are applied for merging the detected blobs in order to recover the region corresponding to the moving object. These image-based techniques [27], [33] rely on the proximity of the blobs in the image and frequently merge regions that belong to separate objects.

Among the detected regions, some small regions should be merged into a larger one or have a trajectory of their own. In both cases, based on the graph representation, these regions belong to a connected component of the graph. In our approach, we cluster the detected regions in the graph rather than in a single image as used in previous efforts [27], [33]. Indeed, clustering through the graph prevents us from merging regions belonging to objects having a distinct

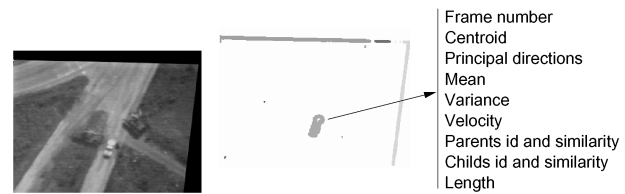


Fig. 5. Description of the attributes associated to each node of the graph. Each mark represents a moving region.

trajectory since clustering based on image proximity is done within a connected component of the graph.

The robustness of this clustering technique is also improved by maintaining a dynamic template of the moving objects for each connected component and, therefore, for each moving object in the scene. Several techniques were proposed for automatically updating a template description of the moving objects such as weighted shape description [33] or cumulative motion images [13]. The main drawback of these approaches is that errors in shape description (i.e., boundaries) are propagated and, therefore, these techniques are not suitable for streams obtained from a moving camera. We propose an approach based on a *median shape template* which is more stable and produces a robust description of templates. The templates are computed by applying a median filter (after aligning the centroid and the orientation of each blob) over the last five detected frames of the region.

This dynamic template allows us to infer a robust graph description of moving objects. Indeed, in video surveillance applications, objects often stop, then resume their motion, resulting in several connected components in the graph. These connected components are merged by using the dynamic template of the object being tracked: We propagate each node without a successor into a given number of frames and search for the matching regions in these areas. This defines a set of possible matches which are incorporated in the graph structure by defining new edges connecting the matched regions. This step is illustrated in Fig. 6, where the object, not detected in frame 104, is depicted by the node.

### 3.4 Extraction of Objects Trajectories

As new frames are acquired and processed, we incrementally construct the graph representation of moving objects. Deriving the trajectories of the objects from the graph and from the newly detected regions amounts to extracting a path along each graph's connected component. We propose an approach for automatically extracting the trajectories of all moving objects through the search of an optimal path representing object's trajectory. Furthermore, the starting node (source), as well as the destination node (goal), is not known in advance. Therefore, we consider each graph node without a parent as a potential source node and each node without a successor as a potential goal node. This property, along with the graph construction method described in Section 3.2, makes the proposed approach very similar to the Reid's Multiple Hypothesis Tracking (MHT) [40], [12], [11]. Indeed, the graph construction allows us to automatically create new tracks as new moving objects enter the field of view and to detect the termination of a track if the moving object is no longer detected for a period of time. These two properties are completed by a track continuation over several frames in the absence of detection (as described in Section 3.3 and depicted in Fig. 6). However, the MHT is not suitable for

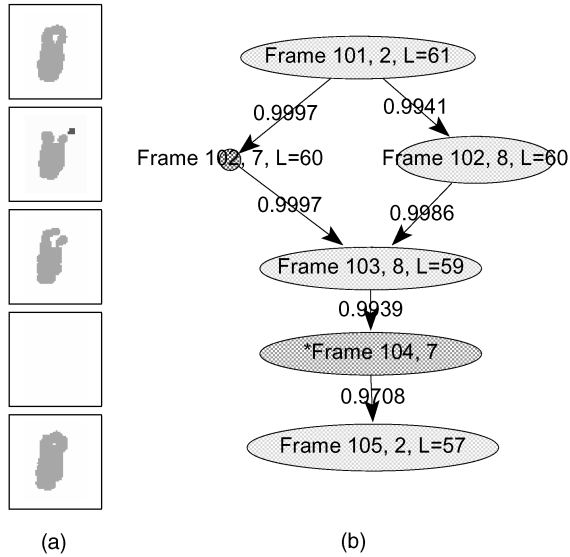


Fig. 6. Propagation of the nodes in order to recover the description of undetected objects. In (a), we show the detected region at each frame and, in (b), the associated graph where the node represents a node inferred from the median shape of the template.

regions tracking since, in many cases, one single moving object is split into a collection of regions due to the aperture problem. Therefore, we have to allow the tracker to model more than one child per node's representation.

A path or a trajectory is defined to be a sequence of measurements that are assumed to originate from the same moving object. Each graph's node is compared with the actual moving object detected in the next frame on the basis of a similarity measure that integrates regions cross correlation and the distance of their centroids. The purpose of this similarity measure is to associate a cost to each edge between two nodes of the graph. We have chosen the following measure:

$$c_{ij} = \frac{C_{ij}}{1 + d_{ij}^2}, \quad (6)$$

where  $C_{ij}$  is the gray level and shape correlation between the graph's nodes  $i$  and  $j$  and  $d_{ij}$  represents the distance between the centroids of the moving regions represented by  $i$  and  $j$ . The purpose of this measure is to give a larger value to similar regions (in terms of gray-level distribution) while penalizing distant regions. The centroid distance considered here corresponds to regions centroid distance after camera motion compensation. This has the advantage that the method is not limited to small camera motion.

Characterizing the object trajectory as a path that maximizes the costs associated to a collection of nodes allows us to rely on the cost measure to select the nodes that originated from the same moving object.

The edge cost given by (3) allows us to extract the local optimal path. Indeed, a graph search algorithm based only on the edge cost will provide a suboptimal solution since there are no constraints on the destination or goal node that have to be reached. In our different experiments, we have observed that this criterion yields a part of the trajectory. The goal source is selected based on the highest value of the cost, regardless of the other nodes belonging to the same connected component.

In the graph description used, each graph's connected component represents a moving object in the scene and each node's location in the graph allows us to characterize how far this node is from a potential goal node, i.e., a newly detected region. Such a characterization is done by assigning to each node the maximal length of graph's path starting at this node. The computation of the *node's length* is carried very efficiently by starting at the bottom of the graph, i.e., nodes without successor, and assigning, for each parent node, the maximum length of his successors plus one. The length of a node  $i$  is given by the following equation:

$$l_i = \max\{l_j, j \in \text{successor}(i)\} + 1 \quad (7)$$

with the initial estimate:  $l_i = 1$ , if  $\text{successor}(i) = 0$ .

The combination of (6) and the length of each node allows us to define a new cost function for each node. The cost function associated to the edge connecting the node  $i$  to the node  $j$  is then defined by:

$$C_{ij} = l_j c_{ij}, \quad (8)$$

where  $c_{ij}$  is defined by (6) and  $l_j$  is the length of the node  $j$  defined by (7). This cost function recovers the optimal path among the paths starting at the node being expanded.

The extraction of the optimal path is done by starting at the graph's nodes without parent and expanding the node with maximal value of  $C_{ij}$ . This approach is illustrated in Fig. 7, where the trajectories of a truck and a car are displayed along with false detections due to 3D parallax. Indeed, this image sequence displays a strong and consistent 3D parallax; therefore, the regions where a parallax motion is detected are kept since they cannot be discarded based only on temporal consistency. The proposed method does not address motions with 3D parallax.

### 3.5 Evaluation and Quantification

Given that the detection and tracking submodules feed on each other, it is necessary to characterize errors in each of these modules and their ripple effects. We have designed our system so that each submodule can handle temporary failures of the previous one and attempts to compensate for them using temporal coherence.

