HAL
open science

# Video Understanding for Complex Activity Recognition

Florent Fusier, Valery Valentin, François Bremond, Monique Thonnat, Mark Borg, David Thirde, James Ferryman

## To cite this version:

Florent Fusier, Valery Valentin, François Bremond, Monique Thonnat, Mark Borg, et al.. Video Understanding for Complex Activity Recognition. Machine Vision and Applications, 2007, 18, pp.167-188. inria-00276936

**HAL Id: inria-00276936**

**https://inria.hal.science/inria-00276936**

Submitted on 2 May 2008

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

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

**Florent Fusier · Valéry Valentin · François Brémond · Monique Thonnat · Mark Borg · David Thirde · James Ferryman**

# Video Understanding for Complex Activity Recognition

Received: date / Accepted: date

**Abstract** This paper presents a real-time video understanding system which automatically recognises activities occuring in environments observed through video surveillance cameras. Our approach consists in three main stages : Scene Tracking, Coherence Maintenance, and Scene Understanding. The main challenges are to provide a robust tracking process to be able to recognise events in outdoor and in real applications conditions, to allow the monitoring of a large scene through a camera network, and to automatically recognise complex events involving several actors interacting with each others. This approach has been validated for Airport Activity Monitoring in the framework of the European project AVITRACK.

Fusier, Valentin, Brémond and Thonnat

ORION Team

INRIA Sophia-Antipolis,

2004 Route des Lucioles BP 93

06902 SOPHIA ANTIPOLIS, FRANCE Tel.: +33 492387939

Fax: +33 492387939

E-mail: Francois.Bremond@sophia.inria.fr

Borg, Thirde and Ferryman

Computational Vision Group

The University Of Reading

Whiteknights, Reading, RG6 6AY, UK

**Keywords** Video Surveillance · Video Understanding · Scene Tracking · Activity Recognition · Airport Activity Monitoring

## 1 Introduction

Video understanding aims to automatically recognise activities occuring in a complex environment observed through video cameras. We are interested in understanding videos observed by surveillance CCTV camera network for the recognition of long-term activities involving mobile objects of different categories (e.g. people, aircraft, truck, cars). More precisely, our goal is to study the degree of complexity which can be handled with surveillance systems, and in particular to go beyond existing systems [**?**]. We are interested in complex scenes in terms of actors participating to the activities, large spatio-temporal scale and complex interactions. Video understanding requires several processing stages from pixel-based video stream analysis up to high level behaviour recognition. In this paper, we propose an original approach for video understanding. This approach can be divided into three main stages : Scene Tracking, Coherence Maintenance, and Scene Understanding (as shown on Figure 1). Scene Tracking consists first in detecting objects based on their motion, tracking them over time and categorising them (e.g. people, aircraft, truck, cars) per camera, then with a data fusion step in computing the 3D position of mobile objects in a global coordinate system. Coherence Maintenance consists in computing a comprehensive and coherent representation of the 3D scene together with the updating of its evolution in time. Scene Understanding consists in real-time recognition of video events according to end-user needs using coherent Scene Tracking results as input.

Each stage has to cope with specific problems in order to provide accurate and reliable data to the next stage. In this paper, we want more specifically to address three challenges. First, most of video surveillance systems [**?**] are still not able to run around the clock, in particular in outdoor environments. For instance, most of them are not able to handle the large variety of real application conditions such as snow, sunset, fog, night. Our goal is thus to propose a robust enough tracking process to be able to recognise events whatever the application conditions are. Second, even for operational video surveillance systems [**?**], it is still an issue to monitor a large scene with a camera network. Finally, a remaining issue for video surveillance systems is to automatically recognise complex events involving several actors with sophisticated temporal relationships.

In this paper, we want to address these issues with our video understanding approach and to validate it for Airport Activity Monitoring. This work has been undertaken in the framework of the European project AVITRACK. In this application, the aim is to perform real-time recognition of handling operations occuring around an aircraft parked on an apron area. The video sequences are provided by a network containing eight colour $720 \times 576$ resolution cameras with overlapping fields of view.

The first section deals with Scene Tracking, the second section deals with the coherence maintenance of the 3D dynamic scene and the third one describes in details Scene Understanding. In the last section, results are analysed in the Framework of the AVITRACK project.

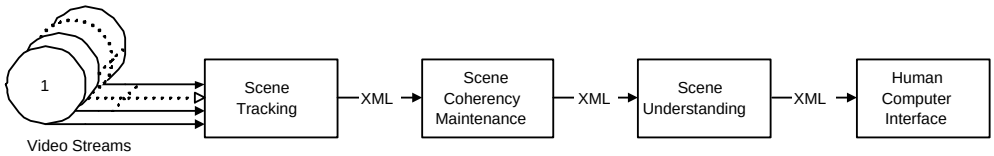



**Fig. 1** Video Understanding Overview. The main data stream starts from the videos up to human computer interface. In this system, no feedback has been setup due mainly to two reasons: the end-users requirement did not mention it and feedback has the tendency to slow down the whole process.

## 2 Scene Tracking

The AVITRACK Scene Tracking module comprises two distinct stages — per camera (2D) object tracking and centralised world (3D) object tracking. The per camera object tracking consists of motion detection (section 2.1) to find the moving objects in the observed scene, followed by object tracking in the image plane of the camera (section 2.2). The tracked objects are subsequently classified using a hierarchical object recognition scheme (section 2.3). The tracking results from the eight cameras are then sent to a central server where the multiple observations are fused into single estimates (section 2.4). In this Section we detail each step of the Scene Tracking module.

### 2.1 Motion Detection

Motion Detection segments the image into connected regions of foreground pixels that represent moving objects. These results are then used to track objects of interest across multiple frames. After evaluating several

motion detection algorithms for the airport apron environment (see results in [**?**]), the colour mean and variance [**?**] background subtraction algorithm was selected for AVITRACK. This algorithm uses a pixel-wise Gaussian distribution over the normalised RGB colour for modelling the background. This algorihtm was extended by including a shadow/highlight detection component based on the work of Horprasert *et al* [**?**] to make motion detection robust to illumination changes. In addition, a multi-layered background approach was adopted to allow the integration into the background of objects that become stationary for a short time period. More detail in [**?**].

### 2.2 Object Tracking

Real-time object tracking can be described as a correspondence problem, and involves finding which object in a video frame relates to which object in the next frame. Normally, the time interval between two successive frames is small, therefore inter-frame changes are limited, thus allowing the use of temporal constraints and object features to simplify the correspondence problem.

The Kanade-Lucas-Tomasi (KLT) feature tracking algorithm [**?**] is used for tracking objects in the AVITRACK system. This combines a local feature selection criterion with feature-based matching in adjacent frames. But the KLT algorithm considers features to be independent entities and tracks each of them individually. To move from the feature tracking level to the object tracking level, the KLT algorithm is incorporated into a higher-level tracking process: this groups features into objects, maintain associations between them, and uses the individual feature tracking results to track objects, while taking into account complex object interactions.

For each object $O$, a set of sparse features $S$ is maintained. $|S|$ — the number of features per object — is determined dynamically from the object's size and a configurable feature density parameter $\rho$:

$$|S| = \frac{\text{area}\,(O)}{|w|^2} \times \rho \tag{1}$$

where $|w|$ is the size of the feature's window ($9 \times 9$ pixels in our case). In experiments $\rho = 1.0$ i.e. $|S|$ is the maximal number of features that can spatially cover object $O$, without overlap between the local feature windows.

The KLT tracker takes as input the set of observations $\{M_j\}$ identified by the motion detector. Here, an observation $M_j$ is a connected component of foreground pixels, with the addition of a nearest neighbour spatial

filter of clustering radius $r_c$, i.e., connected components with gaps $\leq r_c$ are considered as one observation. Given such a set of observations $\left\{M_j^t\right\}$ at time $t$, and the set of tracked objects $\left\{O_i^{t-1}\right\}$ at $t-1$, the tracking process is summarised as:

1. Generate object predictions $\{P_i^t\}$ for time $t$ from the set of known objects $\left\{O_i^{t-1}\right\}$ at $t-1$, with the set of features $S_{P_i^t}$ set to $S_{O_i^{t-1}}$.
2. Run the KLT algorithm to individually track each local feature belonging to $S_{P_i^t}$ of each prediction.
3. Given a set of observations $\left\{M_j^t\right\}$ detected by the motion detector, match predictions $\{P_i^t\}$ to observations by determining to which observation $M_j^t$ the tracked local features of $P_i^t$ belong to.
4. Any remaining unmatched predictions in $\{P_i^t\}$ are marked as missing observations. Any remaining unmatched observations in $\left\{M_j^t\right\}$ are considered to be potential new objects.
5. Detect any matched predictions that have become temporarily stationary. These are integrated into the background model of the motion detector as a new background layer.
6. Update the state of those predictions in $\{P_i^t\}$ that were matched to observations and replace any lost features. The final result is a set of tracked objects $\{O_i^t\}$ at time $t$. Let $t = t+1$ and go to step 1.

In step 3 above, features are used in matching predictions to their corresponding observations to improve the tracking robustness in crowded scenes. This is achieved by analysing the spatial and motion information of the features. Spatial rule-based reasoning is applied to detect the presence of merging or splitting foreground regions; in the case of merged objects the motion of the individual features are robustly fitted to (predetermined) motion models to estimate the membership of features to objects. If the motion models are in distinct or unreliable then the local states of the features are used to update the global states of the merged objects. The spatial rule-based reasoning is described in more detail in Section 2.2.1, while the motion-based segmentation method is described in Section 2.2.2. Section 2.2.3 describes the technique in step 5 above, for detecting and handling moving objects that become temporarily stationary.

### 2.2.1 Using Spatial Information of Features

This method is based on the idea that if a feature belongs to object $O_i$ at time $t-1$, then the feature should remain spatially within the foreground region of $O_i$ at time $t$. A match function is defined which returns the number of tracked features $w$ of prediction $P_i^t$ that reside in the foreground region of observation $M_j^t$:

$$f\left(P_i^t, M_j^t\right) = \left|\left\{w : w \in S_{P_i^t}, w \in M_j^t\right\}\right| \tag{2}$$

In the case of an isolated (non-interacting) object, (2) should return a non-zero value for only one prediction-observation pair; ideally $f\left(P_i^t, M_j^t\right) = \left|S_{P_i^t}\right|$ – this is normally less due to lost and incorrectly-tracked features. For interacting objects, such as objects merging, occluding each other, undergoing splitting events, etc., a table of score values returned by (2) is constructed, and a rule-based approach is adopted to match predictions to observations.

The first rule handles the ideal matches of isolated objects, i.e. one-to-one matches between predictions and observations:

$$\begin{aligned} &f\left(P_i^t, M_j^t\right) > 0 \quad \text{and} \\ &f\left(P_k^t, M_j^t\right) = 0, \quad f\left(P_i^t, M_l^t\right) = 0 \quad \forall k \neq i, l \neq j \end{aligned} \tag{3}$$

The second rule handles the case when an object at time $t-1$ splits into several objects when seen at time $t$. This occurs when several observation regions match with a single prediction $P_i^t$ - in other words, the set of observations is partitioned into two subsets: the subset $M1$ of observations that match only with $P_i^t$ and the subset of those that do not match with $P_i^t$:

$$\begin{aligned} &f\left(P_i^t, M_j^t\right) > 0 \quad M_j^t \in M1 \subseteq M,\ |M1| > 1 \quad \text{and} \\ &f\left(P_k^t, M_j^t\right) = 0,\ \forall M_j^t \in M1, k \neq i \quad \text{and} \\ &f\left(P_i^t, M_l^t\right) \ = 0,\ \forall M_l^t \notin M1 \end{aligned} \tag{4}$$

The prediction is then split into new objects, one for each of the matched observations in $M1$. The features of the original prediction $P_i$ are assigned to the corresponding new object depending on whether they reside within its observation region or not. In this way, features are maintained throughout an object splitting event.

The third matching rule handles merging objects. This occurs when more than one prediction matches with an observation region:

$$\begin{aligned} &f\left(P_i^t, M_j^t\right) > 0 \quad P_i^t \in P1 \subseteq P,\ |P1| > 1 \quad \text{and} \\ &f\left(P_i^t, M_k^t\right) = 0,\ \forall P_i^t \in P1, k \neq j \quad \text{and} \\ &f\left(P_l^t, M_j^t\right) = 0,\ \forall P_l^t \notin P1 \end{aligned} \tag{5}$$

In this case the state of the predictions (such as position and bounding box) cannot be obtained by a straightforward update from the observation's state, since only one combined (merged) observation is available from the motion detector. Instead, the known local states of the tracked features are used to update the global states of the predictions. The prediction's new centre is estimated by taking the average relative motion of its local features from the previous frame at time $t-1$ to the current one. This is based on the assumption that the average relative motion of the features is approximately equal to the object's global motion - this may not always be true for non-rigid objects undergoing large motion, and may also be affected by the aperture problem due to the small size of the feature windows. The sizes of the bounding boxes of the predictions are also updated in order to maximise the coverage of the observation region by the combined predictions' bounding boxes. This handles cases where objects are moving towards the camera while in a merged state and hence their sizes increase. If not done, the result is parts of the observation region that are not explained by any of the predictions.

### *2.2.2 Using Motion Information of Features*

The motion information (per feature 2D motion vectors) obtained from tracking the local features of a prediction $P_i$ is also used in the matching process of step 3 above. Features belonging to an object should follow approximately the same motion (assuming rigid object motion). Motion models are fitted to each group of $k$ neighbouring features of $P_i$. These motion models are then represented as points in a motion parameter space and clustering is performed in this space to find the most significant motion(s) of the object [**?**]. A weighted list is maintained per object of these significant motions and the list is updated over time to reflect changes in the object's motion - if a motion model gains confidence its weight is increased; if a new motion model is detected, it is added to the list, or replaces an existing lower probable one. The motion models are used to differentiate the features of merged objects by checking whether a feature belongs to one motion model or

the other. This allows tracking through merging/occlusion and the replenishment of lost features. The motion models of an object are also used to identify object splitting events – if a secondary motion becomes significant enough and is present for a long time, splitting occurs. Although the underlying assumption is of rigid object motion, the use of a weighted list of motion models should allow for the identification of the different motions for articulated vehicles; future work will address this issue.

Two types of motion models have been used for AVITRACK – affine and translational models. The affine motion model is generated by solving for [**?**]:

$$w_t^T F w_{t-N} = 0 \tag{6}$$

where $w_t$ and $w_{t-N}$ are the (homogeneous) location vectors of feature $w$ at time $t$, $t-N$, and $F$ is the fundamental matrix representing the motion. For the affine case, $F$ has the form:

$$F = \begin{bmatrix} 0 & 0 & f_{13} \\ 0 & 0 & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \tag{7}$$

$F$ is obtained through a minimisation process based on eigen analysis, as described in [**?**]. The affine motion model is then represented in terms of 5 motion parameters: $v_{affine} = \langle \alpha, \gamma, \rho, \lambda, \theta \rangle$, where:

$$\alpha = arctan(\frac{-f_{13}}{f_{23}}) \tag{8}$$

$$\gamma = arctan(\frac{f_{31}}{-f_{32}}) \tag{9}$$

$$\rho = \sqrt{\frac{f_{31}^2 + f_{32}^2}{f_{13}^2 + f_{23}^2}} \tag{10}$$

$$\lambda = \frac{f_{33}}{\sqrt{f_{13}^2 + f_{23}^2}} \tag{11}$$

$$\theta = \alpha - \gamma \tag{12}$$

Clustering is performed in the motion parameter space to get the list of most significant motion models for the object.

The second motion model is simply the translational motion in the image plane:

$$v_{translational} = w_t - w_{t-N} \tag{13}$$

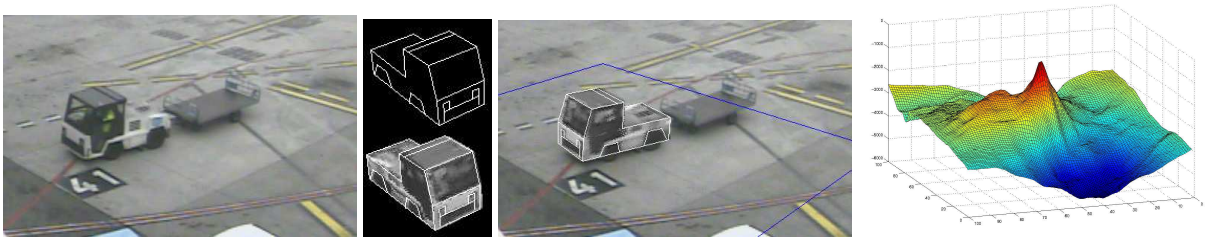
When tested on AVITRACK sequences, it was found that perspective and lens distortion effects cause the affine motion models to become highly dispersed in the motion parameter space and clustering performs

poorly. The translational model, as can be expected, also suffers from these problems and affine motion effects, but the effect on clustering is less severe. At present, the translational model is generally performing better than the affine model. Future work will look into improving the affine model and using perspective motion models.