Few attempts were made in computer vision to evaluate such an integrated system where each component has its own inaccuracies and limitations. Our approach, based on a simultaneous processing of the detection and tracking and the efficient representation of the objects through a graph, allows us to derive a confidence measure after each of these tasks. Also, here we choose to evaluate each of the modules independently. The issues to be addressed are:

- **Detection.** Do we detect all the objects? Among the detected regions, how many of these are false alerts? What is the minimal size, speed of the moving objects that can be detected?
- **Tracking.** The temporal integration of the detected objects and their interrelationship allows us to infer, from a collection of blobs representing the moving objects, a set of paths corresponding to the trajectories of the moving objects. How accurate are these trajectories? What is the stability of the trajectories inferred with regard to failure to detect the moving

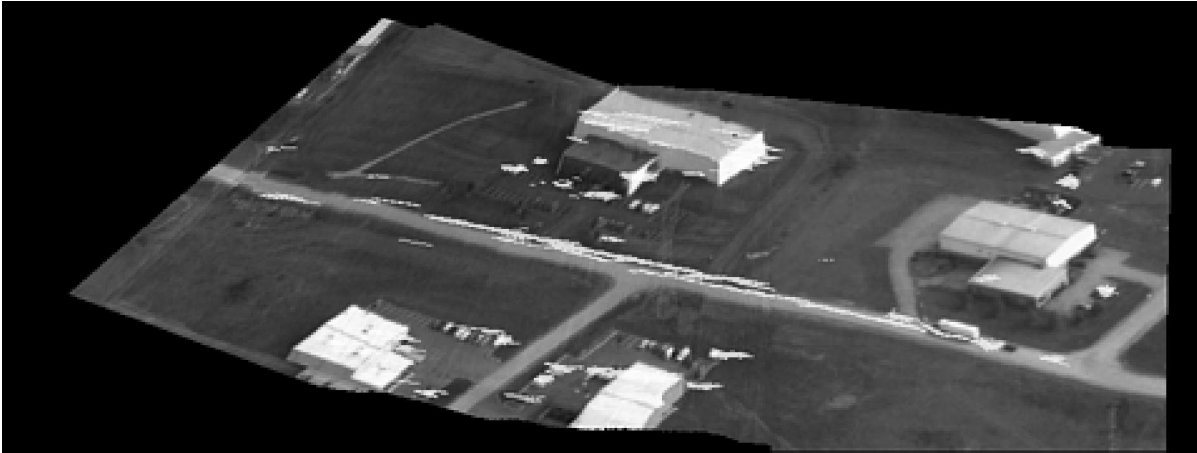


Fig. 7. Trajectories of the truck and the car mapped on the generated mosaic along with tracks of regions corresponding to 3D parallax. These are kept since they display good temporal consistency.

objects in one or several frames, stop and go motion, occlusion, etc.

- **Combined quantification.** Inferring a confidence measure of the output.

### 3.5.1 Evaluation of the Detection Submodule

The detection of moving objects, as described in Section 3.1, is performed after compensating for the motion of the platform. The moving objects are detected as regions where a residual motion subsists. The extracted regions are then due to moving objects present in the scene, to the inaccuracies of compensation algorithm used, and/or to the presence of parallax. Consequently, the number of regions extracted by the detection algorithm is larger than the number of moving objects in the scene and cannot be considered for quantifying the accuracy of the detection algorithm unless a static camera is used or a perfect egomotion estimation is achieved.

Our approach is based on a temporal integration of the moving objects over a certain number of frames which we call: the system's *latency time* (set here to five frames). This latency time, or delay, helps us in selecting the moving regions and distinguishing these blobs from inaccuracies due to the compensation of the camera's motion. Moreover, confidence in the extracted moving region increases as new occurrences of the objects are detected in the processed frames. Indeed, the length (see (7)) associated to each graph's node (i.e., moving region) represents the number of frames in which the object was detected. This scalar value allows us to discard detected blobs which are due to misregistration of the motion compensation algorithm since these regions have no temporal coherence, characterized by a small length. Table 1 gives some results obtained over several set of video streams acquired by the Predator UAV (Unmanned Airborne Vehicle) and VSAM (Video Surveillance and Activity Monitoring) platforms. These video streams represent a variety of scenes involving human activity and were used to evaluate the performance of our system.

The numerical values represent the output obtained at different stages of processing. The "Moving Objects" column represents the true number of objects moving in


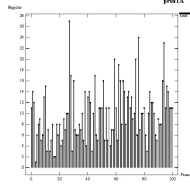

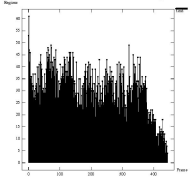

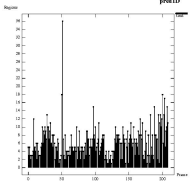

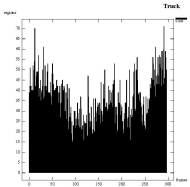

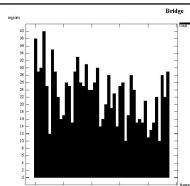
the video stream, as provided by the user. The next two columns represent the output of the detection and tracking submodules, respectively. As we can see, the number of regions detected is fairly large compared to the true number of moving objects. These numbers correspond to the number of regions where the normal flow field was larger than a given threshold ( $10^{-5}$ , in all the experiments). The detection column gives the distribution's plot of the number of these regions over the processed sequence. Also, the associated mean and variance are given as indicative values. The temporal integration of these regions, over a set of frames, allows us to reduce this number of regions (given in the fourth column) and discard *false detections* since regions due to noise are not temporally coherent. However, some inaccuracies of the egomotion model or the presence of a parallax can cause some regions to have a coherent temporal signature. Finally, the column "paths," represents the number of trajectories considered as valid, i.e., coherent temporal regions detected for more than 10 frames, which represents the latency time used in the tracking. In some cases, the number of trajectories is larger than the number of moving objects in the stream. This is due to object trajectories being fragmented into several paths and to failures in matching similar regions representing the same object. The remaining trajectories are due to regions with good temporal coherence which do not correspond to moving objects and are, mostly, due to strong parallax.

### 3.5.2 Evaluation of the Tracking Submodule

The temporal integration of the detected objects and their interrelationship allows us to infer from a collection of blobs, representing the moving objects a set of paths corresponding to the trajectories of the moving objects. The size of these paths (i.e., the number of node belonging to the path) allows us to easily filter out the regions where a temporal variation was detected with no coherence over time.

Deriving, from this set of paths, a measure to evaluate the tracking module is difficult since several issues have to be considered: Among the detected objects, how many were tracked correctly?

TABLE 1  
Quantitative Analysis of the Detection/Tracking Modules

video stream	Moving Objects	Detection			Tracking			Metrics	
		detected regions	mean	$\sigma$	Regions	Paths	Persis.	DR	FAR
	1		9	5	1	1	100%	1.	0.
	2		29	15	3	3	95%	1.	0.2
	4		6	3	4	5	85%	1.	0.
	2		34	11	10	5	50%	1.	0.8
	7		22	8	15	12	85%	1.	0.53

The column "Moving Object" displays the number of moving objects (provided by the user) in the processed image sequence. The "Detection" column shows some statistics on the number of detected moving regions. The temporal integration of these detected regions is displayed in the "Tracking" column. The use of a dynamic template and temporal consistency allows us to reject false detections. A trajectory is considered only if we manage to track an object during a least 10 frames. Therefore, we display the number of moving regions, the number of inferred paths, and their persistence. The last column shows the DR and FAR.

For each tracked object, how many paths form its trajectory? These issues are relevant in the case of stop and go motions or occlusion. Indeed, in the first case, the trajectory of the object is fragmented into a set of paths. These paths have to be merged into a single trajectory in order to recognize the stop and go motion that occurs, for example, in a checkpoint. The second case, partial or total occlusion, is more subtle since, before merging the collection of paths, one has to identify the object being tracked in order to recognize its different occurrences in the video stream. In order to quantify the performance of the tracker, we have defined a ratio that represents how long the tracker maintain an identification of the same moving object. This characterizes the persistence of the tracks. In Table 1, we

display, for a set of video streams, the number of paths detected and the number of moving objects in the scene along with the persistence ratio.

### 3.5.3 Quantification: Confidence Measure Definition

Finally, we have defined two metrics for characterizing the Detection Rate (DR) and the False Alarm Rate (FAR) of the system. These rates, used to quantify the output of our system, are based on:

- TP (true positive): detected regions that correspond to moving objects,
- FP (false positive): detected regions that do not correspond to a moving object, and
- FN (false negative): moving objects not detected.



These scalars are combined to define the following metrics:

$$DR = \frac{TP}{TP + FN} \quad \text{and} \quad FAR = \frac{FP}{TP + FP}.$$

These metrics are reported in Table 1. As the number of moving objects is small, these measurements may have large variances. This table shows that the large number of moving objects generated by the detection is reduced by the tracking, leading to a perfect detection rate in all examples. The large FAR in the last two experiments is due to 3D structures. In this case, further processing is needed in order to distinguish true motion from parallax.