### *2.2.3 Stationary Objects*

For the apron environment, activity tends to happen in congested areas near the aircraft with several vehicles arriving and stopping for short periods of time in the vicinity of the aircraft, creating occlusions and object merging problems. To allow objects to be differentiated and the tracking of moving objects in front of stopped objects, the motion detection process described in Section 2.1 was extended to include a multiple background layer technique. The tracker identifies stopped objects by one of two methods: by analysing an object's regions for connected components of foreground pixels which have been labelled as 'motion' for a certain time window; or by checking the individual motion of local features of an object. Stationary objects are integrated into the motion detector's background model as different background layers.

This technique is similar in idea to the temporal layers method described by Collins *et al* [**?**], except that their method works on a pixelwise level, using intensity transition profiles of pixels to classify them as 'stationary' or 'transient'. This is then combined with pixel clustering to form moving or stationary regions. This method performed poorly when applied to AVITRACK sequences, due mainly to stationary objects becoming fragmented into many layers as the duration objects remain stationary increases. This results in different update rates to the layers and incorrect re-activation once an object starts moving again. In the case of AVITRACK, the aircraft can remain stationary for up to half an hour - it is imperative that the object remains consistent throughout this time, its background layer gets updated uniformly and it is re-activated as a whole. The method adopted for AVITRACK works at the region-level and is handled by the tracker rather than at the motion detection phase, where the motion information of the local features can provide robust information on an object's motion. This use of region-level analysis helps to reduce the creation of a large number of background layers caused by noise.



(a) (b) (c) (d)

**Fig. 2** (a) Frame of sequence S21 showing a transporter vehicle. (b) Edge based and appearance based 3D model for the transporter vehicle. (c) The appearance model fitted to the vehicle, with the ground-plane (x,y) search area shown in blue. (d) x,y-slice of the evaluation score surface in the $(x, y, \theta)$ search space.

### 2.3 Object Recognition

To efficiently recognise the people and vehicles on the apron, a hierarchical approach is applied that comprises both bottom-up and top-down classification. The first stage categorises the top-level types of object that are expected to be found on the apron (people, ground vehicle, aircraft or equipment); this is achieved using a bottom-up Gaussian mixture model classifier trained on efficient descriptors such as 3D width, 3D height, dispersedness and aspect ratio. This was inspired by the work of Collins *et al* [**?**] where it was shown to work well for distinct object classes.

After the first coarse classification, the second stage of the classification is applied to the vehicle category to recognise the individual sub-types of vehicle. Such sub-types cannot be determined from simple descriptors and hence a proven method is used [**?**,**?**] to fit textured 3D models to the detected objects in the scene.

Detailed 3D appearance models were constructed for the vehicles and encoded using the 'facet model' description language introduced in [**?**]. The model fit at a particular world point is evaluated by back-projecting the 3D model into the image and performing normalised cross-correlation (NCC) of the facets' appearance model with the corresponding image locations. To find the best fit for a model, the SIMPLEX algorithm is used to find the pose with best score in the search space, assuming the model's movements are constrained to be on the ground-plane. See Figure 2 for an example. The initial pose of the 3D model $(x_0, y_0, \theta_0)$ used to initialise the search, is estimated from the centroid of the object (projected on to the ground-plane) and its direction of motion. The $x, y$ range in the search space is estimated from the image-plane bounding box of the object when projected on to the ground plane; while the $\theta$ search range is currently restricted to $\theta_0 + / - 15$ degrees.

The 3D model fitting algorithm is computationally intensive and cannot be run in real-time. This problem is solved by running the algorithm on a background (threaded) process to the main (bottom-up) tracking system and updating the object classification when it is available. A processing queue is used to synchronise the two methods together. For apron monitoring the sub-type category only becomes important when a vehicle enters specific spatial zones near the aircraft; the time between a vehicle entering the scene and entering such a zone is generally adequate to perform model-based categorisation at least once for each object. Running the classifier as a background process, means that the object location and orientation are measured for a *previous* frame, thus creating a latency in object localisation – this is a compromise required to achieve real-time performance. This problem is corrected in the Data Fusion module by applying an efficient object localisation strategy described in the following section.

## 2.4 Data Fusion

The method applied for data fusion is based on a discrete nearest neighbour Kalman filter approach [**?**] with a constant velocity model; the main challenge in apron monitoring relates to the matching of tracks to observations, this is not solved by a probabilistic filter, therefore the simpler deterministic filter is sufficient. The (synchronised) cameras are spatially registered using coplanar calibration [**?**] to define common 'world' co-ordinates $(x,y,z)$.

To localise objects in the world co-ordinates we devised an efficient strategy giving good performance over a wide range of conditions. For the person class of objects the location is taken to be the bottom-centre of the bounding box of the detected object. For vehicles we formulated a smooth function to estimate the position of the centroid using the measured (2-D) angle to the object. Taking $\alpha$ to be the angle measured between the camera and the object, the proportion $p$ of the vertical bounding box height (where $0 \leq p \leq 1/2$) was estimated as $p = 1/2(1 - \exp(-\lambda a))$; the parameter $\lambda$ was determined experimentally to provide good performance over a range of test data. The vertical estimate of the object location was therefore taken to be $y_{lo} + (p \times h)$ where $y_{lo}$ is the bottom edge of the bounding box and $h$ is the height of the bounding box. The horizontal estimate of the object location was measured as the horizontal centre-line of the bounding box, since this is generally a reasonable estimate [**?**].

The data association step associates existing track predictions with the per camera measurements. In the nearest neighbour filter the nearest match within a validation gate is determined to be the sole observation for a given camera. For multiple tracks viewed from multiple sensors the nearest neighbour filter is:

1. For each track, obtain the validated (i.e. associated) set of measurements per camera.
2. For each track, associate the nearest neighbour per camera.
3. Fuse associated measurements into a single measurements.
4. Kalman filter update of each track state with the fused measurement.
5. Inter-sensor association of remaining measurements to form candidate tracks.

The validated set of measurements are extracted using a validation gate [**?**]; this is applied to limit the potential matches between existing tracks and observations. In previous tracking work the gate generally represents the uncertainty in the spatial location of the object; in apron analysis this strategy often fails when large and small objects are interacting in close proximity on the congested apron, the uncertainty of the measurement is greater for larger objects hence using spatial proximity alone larger objects can often be mis-associated with the small tracks. To circumvent this problem we have extended the validation gate to incorporate velocity and category information, allowing greater discrimination when associating tracks and observations.

The observed measurement is a 7-D vector:

$$\mathbf{Z} = [x, y, \dot{x}, \dot{y}, P(p), P(v), P(a)]^T \tag{14}$$

where $P(\cdot)$ is the probability estimate that the object is one of three main taxonomic categories (p = Person, v = Vehicle, a = Aircraft) normalised such that $P(p) + P(v) + P(a) = 1$. This extended gate allows objects to be validated based on spatial location, motion and category, which improves the accuracy in congested apron regions. The effective volume of the gate is determined by a threshold $\tau$ on the normalised innovation squared distance between the predicted track states and the observed measurements:

$$d_k^2(i,j) = \left[\mathbf{H}\widehat{\mathbf{X}}_k^-(i) - \mathbf{Z}_k(j)\right]^T \mathbf{S}_k^{-1} \left[\mathbf{H}\widehat{\mathbf{X}}_k^-(i) - \mathbf{Z}_k(j)\right] \tag{15}$$

where $\mathbf{S}_k = \mathbf{H}\widehat{\mathbf{P}}_k^-(i)\mathbf{H}^T + \mathbf{R}_k(j)$ is the innovation covariance between the track and the measurement. This takes the form:

$$\mathbf{S}_k = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} & 0 & 0 & 0 & 0 & 0 \\ \sigma_{yx} & \sigma_y^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{\dot{x}}^2 & \sigma_{\dot{x}\dot{y}} & 0 & 0 & 0 \\ 0 & 0 & \sigma_{\dot{y}\dot{x}} & \sigma_{\dot{y}}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{P(p)}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{P(v)}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{P(a)}^2 \end{bmatrix} \tag{16}$$

where $\sigma_{P(p)}^2$, $\sigma_{P(v)}^2$ and $\sigma_{P(a)}^2$ are the innovation variance of the probability terms. For the kinematic terms the predicted state uncertainty $\widehat{\mathbf{P}}_k^-$ is taken from the Kalman filter and constant *a priori* estimates are used for the probability terms. Similarly, the measurement noise covariance $\mathbf{R}$ is estimated for the kinematic terms by propagating a nominal image plane uncertainty into the world co-ordinate system using the method presented in [**?**]. Measurement noise for the probability terms is determined *a priori*. An appropriate gate threshold can be determined from tables of the chi-square distribution [**?**].