## 4 BEHAVIOR INFERENCE

Given trajectories of moving regions, the task of the *behavior analysis* module is to identify the tracked moving regions as mobile objects and interpret the scenarios relative to their behaviors. In our case, the mobile objects correspond either to humans or to vehicles. To achieve this task, the system must correctly bridge the gap between the numerical image features of mobile objects and the symbolic description of the dynamic activities (e.g., “the car is speeding up”). This process is difficult as numerous sources of uncertainty and variation in the perceived data and the patterns of activities exist, requiring the system to handle uncertainties at every processing level systematically. Activities to be recognized are also varied depending on the applications. They can be as simple as a short movement such as “waving hands” or as complex as a long series of events described by some temporal and logical expressions. Our goal is to develop a behavior analysis module that can correctly abstract image feature information into symbolic scenarios in a systematic way so that these scenarios can be easily reused and adapted to the goal of applications or the quality of input data.

### 4.1 Related Work

There is a large amount of related work in the traditional AI field. For example, Galton, in [18], generates complex descriptions of human actions based on a set of generic basic spatial and temporal propositions. In [37], Neumann states that symbolic descriptions must be linked with properties defined at the image level and describes car scenarios based on a pyramidal hierarchy of motion verbs with elementary motion verbs at the base of the pyramid (corresponding to simple events) and complex ones at its top (corresponding to scenarios). This work, however, is targeted at defining activities at a symbolic level and not at establishing a link to the lower image data level.

Among computer vision research, a number of approaches attempting to match image features to symbolic activities have been investigated. For example, Davis and Bobick [13] developed an activity recognition method based on view-based template matching techniques. In this method, action is represented by a temporal template which is a static vector-image computed from accumulative motion properties at each point of the image sequences. An action is recognized by matching this template with the templates of known actions. Even though impressive results are obtained for short events in a constrained environment, temporal segmentation of complex events, confusion

among similar movements, and occlusion are difficult to handle. Davis et al. [14] represent simple periodic events (e.g., walking) by constructing dynamic models of periodic pattern of people’s movements and is dependent on the robustness of tracking. Inspired by a similar application to speech recognition, the *Hidden Markov Model* (HMM) has also been applied to activity recognition. Starner and Pentland [42] use an HMM to represent a simple event and recognize this event by computing the probability that the model produce the visual observation sequence. Parameterized-HMM [46] and coupled-HMM [3] are introduced to recognize more complex events such as an interaction of two mobile objects. Even though HMMs are robust against various temporal segmentations of events, the structure and probability distributions are not transparent to human and need to be learned using iterative methods. Therefore, for complex events, such networks and the parameter space may become prohibitively large. Some approaches use HMMs to represent primitive actions which are combined to define longer temporal events. For example, in [1], similar to probabilistic grammar parsing applied in natural language understanding, the stochastic context-free grammar parsing algorithm is used to compute the probability of a temporally consistent sequence of primitive actions recognized by HMMs. However, as the grammar becomes more complicated (as in the case of complex activities), the framework of stochastic parsing may become difficult. Finally, AI techniques such as temporal logic and algebra have also been applied in recent work. For example, in [39], events are represented by a network of subevents constrained by some temporal relations. The recognition methods, however, may require accurate event detection sensors. In [26], activities involving multiple agents in a football match are recognized by belief networks of events and specific functions that evaluate temporal relations between two events.

In this paper, we propose an alternative representation method, parts of which are similar to some of the above approaches. The key differences are that the links from high-level event descriptions to low-level image features are explicit in our case and that we represent temporal relations that we have found more common in our video surveillance domain.

### 4.2 Context

Context plays an important role in how actions are perceived and can guide their recognition. For example, to detect a burglary in a supermarket at night when the store is closed, it is sufficient to detect whether there is someone performing an action “picking up merchandise.” However, during the normal operating hours of the supermarket, this action does not indicate a burglary. Hence, we need different properties and methods to recognize it. In the following, we explain how we define, represent, acquire, and organize context and how its contents can be reused in other applications.

Several definitions of context have been used in previous work. For example, Nagel [36] defines the context of an action analysis process as a complex structure, comprised of generic descriptions for a spatial structures, temporal changes associated with these structures, and the intention

aspect of the action. Strat [43] defines the context of an image understanding process as all information that may influence the way a scene is perceived. In this paper, we define **Contextual Information** as the accessory information, other than sensed data, that is used during the processing to help the process to complete the task efficiently [5].

In our current application, we use two kinds of context: spatial context and mission context. The main source of our context is the scene environment information.

- **Spatial context** contains the spatial structures of the scene, the symbolic names (e.g., roads, checkpoint zones), and the static reference object of the environment that belongs to the area (e.g., the checkpoint).
- **Mission context** provides a priori expectations about scenarios to be detected (according to the goal of applications), specific recognition methods, and their parameters. For example, the mission context for “*monitoring a checkpoint*” consists of a set of recognition methods for interesting behaviors related to the checkpoint such as “*avoiding the checkpoint*” and “*passing through the checkpoint*” and a set of parameters required by the methods such as the expected size of vehicles.

In an ideal case, spatial context should be obtained from the identities and the world coordinates of objects in the scene which can be projected on to a 2D camera viewing plane at each frame, given the viewpoint and the parameters of the camera. However, since these parameters are not known for our examples, we use a 2D mosaic map as spatial context. This 2D mosaic is generated from the input sequence of interest based on affine transformation between consecutive images. The map is preprocessed by decomposing it into a partition of zones delimited by polygons. These zones and other static objects on the map are defined by a human operator. At each frame, spatial context is warped automatically into the current viewing frame using the affine parameters obtained from the stabilization method described in Section 3.1.1.

This decomposition helps us organize a larger and complex space into smaller and simpler subspaces. Each polygonal zone has a symbolic name (e.g., road, checkpoint, etc.) which links to other contextual information (e.g., mission context) that is related to that zone. For example, *checkpoint* links to the mission context related to “*monitoring the checkpoint*.” When a car is approaching a checkpoint, via this link we can trigger recognition methods to check if it is passing through the checkpoint or it is trying to avoid the checkpoint.

Other contextual information referenced by the link is kept in a library. For example, the mission context related to “*monitoring a checkpoint*” may consist of several routines that recognize all scenarios related to “*checkpoint*” such as “*going through the checkpoint*,” “*avoiding the checkpoint*,” etc. These routines are kept in a library of recognition methods and can be reused when we monitor similar scenarios or the same scenarios at different scenes.

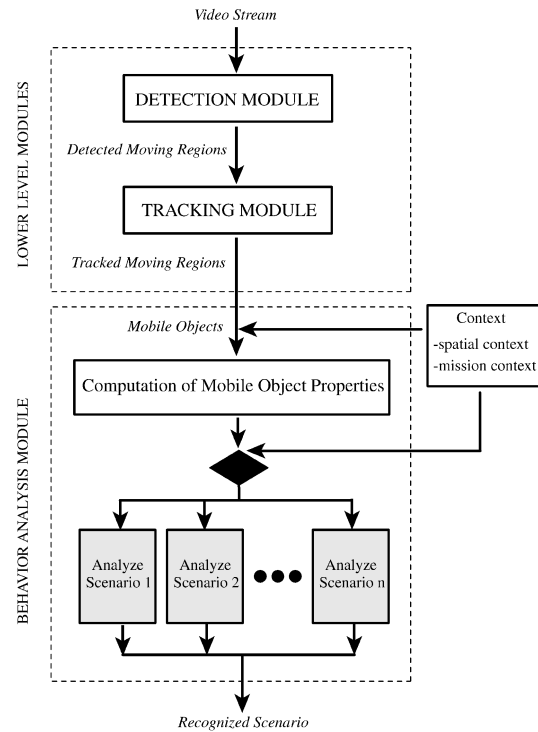


Fig. 8. Overview of the behavior analysis module.