Matched observations are combined to find the fused estimate of the object, this is achieved using *co-variance intersection*. This method estimates the fused uncertainty $\mathbf{R}_{fused}$ for $N$ matched observations as a weighted summation:

$$\mathbf{R}_{fused} = \left(w_1\mathbf{R}_1^{-1} + \ldots + w_N\mathbf{R}_{numcams}^{-1}\right)^{-1} \tag{17}$$

where $w_i = w_i'/\sum_{j=1}^{N} w_j'$ and $w_i' = \psi_i^c$ is the confidence of the $i$'th associated observation (made by camera $c$). The confidence value $\psi_i^c$ represents the certainty that the 2-D measurement represents the whole object. Localisation is generally inaccurate when clipping occurs at the left, bottom or right-hand image borders when objects enter/exit the scene. The confidence measure $\psi_i^c \in [0,1]$ is estimated using a linear ramp function at the image borders (with $\psi_i^c = 1$ representing 'confident' i.e. the object is unlikely to be clipped). A single confidence estimate $\psi_i^c$ for an object $O_i$ in camera $c$ is computed as a product over the processed bounding box edges for each object in that camera view.

If tracks are not associated using the extended validation gate the requirements are relaxed such that objects with inaccurate velocity or category measurements can still be associated. Remaining unassociated

measurements are fused into new tracks, using a validation gate between observations to constrain the association and fusion steps. Ghosts tracks without supporting observations are terminated after a predetermined period of time. To track objects that cannot be located on the ground plane we have extended the tracker to perform epipolar data association (based on the method presented in [**?**]).

## 3 Coherence Maintenance of the 3D Dynamic Scene

### 3.1 Introduction

High-level interpretation of video sequences emerges from the cooperation of a vision (Scene Tracking) process and a reasoning (Scene Understanding) process. The aim of coherencey maintenance is to perform a 3D analysis of mobile object dynamics in order to improve the Scene Tracking robustness. In particular the coherence maintenance process can handle complex scenes using large spatial and temporal scale to cope for instance with occlusions, or misdetections during several frames and changes of mobile object type. For instance, a task is to prevent mobile objects to disappear when no significant reason can justify it.
The coherence maintenance is performed through two tasks : Long-Term and Global Tracking. In the Long-Term Tracking, the temporal coherence of each individual mobile object is checked. In the Global Tracking, the spatio-temporal coherence of all the mobile objects is checked.

### 3.2 Long-Term Tracking

The aim of Long-Term Tracking is to use a temporal window to improve the tracking of detected mobile objects when the Frame-to-Frame Tracker face complex situations such as occlusions. This algorithm is based on temporal graph analysis. This algorithm has been previously applied to video communication [**?**] and it has been extended to address new issues for airport activity monitoring.

The proposed Long-Term Tracker takes as input the Data Fusion results. They consist in a list of tracked mobile objects located in a common 3D reference frame. Long-Term Tracking uses a temporal window (typically ten or twenty frames) in order to track mobile objects taking advantage of this temporal information to investigate potential object paths, in order to improve the tracking reliability and accuracy.

This algorithm uses both a predefined model of the expected objects in the scene and template models of the currently tracked objects. The predefined object models are used by the Long-Term Tracker to filter the mobile object types computed by previous modules (Frame-to-Frame and Data Fusion). Typically, a predefined model of an expected object is defined by its 3D dimensions (mean values and variances), its motion model (speed and ability to change direction) and its usual location zones defined in the contextual knowledge (see section 4.2). Currently, three different models are used : Person, Vehicle and Aircraft. The template model of a tracked object contains its current dynamics (speed, direction) and appearence (2D and 3D size and color histogram).

The algorithm first computes all the possible paths related to the detected objects. A path represents a possible trajectory of one mobile object (e.g. a person, a vehicle, or an aircraft) inside the temporal window. A path is then composed of a sequence of mobile objects detected at each frame during the temporal interval. A path $P_i$ represents a possible trajectory of one tracked object in the scene during the temporal interval $[t_f - T; t_f]$, where $t_f$ is the time of the last (current) image processed and $T$ is the size of the temporal window used to analyse paths. $P_i$ is composed of a temporal sequence of mobile objects $M_i(t), t \in [t_f - T; t_f]$, fulfilling two conditions:

- $(a_1)\ \forall\, t\ \exists!\ M_i(t); M_i(t) \in P_i$
- $(a_2)\ dist_{2D}(M_i(t); M_i(t+1)) \leq D_{max}$.

To reduce the exploration space of all the possible paths related to a mobile object (and avoid combinatorial explosion), three constraints are used. The first one is the type coherence between each mobile object of the same path (to avoid the type of a mobile object to be too different from the other mobile objects) , the second one is the spatio-temporal coherence between the different mobile objects in the path (to avoid too large displacements) and the third one is the motion model of the predefined objects.

Then, the algorithm sorts out the path set and optimise the associations between possible paths and tracked mobile objects. The exploration of multiple possible paths allows to deal with spontaneous mobile object type changes and with misdetection of mobile objects during a short period of time (less than the temporal window size). For instance, when a path is generated and if no more mobile objects are detected for this path, virtual

mobile objects are created in order to continue the tracking, over the temporal window. This algorithm is appropriate when a mobile object is not detected and when it reappears several frames later.

## 3.3 Global Tracking

Although the Frame-to-Frame and Long-Term Trackers are efficient, several limitations remain which result in sending incoherent information to the Scene Understanding module. These limitations include:

- loss of tracked objects due to occlusions (static or dynamic occlusion) or due to the integration of a mobile object in the background image after a long stationary period,
- remaining over or under detections due to severe shadows or lack of contrast,
- mix of tracked object identity when several objects are merging or crossing each other.

To cope with these limitations, a high level module called Global Tracker has been added. This module is in charge of improving the data computed by the Long-Term Tracker in order to provide coherent data to the Scene Understanding process. For this purpose, the Global Tracker uses also as input data the a priori knowledge of the observed environment (static and dynamic contextual information, see section 4.2).

### *3.3.1 Principle*

The aim of the Global Tracker is to correct the detected mobile objects wrongly tracked by the previous processes using 3D spatio-temporal analysis. The Global Tracker uses a general algorithm which uses a set of methods (i.e. implementing rules) for its adaptation to each specific application. The Global Tracker is initialised by the methods to be applied which are ordered according to their priority. Each method can have one or several parameters (e.g. type of objects involved in the method) to correspond to different situations. For instance, a method that keeps track of parked vehicles even if they are integrated into the background has been designed and implemented taking into account different parking durations.

In airport activity monitoring application, a method is needed to handle two types of parked vehicles : the Tanker and Ground Power Unit (G.P.U.). This is needed because they are usually parked during half an hour for the Tanker and about ten minutes for the G.P.U. By defining a generic prototype for a method, the architecture of the Global Tracker allows developpers to easily add new rules to solve specific problems to a given

application.This generic prototype follows the model:

if *conditions(mobile object type, position, size, direction)* then *corrections(remove mobile object, merge mobile objects, merge trajectories)*

The Global Tracker has been successfully applied to building access control, bank agencies, and airport activity monitoring applications, showing its genericity. We illustrate the Global Tracker process through two examples.

### *3.3.2 Loss of tracked objects*

First, when a vehicle has stayed for a long time at the same place, the update of the background image (called also reference image) tends to integrate the vehicle into the background, resulting in a misdetection and a loss of this vehicle. This situation usually occurs while a vehicle is parked in specific zones, for instance when the Tanker vehicle is parked below the wing of the aircraft during a refuelling operation.

To cope with this limitation, a method has been defined to keep track of a vehicle when it is parked in specific zones and lost by previous modules. This method checks if a vehicle disappear in an inappropriate zone. This method ends in two cases : when the vehicle restarts (the method then links the newly detected vehicle with the lost one) or after a predefined period of time (the method definitely deletes the vehicle track).

### *3.3.3 Over detection of mobile objects*

Second, another problem occurs when a single vehicle is wrongly detected as several mobile objects. This situation appears for instance when different parts of a vehicle are detected as different individual mobile objects (as shown on Figure 3 (b)). In particular, this is the case when a vehicle is operating: starting and stopping several times in a short spatial and temporal interval. To solve this shortcoming, a method has been added to merge all the wrongly detected mobile objects into the real vehicle. This method takes into account the predefined vehicle models, the 3D position and the motion of the tracked mobile objects. To be merged, mobile objects should have the same motion (minus a threshold error), should be in the same neighborhood, and the resulting vehicle should match with one of the predefined vehicle models (as illustrated on Figure 3 (b)).

(a) (b)

**Fig. 3** (a) Before the Global Tracker, a Loader vehicle is detected as several mobile objects (over detection). (b)After the Global Tracker, the Loader vehicle is correctly tracked as one mobile object.

The Global Tracker approach is effective when an error is particularly representative of a typical problem and can be explained by logical formulas. However, when an error rarely occurs, no correcting method is added to prevent further errors. Globally, the utilisation of both a Long-Term and a Global Tracker has enhanced the system perfomance and has allowed to track on a large scale simultaneously people, vehicles and aircraft interacting with each others on the apron area. Thus, the Scene Understanding process is able to recognise activities in more complex situations.

## 4 Scene Understanding

### 4.1 Introduction

The aim of Scene Understanding is to provide a high level interpretation of the tracked mobile objects trajectories in term of human behaviours, vehicle activities, or their interactions. This process consists in detecting video events which have been learned through examples or predefined by application experts.

Scene Understanding has been a main problem of focus in Cognitive Vision for the last decade. There are now many research units and companies defining new approaches to design systems that can understand human activities in dynamic scenes. Two main categories of approaches are used to recognise video events based on (1) a probabilistic/neural network, or on (2) a symbolic network corresponding to the events to be recognised.

For the computer vision community, a natural approach consists in using a probabilistic/neural network. The nodes of this network correspond usually to video events that are recognised at a given instant with a computed probability. For instance, Howell and Buxton [**?**] proposed an approach to recognise a video event based on a neural network (time delay Radial Basis Function). Hongeng *et al* [**?**] proposed a video event recognition method that uses concurrence Bayesian threads to estimate the likelihood of potential events. These methods are efficient to recognise short events that can be frequently observed with the same visual characteristics in the representative video database in order to obtain an efficient learning phase. However, they fail to recognise complex events (e.g. involving several actors) lasting a long time period.

For the artificial intelligence community, a natural way to recognise a video event is to use a symbolic network which nodes correspond usually to the boolean recognition of video events. For instance, Gerber *et al* [**?**] defined a method to recognise a video event based on a fuzzy temporal logic. Pinhanez and Bobick [**?**] have used Allen's interval algebra to represent video events and have presented a specific algorithm to reduce its complexity. Shet *et al* [**?**] have recognised activities based on Prolog rules.

A traditionnal approach consists in using a declarative representation of video events defined as a set of spatio-temporal and logical constraints. For instance, Rota and Thonnat [**?**] have used a constraint resolution technique to recognise video events. This method recognises a video event by searching in the set of previously recognised video events a set of components (sub video events) matching the video event model to be recognised. To reduce the processing time for the recognition step, they are checking the consistency of the constraint network using the AC4 algorithm. However, this method processes temporal constraints and atemporal constraints in the same way without taking advantage of the temporal dimension. Thus, if the system fails to recognise a video event, it will have to retry the same process (reverify the same constraints) at the next instant, implying a costly processing time. A second problem is that this algorithm has to store and maintain all occurrences of previously recognised video events.

Another approach consists in using a symbolic network and to store partially recognised video events (expected to be recognised in the future). For instance, Ghallab [**?**] has used the terminology chronicle to express a video event. A chronicle is represented as a set of temporal constraints on time-stamped events. The recognition algorithm keeps and updates partial recognition of video events that can be recognised in the future, using the propagation of temporal constraints based on RETE algorithm. This method has been

applied to the control of turbines and telephonic networks. Chleq and Thonnat [**?**] made an adaptation of this temporal constraints propagation for video surveillance. This method recognises a scenario by predicting the expected video events to be recognised at the next instants. Thus, the video events have to be bounded in time to avoid the never ending expected events. A second problem is that this method has to store and maintain all occurrences of partially recognised video events, implying a costly storing space.

All these techniques allow an efficient recognition of video events, but there are still some temporal constraints which cannot be processed. For example, most of these approaches require that the video events are bounded in time (Ghallab [**?**]).

Vu *et al* [**?**] have combined the previous approaches to optimise the temporal constraint resolution by ordering in time the components of the video events to be recognised. This method aims to reduce the processing time (1) when searching in the past (list of previously recognised video events) for an occurrence of a given video event model and (2) when trying to recognise a video event involving several actors by avoiding checking all combinations of actors.

We have extended this last method to address Complex Activity recognition involving several physical objects of different types (e.g. individuals, ground vehicles of different types, aircrafts) over a large space observed by a camera network and over an extended period of time.
So we propose a method to recognise video events based on spatio-temporal reasoning taking full advantage of a priori knowledge about the observed environment and of video event models. To define video events, a modeling language called Video Event Description Language has been designed so that application domain experts can easily model off-line the activities they are interested in. Then at video frame rate, the Video Event Recognition algorithm is able to process the given video streams to recognise the predefined video event models.

### 4.2 Contextual Knowledge about the Observed Environment

Contextual knowledge of the observed environment is all the a priori information about the empty scene. This is the necessary information which allows to give the meaning of the activities happening in the scene. Automatic interpretation needs contextual information because each observed environment is specific and each activity changes according to the contextual environment where it takes place. Only general video events

corresponding to generic behaviours such as walking or entering an area (for human activities) do not need contextual information. We have distinguished two kinds of contextual knowledge : static and dynamic.

### *4.2.1 Static Contextual Knowledge*

The static contextual knowledge corresponds to all the information relative to non moving objects and to the 3D empty scene. The 3D empty scene information contains both geometric and semantic description of the specific zones and the equipment located in the observed environment.
The static contextual knowledge is structured into six parts :

- a list of 3D referentials where different referentials are defined to manage distant areas composing the scene. For each 3D referential, the calibration matrices and the position of the video cameras.
- a list of ground planes to describe the different levels where mobile objects objects are evolving,
- a list of geometric static zones corresponding to the different zones of interest in the observed environment,
- a list of abstract areas corresponding to one or several static zones. These areas contain the expected classes of mobile objects (e.g. in a pedestrian area, only persons are expected),
- a list of walls to describe for instance airport walls,
- a list of equipment associated with its characteristics.

The geometric description of a zone of interest contains a polygon defined in a plane. The geometric description of a static object (e.g. a piece of equipment) corresponds to a generalised cylinder defined by its height and its polygonal basis. The semantic description of a zone or a contextual object contains six attributes with both symbolic or numerical values : its type (equipment or area), its function (e.g. wall, entrance zone, door), its name, its characteristics (e.g. blue, fragile), its usual distance and time for interacting with the static object.

In Airport Apron Environment, contextual information corresponds to an apron area (as illustrated on Figure 4). An apron area is basically an empty area containing zones of interest related to specific operation functionnalities. These zones are mainly waiting or parking zones for the different vehicles expected on the apron. Zones of interest represent the static contextual information, which is usually relevant enough to perform video understanding. However, in Airport Activity Monitoring application, more dynamic information is required.

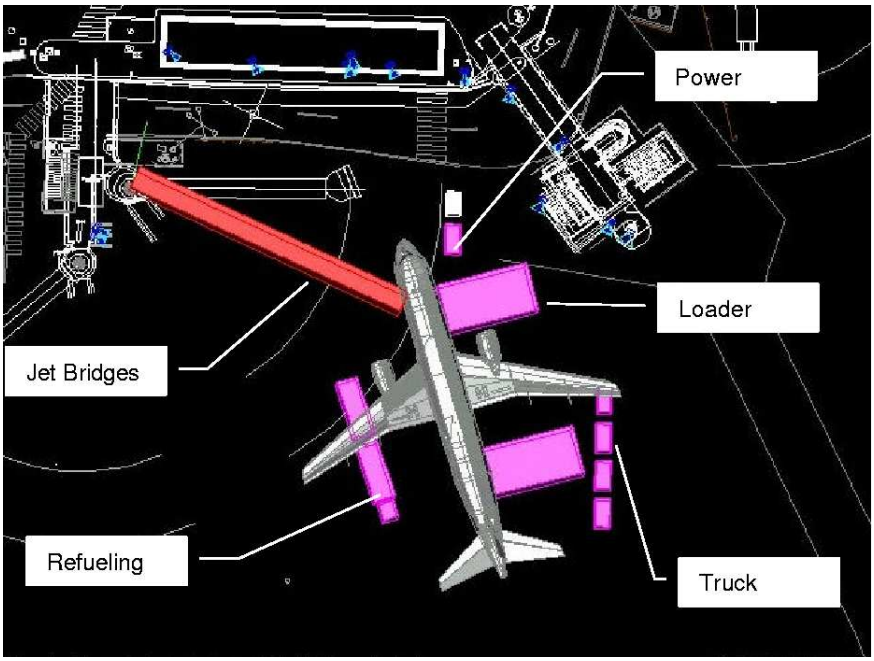



**Fig. 4** Overview of the airport apron area showing the main servicing operations.

### *4.2.2 Dynamic Contextual Knowledge*

The dynamic contextual knowledge gathered all zones of interest relative to vehicles which can interact with other vehicles or people. This knowledge is needed to recognise activities involving the vehicles when they are parked. Dynamic vehicle interaction zones are added to the static context only when the vehicles stay for a while in their respective parking zone and are removed when the vehicles leave. Given that vehicles do not stop each time exactly at the same place, the zones of interest related to vehicles are defined in the vehicle referential. For instance, a dynamic zone can been used for a vehicle door to allow the detection of the driver exiting the vehicle.