### 4.3 Scenario Representation

To bridge the gap between a high level, symbolic description of an activity and the signal level information provided by the sensors, a hierarchy of **entities** is proposed. The entities involved in the activity recognition process are organized into three levels of abstraction: **image features**, **mobile object properties**, and **scenarios**. First, the image processing module detects moving regions and computes several 2D **image features** using the visual data from a single frame. Currently, we use eight features which include *height*, *width*, *speed*, *motion direction*, and *the distance to a reference object*. The tracking module then tracks the detected regions which can correspond to noise, to a part of a mobile object (e.g., an arm of a person), to one mobile object (e.g., a person), or to a group of mobile objects (e.g., a crowd). Image features are imprecise and are not interpreted. The behavior analysis module generates hypotheses to consider the tracked moving regions as mobile objects composed of one or several regions. Several layers of spatial and temporal **properties** of these mobile objects are then computed. **Scenarios** relative to the behavior of mobile objects are analyzed based on the mobile object properties. Fig. 8 shows the overview of our behavior analysis module. For each mobile object, we analyze a set of predefined scenarios. The system will output the recognized scenario with the highest priority, recognition value, and likelihood degree. Priorities of scenarios are provided as context by the user based on the goal of applications. In the figure, spatial context is used to compute mobile object properties. Mission context is used to access the scenario recognition methods related to the goals of the application. We describe mobile object properties and scenarios in the following.

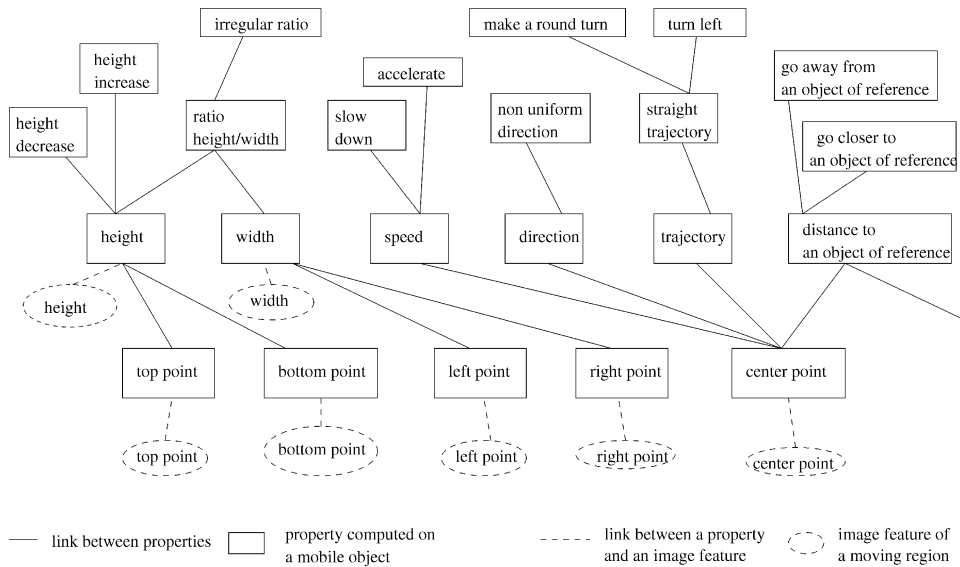


Fig. 9. Several layers of spatial temporal mobile object properties are defined.

### 4.3.1 Mobile Object Properties

Mobile object properties correspond to either static properties (e.g., *the width* and *the height*) or instantaneous actions (e.g., *entering the checkpoint* and *to be in the zone*) of mobile objects and can be computed from input image sequences. They have a numerical value and can be defined in three ways: 1) from the corresponding image features, 2) from a set of subproperties, and 3) from the temporal evolution of property values. Fig. 9 shows several layers of mobile object properties used in our system.

At the lowest layer, mobile object properties are computed from the corresponding image features by applying some specific functions on the feature values collected over a few number of frames to smooth the noisy data (e.g., to remove the outliers or just to average the noisy data). In our current system, three functions are used: mean, multi-Gaussian, and filtering. Mean function computes the arithmetic average of a list of features taken from a temporal sequence. Multi-Gaussian function takes in the current feature and estimates the property value based on the (multi-Gaussian) distribution of the features collected from the first frame to the current frame. The filtering function takes in a sequence of feature values and estimates the current value using linear regression. At a higher layer, for a mobile object property that is defined from a set of subproperties or context, we provide a set of arithmetic functions (e.g., a division function to compute the ratio of width property and height property) and other specific functions (e.g., a function that estimates a trajectory or a function that determines whether the mobile object lies within a polygonal zone defined as context). For a mobile object property that is defined from temporal evolution of property values, several functions that take in a temporal sequence of properties and compute the nature of its change (e.g., increasing, decreasing, and stable) are provided. To indicate how reliable the property value is, a likelihood degree, ranging from 0 (unreliable) to 1 (reliable), is computed using temporal coherence: If the current value is coherent with the old ones, we increase the likelihood

degree by 0.1; otherwise, it is decreased by the same amount.

Mobile object properties are defined in a generic way (i.e., independent of applications) and are reusable for different applications. They are represented by a model, shown in Fig. 10a, composed of the following slots: property name, involved objects and their roles, list of subproperties, property value (numeric), method to compute the value, likelihood degree of property, and a set of methods to compute the likelihood degree.

### 4.3.2 Scenario Modeling

Scenarios correspond to activities that occur across a long sequence of frames. They are described and recognized recursively based on the properties of the mobile objects involved in the scenarios. At level 0, a scenario is recognized directly from the associated mobile object properties. For example, *speed of a car* (to determine whether the speed of a mobile object is matched with the expected speed of a car) is recognized by comparing the property *speed* of the mobile object with the Gaussian distribution of the car speed (provided as a context by the user). At level *n*, a scenario is recognized from a temporal combination of subscenarios

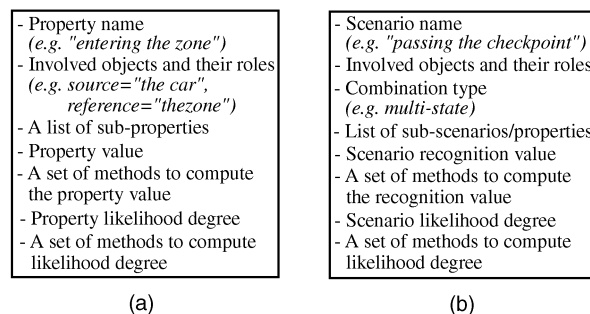


Fig. 10. Models of property and scenario are defined similarly. (a) Property Model. (b) Scenario Model.

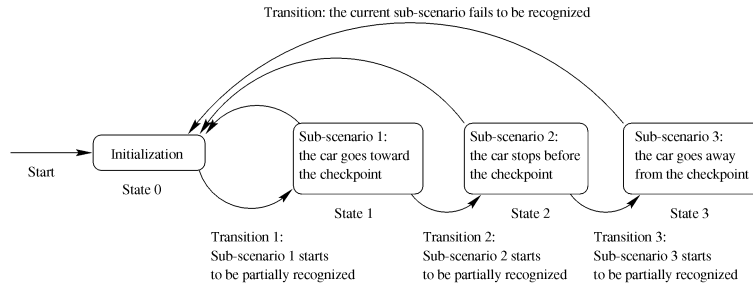


Fig. 11. Automaton for the scenario "the car avoids the checkpoint."

recognized at lower levels. We define two types of temporal combinations: **single-state** and **multistate**.

In the case of a **single-state combination**, a scenario corresponds to a constraint that a set of subscenarios must be recognized at the same instant. For example, the scenario "the car goes toward the checkpoint" represents a single-state combination of three subscenarios: "the distance between the car and the checkpoint is decreasing," "the direction of the car is toward the checkpoint," and "the speed of the car is decreasing." If these subscenarios occur simultaneously, this scenario is said to be recognized. Scenarios defined through a single-state combination are called single-state scenarios.

In the case of a **multistate combination**, a scenario corresponds to a temporal sequence of subscenarios and is called a multistate scenario. For example, the scenario "the car avoids the checkpoint" represents a multistate combination of three subscenarios: "the car goes toward the checkpoint," "the car stops before the checkpoint," and "the car goes away from the checkpoint." This scenario is recognized when its three subscenarios are consecutively recognized.

Scenarios are represented by a model (see Fig. 10b) that is similar to the property model but composed of eight parts: scenario name, involved mobile objects, a combination type (*single state* or *multistate*), a list of subscenarios, a recognition value, a set of methods to compute the recognition value, a likelihood degree, and a set of methods to compute the likelihood degree. The recognition value of a scenario is a numerical value, ranging from 0 (unrecognized) to 1 (recognized), that indicates the degree of recognition.