In Airport Activity Monitoring, zones of interest of ground service vehicles and of aircraft have been dynamically added to the static contextual knowledge. Interaction zones of the aircraft corresponds to:

- its input/output zones, such as back or front doors (used to load or unload baggages)
- specific areas (e.g. the Refuelling area where the Tanker is parked during the aircraft refuelling operation, the jetbridge area where the jetbridge is parked to allow the crew boarding and unboarding)

Dynamic contextual knowledge is required for the recognition of people and vehicles interacting with each others.

### 4.3 Video Event Modelisation

We have defined a representation formalism to help the experts to describe the video events of interest occuring in the observed scene. This formalism contains a language called Video Event Description Language which is both declarative and intuitive (in natural terms) so that the experts of the application domain can easily define and modify the video event models.

Four different types of video events have been designed. The first distinction lies on the temporal aspect of video events : we distinguish states and events. A state describes a situation characterising one or several physical objects at time *t*, or a stable situation over a time interval. An event describes an activity containing at least a change of state values between two consecutive times. The second distinction lies on the complexity aspect : a state/event can be primitive or composite. The video events are organised into four categories:

- primitive state : a visual property directly computed by the Scene Tracking process (e.g. a vehicle is located inside a zone of interest or a vehicle is stopped)
- composite state : a combination of primitive states (e.g. a vehicle is located and stopped inside a zone of interest, as shown on example below)
- primitive event : a change of values for a primitive state (e.g. a vehicle enters a zone of interest)
- composite event : a combination of primitive states/events (e.g a vehicle is parked for a while inside a zone of interest and then leaves it)

**CompositeState**(Vehicle_Stopped_Inside_Zone,
**PhysicalObjects**((v1 : Vehicle), (z1 : Zone))
**Components**((c1 : PrimitiveState Inside_Zone(v1,z1))
(c2 : PrimitiveState VehicleStopped(v1)))
**Constraints**((c2 during c1)))

This composite state enables to recognise that a vehicle *v1* is stopped inside a zone of interest *z1*. It is composed of two primitive states *c1* and *c2*. A temporal constraint *during* must be satisfied for the recognition of the composite state.

Taking into consideration these video event types, a general video event model has been designed. This video model for representing a given video event $E$ contains 5 parts :

- a set of Physical Object variables corresponding to the physical objects involved in $E$ , such as contextual objects including static objects (equipment, zone of interest) and mobile objects (person, vehicle, aircraft). The vehicles can even be specified by different subtypes to represent different vehicles (e.g. GPU, Loader, Tanker, Jet-Bridge)
- a set of component variables corresponding to the sub-events of $E$.
- a set of forbidden component variables corresponding to the events that are not allowed to occur during the detection of $E$.
- a set of constraints (symbolic, logic, spatial and temporal constraints including Allen's interval algebra operators [**?**]) related to previous variables.
- a set of actions corresponding to the tasks predefined by experts that need to be executed when the event $E$ is recognised (e.g. activating an alarm or displaying a warning message).

## 4.4 Video Event Recognition

### *4.4.1 Challenges*

The automatic recognition of activities is a real challenge for Cognitive Vision research because it addresses the recognition of complex activities involving several physical objects of different types (e.g. individuals, groups of people, and vehicles) and even subtypes (e.g. different types of vehicles). The challenge is to perform a real-time video event recognition algorithm able to efficiently recognise all the video events occuring in the scene at each instant. The performance of the proposed approach lies in a temporal video event recognition method which avoids combinatorial explosion and keeps linear complexity using temporal constraint resolution.

### *4.4.2 Algorithm of Video Event Recognition Overview*

The proposed Video Event Recognition algorithm and its advantages compared to the State of The Art are described in detail in Vu *et al* [**?**].

The proposed video event recognition process is able to recognise which video events are occuring in a video stream at each instant. To benefit from all the knowledge, the video event recognition process uses the coherent tracked mobile objects, the a priori knowledge of the scene (static and dynamic contextual information) and the predefined video event models. To be efficient, the recognition algorithm processes in specific ways video events depending on their type. Moreover, this algorithm has also a specific process to search previously recognised video events to optimise the whole recognition. The algorithm is composed of two main stages. First, at each step, it computes all possible primitive states related to all mobile objects present in the scene. Second, it computes all possible events (i.e. primitive events then composite states and events) that may end with the recognised primitive states.

*Recognition of primitive states*

To recognise a primitive state, the recognition algorithm performs a loop of two operations :

1. the selection of a set of physical objects then
2. the verification of the corresponding atemporal constraints until all combinations of physical objects have been tested.

Once a set of physical objects satisfies all the atemporal constraints, the primitive state is recognised. To enhance primitive event recognition, after a primitive state has been recognised, event templates (called triggers) are generated for each primitive event the last component of which corresponds to the recognised primitive state. The event template contains the list of the physical objects involved in the primitive state.

*Recognition of primitive events*

To recognise a primitive event, given the event template partially instantiated, the recognition algorithm consists in looking backward in the past for a previously recognised primitive state matching the first component of the event model. If these two recognised components verify the event model constraints, the primitive event is recognised.

To enhance the composite event recognition, after a primitive event has been recognised, event templates are generated for all composite events the last component of which corresponds to the recognised primitive event.

*Recognition of composite states and events*

The recognition of composed states and events usually implies a large space search composed of all the possible combinations of components and physical objects. To avoid a combinatorial explosion, all composed states and events are decomposed into states and events composed at the most of two components through a stage of compilation in a preprocessing phase. Then the recognition of composed states and events is performed similarly to the recognition of primitive events. This is an efficient method to cope with classical combinatorial explosion problems.

To recognise the predefined events models at each instant, we first select a set of scenario templates (called triggers) that indicate which scenarios can be recognised. These templates correspond to an elementary video event (primitive state or event) or to a composite video event that terminates with a component recognised at the previous or current instant.

For each of these video event templates, solutions are found by looking for components instances already recognised in the past to complete the video event template. A solution of a video event model $\omega$ is a set of physical objects that are involved in the recognised video event and the list of corresponding component instances satisfying all the constraints of $\omega$.

*The algorithm for Video Event Recognition*

We define a ”trigger” as a video event template which can be potentially recognised. There are three types of triggers : the primitive video event models (type 1), the composite video events with specified physical objects (type 2) and the composite video events already recognised at the previous instant (type 3).

Overview of the Algorithm :

**For each** primitive state model

Create a trigger T of type 1 for the primitive event model

**For each** solution $\rho_e$ of T

**If** $\rho_e$ is not extensible **Then**

Add $\rho_e$ to the list of recognised video events

Add all triggers of type 2 of $\rho_e$ to the list LT (List of Triggers)

**If** $\rho_e$ is extensible with $\rho'_e$ recognised at previous instant **Then**

Merge $\rho_e$ with video event $\rho'_e$

Add all triggers of type 2 and 3 of $\rho'_e$ to the list LT

**While** (LT $\neq \varnothing$)

Order LT by the inclusive relation of video event models

**For each** trigger $T_0 \in$ LT

**For each** solution $\rho_0$ of $T_0$

Add $\rho_0$ to the list of recognised video events

Add all triggers of type 2 and 3 of $\rho_0$ to the list LT

At the current instant, we initiate a list LT of triggers with all triggers of first type (i.e. primitive video event models). Once we have recognised a primitive video event $\rho_e$, we try to extend $\rho_e$ with a recognised video event $\rho'_e$ at the previous instant (the extension of a video event is the extension of its ending time). If $\rho_e$ cannot be extended, we add the triggers of type 2 that terminate with $\rho_e$ to the list LT. If $\rho_e$ is extended with $\rho'_e$, we add the triggers of type 2 and 3 that terminates with $\rho'_e$. The triggers of type 2 are the templates of a composite video event instantiated with the physical objects of $\rho'_e$ and the triggers of type 3 are the templates of a composite video event $\rho_0$ already recognised at the previous instant and that terminates with $\rho'_e$. After this step, there is a loop process first to order the list LT by the inclusive relation of video event model contained in the triggers and second to solve the triggers of LT. If a trigger contains a template of a video event $\rho'_0$ that can be solved (i.e. totally instantiated), we add the triggers of type 2 and 3 that terminate with $\rho'_0$.