In contrast with other methods where image features are linked directly to symbolic scenarios, our proposed hierarchical scenario modeling method allows us to easily adapt scenario models to suit the goal of the application of interest.

#### 4.4 Scenario Recognition Methods

We describe the methods to recognize a single state scenario and a multistate scenario in the following.

##### 4.4.1 Single-State Scenario Recognition

In the case of a single-state scenario, a value is associated to quantify the verification of the constraint that all subscenarios are occurring or true at that instant. The verification process combines the recognition values and the likelihood degrees of the subscenarios to infer the recognition value of the single-state scenario. The parameters (e.g., weights) used in the combining process are currently determined by experiments but can also be learned. The likelihood degree, which states the stability of the scenario value, is then computed based on

*temporal coherence* to indicate how reliable the recognition value is. During the processing, if the new scenario value is above a threshold (provided by the user through mission context based on the degree of image noise) and coherent with the old ones, we increase the likelihood degree (ranging from 0 to 1) of the scenario by 0.1; otherwise, it is decreased by the same amount. For example, the single-state scenario "getting closer to a checkpoint" can be verified by taking the average of the weighed recognition values of its subscenarios (e.g., "the distance to the checkpoint is decreasing" and "the direction of the mobile object is towards the checkpoint"), where the likelihood degrees are used as the weights.

##### 4.4.2 Multistate Scenario Recognition

If a scenario is represented by a temporal sequence of subscenarios, it is recognized by a finite-state automaton whose states correspond to the subscenarios. The automaton transitions are computed from the recognition values and the likelihood degrees of subscenarios. The transition between state  $n - 1$  and state  $n$  occurs when the recognition value and the likelihood degree of the subscenario corresponding to state  $n - 1$  is achieved (i.e., above the threshold values) and the recognition values of state  $n$  are larger than 0. The recognition value of a multistate scenario is computed as the average recognition value of its subscenarios. For example, after the transition to state  $n$  is made, the recognition value is computed as the cumulative recognition values of the previously visited states (i.e., the recognition value of the subscenario at the time the transition to the next subscenario is made) up to state  $n - 1$  added by the current recognition value of state  $n$ . The recognition value is normalized by the total number of state transitions. The likelihood degree of a multistate scenario is computed by averaging the likelihood degrees of its subscenarios.

At each frame, all subscenarios (states) are evaluated and all possible transitions are considered. That is, all possible temporal sequences of subscenarios are evaluated in parallel such that, if evidence for a multistate scenario exists, it will be detected. Fig. 11 shows a finite-state automaton that recognizes the multistate scenario "a car is avoiding a checkpoint." This scenario is composed of three single-state subscenarios: "the car goes toward the checkpoint," "the car stops before the checkpoint," and "the car goes away from the checkpoint." It is recognized when the three subscenarios are consecutively recognized and the likelihood degree is high enough (i.e., above a threshold).

All scenario recognition methods are organized as a library. They are associated with a mobile object through

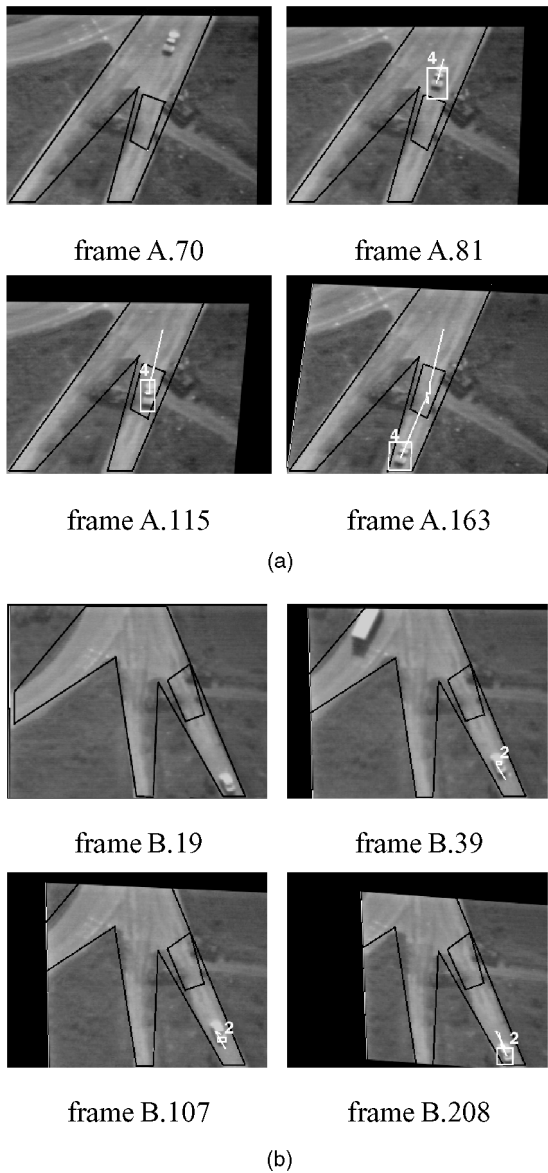


Fig. 12. Two competing multistate scenarios related to the monitoring of a checkpoint. (a) A car is passing through the checkpoint. (b) A car is avoiding the checkpoint.

the link provided by the mission context. For example, “road” provides a link to a set of scenario recognition methods that are related to “monitoring a road.” Through the methods based on the single state constraint verification and the finite state automaton, scenarios described by our scenario modeling approach can be recognized by propagating the recognition values and the likelihood degrees from the input mobile object properties at the bottom level to the scenarios defined at a higher level. When a recognized scenario matches with the goal of a given application, an alarm is triggered. If the object of interest performs an activity that is totally outside the scope of all scenarios in the library, it will not be evaluated.

#### 4.5 Experimental Results

We tested our system on several video-surveillance streams obtained by the Predator UAV. The goal of these experiments is to show that our system can 1) discriminate and correctly

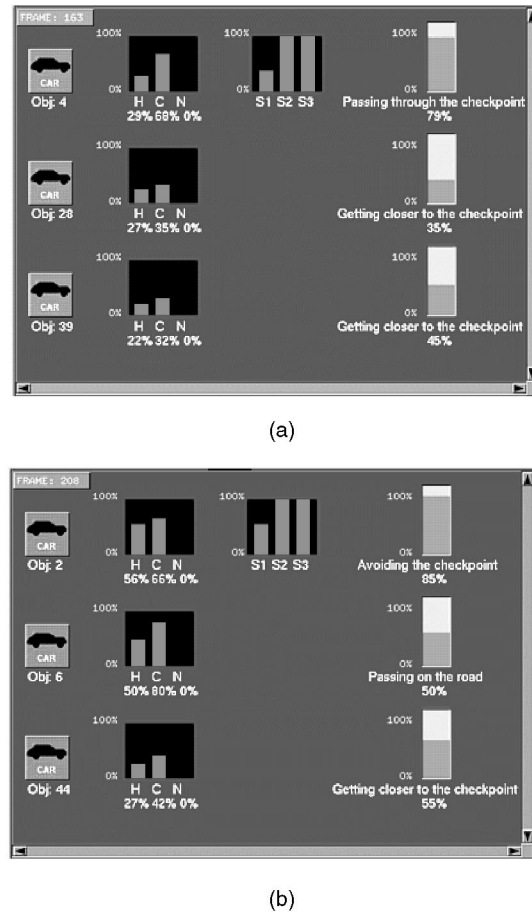


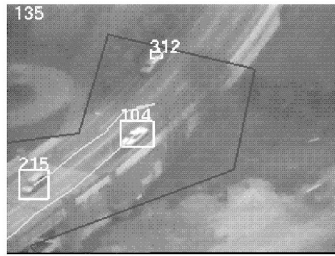
Fig. 13. Scenario with the highest likelihood degree is shown for each mobile object detected in the image sequences. The system can discriminate between two competing actions: “Passing through the checkpoint” and “Avoiding the checkpoint.” (a) Results at frame 163 of sequence A. (b) Results at frame 208 of sequence B.

recognize among competing scenarios, 2) recognize scenarios given imperfect tracking results, and 3) recognize more complicated scenarios involving several mobile objects.

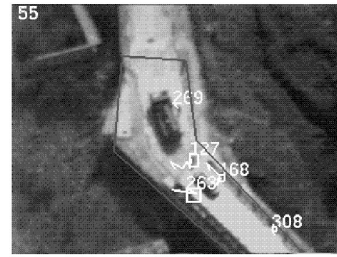
#### 4.5.1 Discrimination Among Competing Scenarios

First, we apply the system on the monitoring of checkpoints (i.e., road blocks). Fig. 12 shows two image sequences acquired at 15 Hz from an airborne platform. These sequences depict two competing car behaviors related to the monitoring of a checkpoint. Sequence A shows multistate scenario 1, “a car is passing through the checkpoint” (defined as a normal behavior). Sequence B shows related competing multistate scenario 2, “a car is avoiding the checkpoint” (defined as an abnormal behavior). The map of the scene is obtained from a 2D mosaic. We have drawn two polygons to delimit two contextual zones: the road and the checkpoint. The rectangles correspond to the bounding boxes of the moving regions associated with the detection of the car and the lines correspond to the car trajectory computed by the system. Both scenario 1 and 2 are analyzed for each detected mobile object on the road in these image sequences.

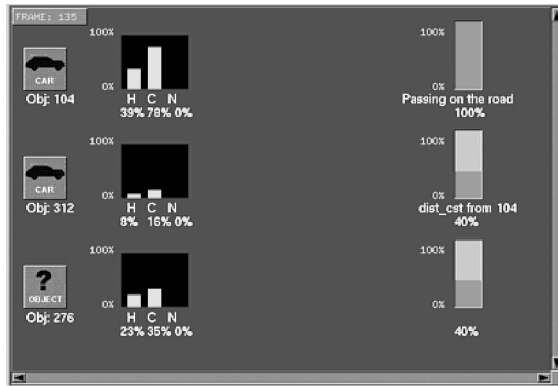
Fig. 13 shows the results of behavior analysis of sequence A and B. Detected mobile objects and their recognized behavior are shown in each row. The first



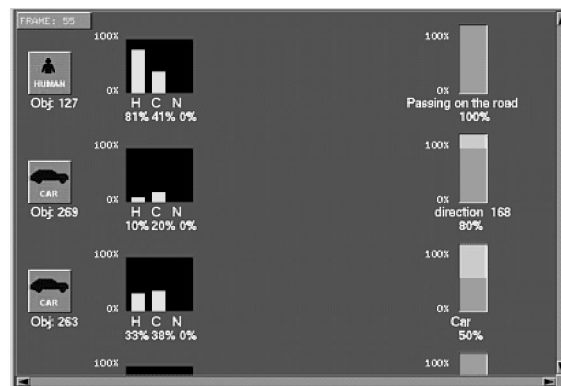
frame C.135



frame D.55



results of sequence C



results of sequence D

Fig. 14. Correct vehicle behaviors are recognized on a sequence containing a 3D parallax. In this sequence, objects are tracked unreliably and are confused at times with other vehicles appearing near by.

column shows the ID number and the most likely category of mobile objects: human (H), car (C), or noise (N). The second column is a bar graph that shows the recognition result of each category. The last two columns show the most likely scenario of the mobile objects and the recognition results of its composite subscenarios if the scenario is a multistate scenario. The numbers shown under the graph bars indicate the derived likelihood degrees.

In Fig. 13a, at frame 163 of sequence A, object 4 is recognized as "a car" (as opposed to human and noise) and its behavior is analyzed. Scenario 1, "passing through the checkpoint," is recognized with the highest likelihood degree by consecutively recognizing its subscenarios: "the car is going toward the checkpoint" (S1), "the car is in the checkpoint" (S2), and "the car is leaving the checkpoint" (S3), with the likelihood degree of 0.38, 1.0, and 1.0, respectively. On the other hand, scenario 2 is not recognized for object 4 because subscenario "the car stops before the checkpoint," required as a second state of the scenario "the car is avoiding the checkpoint," fails to be recognized. Mobile objects 28 and 39 do not correspond to either a human or a car. Even though the behaviors of these noisy moving blobs are analyzed, the results show that "getting closer to the checkpoint" is matched with very low likelihood degrees of 0.35 and 0.45, respectively. These behavior recognition results can, in turn, be used as feedback to low-level vision routines to improve the detection and tracking results.

Fig. 13b shows the results of sequence B. At frame 208, object 2 is recognized comparatively as both "a car" and "a human" due to the poor quality of the moving region detection results. Scenario 2, "the car avoids the checkpoint," is recognized with the likelihood degrees of its three

Fig. 15. Behaviors of small mobile objects (human) with short tracking are correctly recognized.

subscenarios as 0.55, 1.0, and 1.0. Competing scenario 1, "the car passes through the checkpoint," fails to be recognized for this object because the subscenario, "the car is in the checkpoint," is not successfully recognized.

#### 4.5.2 Scenario Recognition on Poor Quality Tracking Results

We applied our system to two other different sequences by changing only the context to test our system on noisy data and inaccurate tracking results.

Fig. 14 shows frame 135 of an image sequence where several cars are crossing a bridge. In this sequence, the 3D geometry of the bridge causes a 3D parallax problem, which sometimes leads to a faulty tracking of the vehicles. In the same figure, we show our recognition result, where object 104 is recognized as "a car" and the single-state scenario "passing on the road" is recognized with the highest likelihood degree. The recognition value of this scenario is computed by combining the recognition of its subscenarios such as "the mobile object is on road" and "the trajectory is straight." Other scenarios such as "speeding up" and "slowing down" are also analyzed, but are recognized with lower priority, recognition value and likelihood degree. Since a checkpoint is not defined in the context, scenarios regarding the checkpoint are not evaluated. As for object 312, since it has been tracked for only a few frames, it is recognized as a car with a low likelihood degree (below 20 percent) and the scenario "distance to object 104 is constant" is not very reliable (40 percent). Mobile object 276 corresponds to a car that was tracked for a long time but is currently out of the viewing area.

Fig. 15 shows frame 55 of another sequence where object 127, which corresponds to "human," is wandering

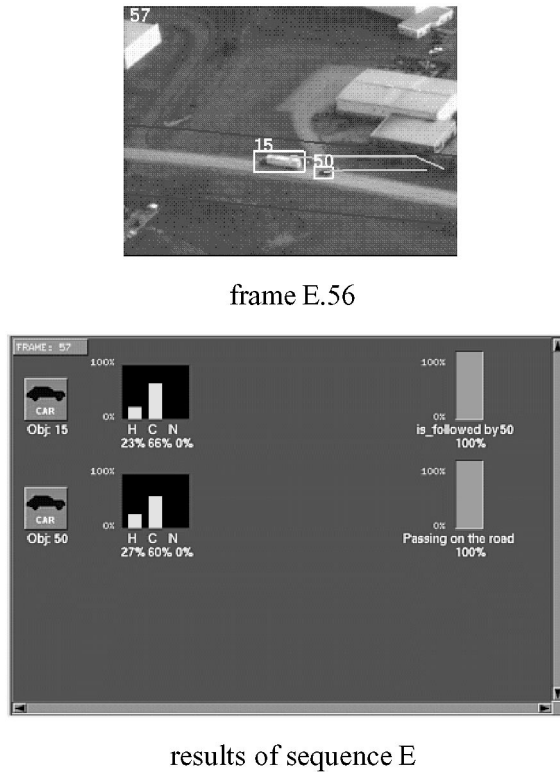


Fig. 16. Vehicle behaviors related to another vehicle and results of scenario recognition.

around on the road. In this sequence, the mobile objects representing humans are small and tracked for only a few frames (20 frames out of 200 frames). The same figure shows the scenario recognition results for all detected moving objects. Object 127 is recognized as a human and the most likely behavior is “*passing on the road,*” which matches our interpretation.

#### 4.5.3 Recognition of Scenarios Involving Several Mobile Objects

We also tested our system on the single-state scenario, “*the vehicle is followed by another vehicle,*” which involves two mobile objects. Fig. 16a shows frame 56 of an image sequence where a car is following a truck. Our system recognizes that mobile object 15 (a truck) “*is followed by*” mobile object 50. This scenario is recognized because all constraints such as “*the distance between object 15 and object 50 is almost constant,*” “*both objects are on the road and heading in the same direction,*” and “*their trajectories are aligned*” are verified.

#### 4.5.4 Discussion of Results

To achieve the goal of a generic behavior analysis system, the system must be applicable to a large number of sequences in different scenes and, at the same time, adaptable to the goals of various applications. This requires a general representational framework that provides a link from low-level image data to long and complex activities.