*Finding a solution for a Video Event Model*

The algorithm for finding a solution for a video event template (trigger) consists in a loop of (1) selecting a set of physical objects then of (2) verifying the corresponding constraints until all combinations of physical objects have been tested. This selection of physical objects leads the recognition algorithm to an exponential combination in function of the number of physical object variables. However, in practice, there are few physical object variables in video event models, so the recognition algorithm can still be real time. More-

over, only for primitive states all combination of physical objects need to be checked. For a other video event models, some physical object variables are already instantiated through the template which has triggered its recognition.

Once the physical objects have been selected, we check all atemporal constraints. These atemporal constraints are ordered with the occurrence order of the physical objects variables (in a compilation phase) to speed up the recognition process. If the video event is a primitive one, after the verification of its atemporal constraints, the video event is said to be recognised. If the video event is a composite one, its temporal constraints still need to be verified. To verify the temporal constraints of a composite video event model, we extract from the set of recognised video event instances the components satisfying the constraints defined in the video event model. To optimise the searching process, the components are ordered in time. Once we find a solution of a video event model, we store the recognised video event instance and we add to the list of triggers LT the template terminating with this video event. To avoid redoing at each instant previous successfull recognition of video events, the video event models are compiled into two component video event models during a pre-processing step.

This proposed recognition algorithm is described in detail in Vu *et al* [**?**] and has been validated first in real world conditions (i.e. with a large set of complex video events described by aeronautical experts) with the Airport Activity Monitoring application.

## 5 Video Understanding for Airport Activity Monitoring

We have validated the whole video understanding approach with an Airport Activity Monitoring application in the framework of the European Project AVITRACK. To evaluate the approach, we have focus on the evaluation of four characteristics of the proposed system. The system has demonstrated its abilities to : (1) model airport servicing operations, (2) recognise a large diversity of events, (3) recognise complex events and (4) run reliabilly through everyday airport conditions.

### 5.1 Modeling of Airport Servicing Operations

The aim of Scene Understanding in Airport Apron Environment is to automatically recognise activities around a parked aircraft in an airport apron area for shortening airplane shift time and security purposes. This goal

means to recognise in real-time the on-going operations which are processed on the apron at each instant, involving both people and vehicles interacting with each others. These activities consist of operations of growing complexity from simple operations such as vehicle arrivals on the apron area (involving a single vehicle) to more complex activities such as baggages unloading (involving several vehicles and people interacting with each others). Operations have been modeled using aeronautical experts knowledge. Thanks to the video event description language, the experts have defined more than fifty video events which have been built on the set of generic video events.

Exploitation of automatic activity recognition represents two main issues for end-users (e.g. handling companies, airport management, airline companies) : safety and security. Safety consists in checking the handling operations rules such as vehicle speed limit, and security deals with the detection of abnormal behaviours around a parked aircraft. Scene Understanding can also be used in cost-control and quality-control issues in generating automatically information about processed operations such as activity reports, and in apron allocation and ground traffic control knowing which operations are on-going on the apron ground.

To help the experts to define the video event models, we have designed a set of general video events to be used to build composite events. Currently this set is composed of twenty-one video events :

- ten primitive states : (e.g. a person is located inside a zone of interest, a vehicle is stopped)
- five composite states (e.g. a vehicle is stopped inside a zone of interest : composition of two primitive states stop and located inside a specific zone)
- six primitive events (e.g. a vehicle enters a zone of interest or a person changes from one zone to another)

The flexibility of the video event description language has been demonstrated through the modeling of a large quantity of activities in different application domains such as metro station monitoring [**?**], bank agency monitoring [**?**], inside-train monitoring, car parking monitoring and now in airport apron monitoring. This diversity of application represents a good feedback from end-users to evaluate how this language is expressive and easy to use.

### 5.2 Recognition of a Large Diversity of Events

End-users such as companies which handle apron activities are interested in detecting vehicle arrivals in order to know that these vehicles are arrived and are ready to play their role in the activities. That explains that the

first focus was on activity recognition such as vehicle arrivals or activities involving few vehicle or people. In every operation occuring on the apron area, observation of video sequences shown that a general scenario was repeated : an operative vehicle first enters in the ERA zone (the apron, the Entrance Restricted Area), then enters in its access zone and then park in this zone. For a given vehicle, operations or interactions with other ground vehicles or people start only when it has stopped and is parked.

The first modeled activities are arrivals of the following vehicles :

- the Ground Power Unit, involved in the aircraft arrival preparation,
- the Tanker, involved in the aircraft refuelling operation,
- the Loader, involved in the baggages loading and unloading operations,
- the Transporter, involved in the baggages transport via containers,
- the Tow Tractor, involved in the aircraft tow before its departure,
- the Conveyor, involved in the back door baggages unloading operation,
- the Aircraft, when it drives to its parking position.

As previously seen, the way a vehicle is involved in an operation is always the same : the vehicle arrival occurs first and then when parked, the vehicle operates or interacts with other ground vehicles or workers on the apron. From vehicle arrivals observations on numerous relevant video sequences has been extracted a general model to recognise a vehicle arrival. An arrival actually occurs as a three stages process : first the vehicle arrives in the apron (the Entrance Restricted Area), then it drives to its access zone, and finally stops there before any interaction begins. Arrivals thus represent three scenarios per vehicle (as shown with the Loader arrival, described below).

Using the general video event models for the vehicle arrival models, new video events have been modeled. For instance, one composite video event using four general video events has been defined to recognise the aircraft arrival preparation involving the G.P.U vehicle (as shown below). This operation involves six physical objects : the G.P.U. vehicle, its driver and four zones of interest. The system automatically recognises that the G.P.U. vehicle arrives in the E.R.A. zone, then enters in its access area, and then stop in it (this recognises the G.P.U. arrival). Then the drivers is recognised as exiting the G.P.U. (this is one application of dynamic contextual information) and then he deposits chocks and stud at the location where the aircraft is expected to park.

**CompositeEvent**(Aircraft_Arrival_Preparation,

**PhysicalObject**((p1 : Person), (v1 : Vehicle), (z1 : Zone),

(z2 : Zone), (z3 : Zone), (z4 : Zone))

**Components**((c1 : CompositeState Gpu_Arrived_In_ERA(v1,z1))

(c2 : CompositeEvent Gpu_Enters_Gpu_Access_Area(v1,z2)))

(c3 : CompositeState Gpu_Stopped_In_Gpu_Access_Area(v1,z2)))

(c4 : CompositeState Handler_Gets_Out_Gpu(p1,v1,z2,z3)))

(c5 : CompositeEvent Handler_From_Gpu_Deposits_Chocks_Or_Stud(p1,v1,z2,z3,z4)))

**Constraints**((v1− >Type = ”*GPU*”)

(z1− >Name = ”*ERA*”)

(z2− >Name = ”*GPU_Access*”)

(z3− >Name = ”*GPU_Door*”)

(z4− >Name = ”*Arrival_Preparation*”)

(c1 before c2)

(c2 before c3)

(c3 before c4)

(c4 before c5)

(c4 during c3)

(c5 during c3)))

This composite event enables to recognise the Aircraft Arrival Preparation. It involves six physical objects : one person *p1*, one vehicle *v1* and four zones of interest *z1*, *z2*, *z3* and *z4*. It is composed of five components *c1*, *c2*, *c3*, *c4* and *c5* to recognise that the GPU has arrived in the ERA zone, enters in its area and then stops in it. Then the driver exits the GPU (*c4*) and deposites the chocks to prepare the aircraft arrival (*c5*). Constraints are used to specify the vehicle type (a GPU), the zones of interest *z1*, *z2*, *z3*, *z4* and to verify temporal relationships between the components.

**CompositeEvent**(Loader_Arrival,

**PhysicalObjects**((v1 : Vehicle), (z1 : Zone), (z2 : Zone))

**Components**((c1 : CompositeState Vehicle_Arrived_In_ERA(v1,z1))

(c2 : CompositeEvent Loader_Enters_FrontLoading_Area(v1,z2)))

(c3 : CompositeState Loader_Stopped_In_FrontLoading_Area(v1,z2)))

**Constraints**((v1− >SubType = ”*LOADER*”)

(z1− >Name = ”*ERA*”)

(z2− >Name = ”*Front_Loading_Area*”)

(c1 before c2)

(c2 before c3)

This composite event enables to recognise the Loader vehicle arrival. It involves three physical objects : one vehicle *v1* and two zones of interest *z1* and *z2*. It is composed of three components *c1*, *c2* and *c3* to recognise that the Loader has arrived in the ERA zone, enters in its area and then stopped in it. Constraints are used to specify the vehicle type (a Loader), the zones of interest *z1* and *z2* and to verify temporal relationships between the components (*before*).