In this paper, we provide a first, but crucial, step to achieve this framework: a scenario modeling method that establishes the link from low-level image data to symbolic scenarios in a

systematic way. This is achieved through three levels of abstractions: image features, mobile object properties, and scenarios. Properties are generic and can be applied on a large number of sequences. Scenarios are represented by various combinations of a set of available mobile object properties and can be adapted to different applications.

Five sample scenarios were shown in Section 4.5. Even though they appear to be simple, given the tracking problems (e.g., mobile objects are sometimes not detected for a few frames or confused with other mobile objects near by) and behavior representation issues (discussed in Section 4), recognizing these scenarios in real image sequences is, in fact, a complex and challenging task. Another interesting characteristic of these scenarios is the fact that they are symbolic. Scenarios recognized by our system can therefore be combined at the symbolic level using temporal logic and other event reasoning [10], [22], [6] to recognize more sophisticated activities (e.g., a long combination of activities of several individual mobile objects or activities at different scenes).

The effectiveness of our behavior analysis system is still limited by the quality of detection and tracking of mobile objects. Our recognition method can recover the loss of tracking of mobile objects by comparing the spatial location and the characteristics of the lost mobile objects with those of other recently detected mobile objects. If they are similar in the characteristics and in the same proximity, the tracking can be retrieved. However, when the tracking of mobile objects is lost for a long time or when several similar mobile objects appear nearby, the behavior analysis may fail. We are currently developing more robust recognition methods that handle uncertainty in a more rigorous way. We have obtained some satisfactory results toward the first step of the development of these new techniques in [23].

## 5 CONCLUSION AND FUTURE WORK

We have addressed several problems related to the analysis of a video stream. The system is based on the integration of a detection and tracking module and a behavior inference module and can handle noisy data and inaccurate detections. Further research and development is planned in order to improve the stabilization by using other parametric model such as quadratic or homographic projections. These enhancements will improve the quality of the detection algorithm. Also, special consideration will be given to the quantification of false or nondetection rates. This will allow us to infer a confidence measure for the obtained results. Improving the reliability of the behavior inference module will increase its stability with regard to false and nondetection of the moving objects. We propose an alternative approach to represent and recognize behavior. We show by real world examples that our recognition methods can discriminate between related competing scenarios and can handle a small amount of imperfect detection and tracking of mobile objects. Scenarios involving two mobile objects are also successfully recognized. Our recognition methods can be made more robust against poor quality of detection

and tracking results, typical of real video streams, by computing uncertainty in a more rigorous way. More complex activities involving a complex combination of symbolic scenarios computed by our systems can be achieved by reasoning at symbolic level.

## ACKNOWLEDGMENTS

This research was supported by the US Defense Advanced Research Projects Agency (DARPA) under contract DAAB007-97-C-J023, monitored by the US Army, Fort Monmouth, New Jersey.

The authors would like to thank Sarnoff/CMU/Night Vision Labs for providing some of the airborne video streams from the VSAM airborne platform used in this paper as illustrations.

## REFERENCES

- [1] A. Bobick and Y.A. Ivanov, "Action Recognition Using Probabilistic Parsing," *IEEE Proc. Computer Vision and Pattern Recognition*, June 1998.
- [2] A.F. Bobick, A.P. Pentland, and T. Poggio, "VSAM at the MIT Media Laboratory and CBCL: Learning and Understanding Action in Video Imagery PI Report 1998," *Proc. DARPA Image Understanding Workshop*, pp. 85-91, 1998.
- [3] M. Brand, N. Oliver, and A. Pentland, "Coupled Hidden Markov Models for Complex Action Recognition," *IEEE Proc. Computer Vision and Pattern Recognition*, 1997.
- [4] F. Brémond and G. Medioni, "Scenario Recognition in Airborne Video Imagery," *Proc. DARPA Image Understanding Workshop*, 1998.
- [5] F. Brémond and M. Thonnat, "Issues of Representing Context Illustrated by Video-Surveillance Applications," *Int'l J. Human-Computer Studies*, special issue on context, 1998.
- [6] H. Buxton and S. Gong, "Visual Surveillance in a Dynamic and Uncertain World," *Artificial Intelligence*, vol. 78, nos. 1-2, pp. 431-459, 1995.
- [7] I. Cohen and I. Herlin, "Non Uniform Multiresolution Method for Optical Flow and Phase Portrait Models: Environmental Applications," *Int'l J. Computer Vision*, vol. 33, no. 1, pp. 1-22, 1999.
- [8] I. Cohen and G. Medioni, "Detecting and Tracking Moving Objects in Video from an Airborne Observer," *Proc. DARPA Image Understanding Workshop*, 1998.
- [9] I. Cohen and G. Medioni, "Detecting and Tracking Moving Objects for Video Surveillance," *IEEE Proc. Computer Vision and Pattern Recognition*, June 1999.
- [10] D. Corral, "Deliverable 3: Visual Monitoring and Surveillance of Wide-Area Outdoor Scenes," Technical Report Esprit Project 2152: VIEWS, June 1992.
- [11] I.J. Cox and S.L. Hingorani, "An Efficient Implementation of Reid's Multiple Hypothesis Tracking Algorithm and Its Evaluation for the Purpose of Visual Tracking," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 2, pp. 138-150, Feb. 1996.
- [12] I.J. Cox and M.L. Miller, "On Finding Ranked Assignments with Application to Multi-Target Tracking and Motion Correspondence," *AeroSys*, vol. 32, no. 1, pp. 486-489, Jan. 1995.
- [13] J.W. Davis and A.F. Bobick, "The Representation and Recognition of Human Movement Using Temporal Templates," *IEEE Proc. Computer Vision and Pattern Recognition*, pp. 928-934, June 1997.
- [14] L. Davis, R. Chelappa, A. Rosenfeld, D. Harwood, I. Haritaoglu, and R. Cutler, "Visual Surveillance and Monitoring," *Proc. DARPA Image Understanding Workshop*, pp. 73-76, 1998.
- [15] O. Faugeras, *Three-Dimensional Computer Vision*. MIT Press, 1993.
- [16] M.A. Fischler and R.C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Comm. ACM*, vol. 24, no. 6, pp. 381-395, June 1981.
- [17] B. Flinchbaugh, "Reliable Video Event Recognition for Network Cameras," *Proc. DARPA Image Understanding Workshop*, pp. 81-83, 1998.
- [18] A. Galton, "Towards an Integrated Logic of Space, Time and Motion," *Proc. Int'l Joint Conf. Artificial Intelligence (IJCAI)*, Aug. 1993.
- [19] W.E.L. Grimson, L. Lee, R. Romano, and C. Stauffer, "Using Adaptive Tracking to Classify and Monitor Activities in a Site," *IEEE Proc. Computer Vision and Pattern Recognition*, pp. 22-31, 1998.
- [20] I. Haritaoglu, D. Harwood, and L.S. Davis, "W4S: A Real-Time System for Detecting and Tracking People in 2 1/2-D," *Proc. European Conf. Computer Vision*, 1998.
- [21] C. Harris and M.J. Stephens, "A Combined Corner and Edge Detector," *Alvey88*, pp. 147-152, 1988.
- [22] G. Herzog, "From Visual Input to Verbal Output in the Visual Translator," *Projet VITRA 124*, Universität des Saarlandes, Saarbrücken, Germany, 1995.
- [23] S. Hongeng, F. Bremond, and R. Nevatia, "Representation and Optimal Recognition of Human Activities," *IEEE Proc. Computer Vision and Pattern Recognition*, 2000.
- [24] R.J. Howarth and H. Buxton, "Visual Surveillance Monitoring and Watching," *Proc. European Conf. Computer Vision*, vol. II, pp. 321-334, 1996.
- [25] D.P. Huttenlocher, J.J. Noh, and W.J. Rucklidge, "Tracking Non-Rigid Objects in Complex Scenes," *IEEE Proc. Int'l Conf. Computer Vision*, pp. 93-101, May 1993.
- [26] S. Intille and A. Bobick, "Visual Recognition of Multi-Agent Action Using Binary Temporal Relations," *IEEE Proc. Computer Vision and Pattern Recognition*, June 1999.
- [27] S.S. Intille, J.W. Davis, and A.F. Bobick, "Real Time Closed World Tracking," *IEEE Proc. Computer Vision and Pattern Recognition*, pp. 697-703, 1997.
- [28] M. Irani and P. Anandan, "Robust Multi-Sensor Image Alignment," *Proc. DARPA Image Understanding Workshop*, vol. 1, pp. 639-647, May 1997.
- [29] M. Irani and P. Anandan, "A Unified Approach to Moving Object Detection in 2D and 3D Scenes," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 6, pp. 577-589, June 1998.
- [30] M. Irani, P. Anandan, and S. Hsu, "Mosaic Based Representation of Video Sequences and Their Applications," *IEEE Proc. Int'l Conf. Computer Vision*, pp. 605-611, June 1995.
- [31] M. Irani, B. Rousso, and S. Peleg, "Detecting and Tracking Multiple Moving Objects Using Temporal Integration," *Proc. European Conf. Computer Vision*, pp. 282-287, May 1992.
- [32] T. Kanade, R.T. Collins, A.J. Lipton, P. Burt, and L. Wixon, "Advances in Cooperative Multi-Sensor Video Surveillance," *Proc. DARPA Image Understanding Workshop*, pp. 3-24, 1998.
- [33] A.J. Lipton, H. Fujiyoshi, and R.S. Patil, "Moving Target Classification and Tracking from Real Time Video," *Proc. Fourth IEEE Workshop Applications of Computer Vision '98*, pp. 8-14, 1998.
- [34] C. Morimoto and R. Chellappa, "Fast 3D Stabilization and Mosaic Construction," *IEEE Proc. Computer Vision and Pattern Recognition*, pp. 660-665, June 1997.
- [35] J.R. Muller, P. Anandan, and J.R. Bergen, "Adaptive-Complexity Registration of Images," *IEEE Proc. Computer Vision and Pattern Recognition*, pp. 953-957, 1994.
- [36] H.H. Nagel, "From Image Sequences Towards Conceptual Descriptions," *Image and Vision Computing*, vol. 6, no. 2, pp. 59-74, May 1988.
- [37] B. Neumann, *Semantic Structures: Advances in Natural Language Processing*, D.L. Waltz, ed. chapter 5, pp. 167-206, Hillsdale, N.J.: Lawrence Erlbaum, 1989.
- [38] S. Peleg and H. Rom, "Motion Based Segmentation," *IEEE Proc. Int'l Conf. Pattern Recognition*, vol. 1, pp. 109-113, 1990.
- [39] C. Pinhanez and A. Bobick, "Human Action Detection Using PNF Propagation of Temporal Constraints," *IEEE Proc. Computer Vision and Pattern Recognition*, June 1998.
- [40] D.B. Reid, "An Algorithm for Tracking Multiple Targets," *IEEE Trans. Automatic Control*, vol. 24, no. 6, pp. 843-854, Dec. 1979.
- [41] C. Schmid, R. Mohr, and C. Bauckhage, "Comparing and Evaluating Interest Points," *IEEE Proc. Int'l Conf. Computer Vision*, pp. 230-235, 1998.
- [42] T. Starner and A. Pentland, "Visual Recognition of American Sign Language Using Hidden Markov Models," *Proc. Int'l Workshop Automatic Face- and Gesture-Recognition*, 1995.
- [43] T. Strat, "Employing Contextual Information in Computer Vision," *Proc. DARPA Image Understanding Workshop*, pp. 217-229, 1993.



- [44] R. Szeliski, "Image Mosaicing for Tele-Reality Applications," *Proc. IEEE Workshop Applications of Computer Vision*, pp. 44-53, Dec. 1994.
- [45] R. Szeliski and H.Y. Shum, "Creating Full View Panoramic Image Mosaics and Environment Maps," *Proc. ACM SIGGRAPH '97*, Aug. 1997.
- [46] D. Wilson and A. Bobick, "Nonlinear PHMMs for the Interpretation of Parameterized Gesture," *IEEE Proc. Computer Vision and Pattern Recognition*, June 1998.
- [47] R. Zabih and J. Woodfill, "Non-Parametric Local Transforms for Computing Visual Correspondence," *Proc. European Conf. Computer Vision*, May 1994.
- [48] I. Zoghlami, O. Faugeras, and R. Deriche, "Using Geometric Corners to Build a 2D Mosaic from a Set of Images," *IEEE Proc. Computer Vision and Pattern Recognition*, pp. 420-425, June 1997.



**Gérard Medioni** received the Ingénieur diploma from the Ecole Nationale Supérieure des Télécommunications, Paris, France, in 1977, the MSc and PhD degrees in computer science from the University of Southern California in 1980 and 1983, respectively. Since 1983, he has been a faculty member of the Computer Science and Electrical Engineering Departments at the University of Southern California. He is a codirector of the Computer Vision Laboratory, Institute for

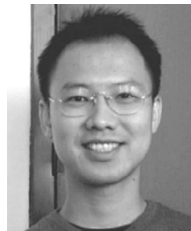
Robotics and Intelligent Systems (USC-IRIS) and a member of the Integrated Media Systems Center. He has held visiting positions at INRIA, France, and Geometrix, Inc. He has published a book, *A Computational Framework for Segmentation and Grouping*, and also numerous articles in many areas of computer vision. He serves as general cochair of IEEE Computer Vision and Pattern Recognition, 2001 in Kauai, Hawaii. He is a senior member of the IEEE and a member of the IEEE Computer Society.



**Isaac Cohen** received the MS degree in applied mathematics and automatics in 1989 and the PhD degree in applied mathematics in 1992 from the University Paris-IX-Dauphine, France. In 1992, he joined the Institut National de Recherche en Informatique et Automatique (INRIA) Rocquencourt, France, as a senior researcher. He worked on medical imaging in the EPIDAURE team and then joined the AIR team, where he worked on segmentation and analysis of environmental satellite image sequences. Since 1997, he has been with the Institute for Robotics and Intelligent Systems at the University of Southern California. His current research interests in computer vision are video segmentation, detection and tracking of moving objects, and spatiotemporal representations of video streams. He is a member of the IEEE and IEEE Computer Society.



**François Brémont** received the MS degree in computer science from the Ecole Normale Supérieure de Lyon (ENS-Lyon, France) in 1992 and the PhD degree in computer science from the Institut National de Recherche en Informatique et Automatique (INRIA-Sophia Antipolis, France) in 1997. From 1999 to January 2000, he was a research associate at the University of Southern California (USC), Los Angeles. In February 2000, he received a research position at INRIA and, since then, he has been working with ORION team. His primary research interests are in artificial intelligence and computer vision: knowledge-based systems, behavior analysis, context representation, uncertainty handling, image sequence processing, pattern recognition, object tracking, learning, and motion analysis.



activity understanding. He is a student member of the IEEE.

**Somboon Hongeng** received the BEng degree in electrical engineering from Tohoku University, Japan in 1992 and the MEng degree in electrical engineering from Nagoya University, Japan, in 1994. He is pursuing the PhD degree in computer engineering at the University of Southern California, Los Angeles. His research interests include computer vision, mathematical pattern recognition, statistical machine learning, artificial intelligence, and multimedia video-based human



**Ramakant Nevatia** received the BS degree from the University of Bombay, India, the MS, and PhD degrees from Stanford University, Palo Alto, California, all in electrical engineering. He has been with the the University of Southern California, Los Angeles, since 1975, where he is currently a professor of computer science and electrical engineering and the director of the Institute for Robotics and Intelligent Systems. He has authored two books: *Machine Perception and Computer Analysis of 3D Curved Objects* and contributed chapters to several others. He has been a regular contributor to the literature in computer vision. His research interests include computer vision, artificial intelligence, and robotics. Dr. Nevatia is a fellow of the American Association for Artificial Intelligence and a member of the ACM. He is an associate editor of *Pattern Recognition* and the *Computer Vision and Image Understanding*. He has served as an associate editor of the *IEEE Transactions on Pattern Analysis and Machine Intelligence* and as a technical editor in the areas of robot vision and inspection systems for the *IEEE Journal of Robotics and Automation*. He also served as a cogeneral chair of the IEEE Conference on Computer Vision and Pattern Recognition in June 1997. He is a fellow of the IEEE and member of the IEEE Computer Society.

► For further information on this or any computing topic, please visit our Digital Library at <http://computer.org/publications/dlib>.