## 5.3 Recognition of Complex Activities

*Complex Activity*

After initial work on simple activity recognition to show the large diversity of events which can be addressed, the next challenge was to handle automatic recognition of complex activities. Complex activities refers to the operations involving several mobile objects (which can be of different types and even subtypes) interacting with each others during an extended time period, on the apron area, observed through the camera network. Such activities are described using video event models involving several physical objects, components (even composite states or events) representing the different stages of the operations, and temporal constraints. The proposed video event representation language allows to simplify the modeling of such operations by using composite video events embedding others composite video events. This modelisation eases the writting of the video event models because this can be done progressively from primitive to more and more complex composite video events. Temporal constraints enable to manage the temporal relationships between the components of composite events. As previously described, the proposed recognition algorithm uses composite video events decomposition and preprocessing phase to efficiently cope with combinatorial explosions that

could occur during the recognition of complex activity.

**CompositeEvent**(Unloading_Operation,

**PhysicalObjects**((p1 : Person), (v1 : Vehicle), (v2 : Vehicle), (v3 : Vehicle),

(z1 : Zone), (z2 : Zone), (z3 : Zone), (z4 : Zone))

**Components**((c1 : CompositeEvent Loader_Arrival(v1,z1,z2))

(c2 : CompositeEvent Transporter_Arrival(v2,z1,z3)))

(c3 : CompositeState Worker_Manipulating_Container(p1,v3,v2,z3,z4)))

**Constraints**((v1− >SubType = ”*LOADER*”)

(v2− >SubType = ”*TRANSPORTER*”)

(z1− >Name = ”*ERA*”)

(z2− >Name = ”*Front_Loading_Area*”)

(z3− >Name = ”*Transporter_Area*”)

(z4− >Name = ”*Container_Worker_Area*”)

(c1 before_meet c2)

(c2 before_meet c3)

This composite event enables to recognise the Unloading operation. It involves eight physical objects : one person *p1*, three vehicles *v1*, *v2*, *v3*, and three zones of interest *z1*, *z2*, and *z3*. It is composed of three components *c1*, *c2*, *c3* to recognise the Loader arrival (a composite event previously detailed), the Transporter arrival, and the worker manipulating containers. Eight constraints are used to specify the vehicle types (Loader and Transporter, the third vehicle corresponds to a container), the zones of interest, and the temporal relationships bewteen the components (*before_meet*).

*Baggages Front Unloading Operation*

In order to illustrate how is modeled an activity, the modeling of the baggages front unloading operation is described.

This usual operation occurs when the aircraft has parked and consists of unloading the passengers baggages through the aircraft front right door. This operation is well structured and procedural. Several physical objects are involved : the Loader and Transporter vehicles, a container, the Loader driver and a ground worker. These physical objects are interacting with each others through four zones of interest. After observation of

video sequences, this operation appears to be a three main stages process. Thus the modeling of this operation is designed in three parts : the Loader arrival, the Transporter arrival, and the containers manipulation. These three parts are also composed of other video events (as shown for instance for the Loader arrival). The operation of ”baggages Unloading” is composed of different steps corresponding to events involving the Loader, the Transporter and the worker person manipulating the baggage containers. First, the Loader vehicle arrives in the ERA zone, then it enters in its restricted area and then parked in this zone. This is the composite event ”Loader_Arrival”. The Loader handler is detected when he opens the right front loading door (event ”Loader_Handler_Detected”). When the Loader has parked, the Transporter then enters and parks in order to allow the baggage containers reception from the Loader. This represents the ”Transporter Arrival”. When the Transporter has parked in its area, it means that it is ready to receive the baggage containers from the Loader in the Worker area zone. This zone corresponds to the place where containers from the Loader are manipulated by the worker person who then attach it to the Transporter. Here we recognise that the worker has arrived in the zone and is ready to get the containers (event “Worker_arrived”), and then when a container is detected in the unloading zone with the worker, the event ”Worker_Manipulating_Container” (illustrated on Figure 5) is recognised. As previously mentioned, the steps which compose the operation are also composite events and are composed of other events. For instance, the “Loader_Arrival” is composed of three combined video events : (1) the “Vehicle_Arrived_In_ERA”, the “Loader_Enters_FrontLoading_Area”, and the “Loader_Stopped_In_FrontLoading_Area” events. The Loader arrival is consequently recognised only when these three events have been recognised, according to the temporal constraints they are related to.

## 5.4 Reliability of Video Understanding

### *5.4.1 Scene Tracking*

The Scene Tracking evaluation assesses the performance of the core tracking components (motion detection, object tracking, object recognition and data fusion) on representative test data.

The detection rate ($TP/(TP+FN)$) and false alarm rate ($FP/(TP+FP)$) metrics were chosen to quantitatively evaluate the motion detection and object tracking performance (where TP, FN and FP are the number of true positives, false negatives and false positives respectively).

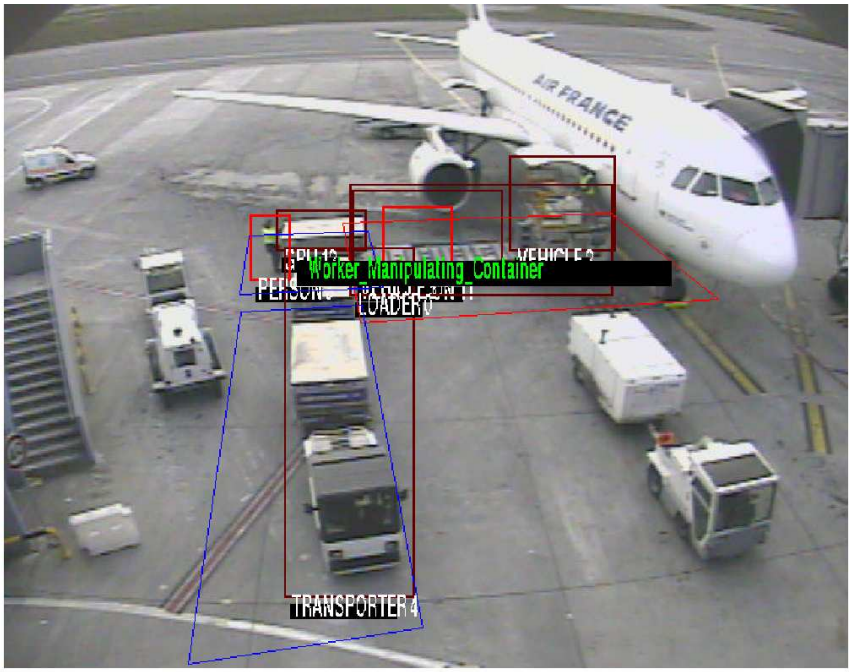



**Fig. 5** This Composite state is recognised when the apron worker is manipulating a container in the working zone. This is recognised when the worker has arrived in this zone and that a container is detected in it. This enables to recognise that they are interacting with each others.

The performance evaluation of the different motion detector algorithms for AVITRACK is described in more detail in [**?**]. The performance of the colour mean and variance motion detector was evaluated on three apron datasets. Dataset 1 (9148 frames) contains the presence of fog whereas datasets 2 and 3 (6023 frames) are acquired on a sunny day. For datasets 1 and 3, the motion detector provides a detection rate of 77% and a false negative rate of 23%. In dataset 2 the detection rate decreases to 60%. The achromatic nature of the scene generates a considerable number of false negatives causing the decrease in detection rate and the increase in false negative rate. The fog in dataset 1 causes a high number of foreground pixels to be misclassified as highlighted background pixels resulting in a decrease in accuracy (93%). A representative result for the motion detection is shown in Figure 6. It can be seen that some objects are partially detected due to the similarity in appearance between the background and foreground objects.

The performance evaluation of the per-camera tracking algorithm is described in more detail in [**?**]. The Scene Tracking evaluation assesses the performance on representative test data containing challenging conditions for an objective evaluation. Two test sequences were chosen, dataset 4 (a subset of dataset 1, 2400

**Fig. 6** Representative motion detection result from dataset 1 showing (Left) reference image and (Right) detection result for the colour mean and variance algrotihm.

frames) contains the presence of fog whereas dataset 5 (1200 frames) was acquired on a sunny day. Both sequences contain typical apron scenes with congested areas containing multiple interacting objects.

For Dataset 4 3435 true positives, 275 false positives and 536 false negatives were detected by the KLT based tracker. This leads to a tracker detection rate of 0.87 and a false alarm rate of 0.07. For Dataset 5 3021 true positives, 588 false positives and 108 false negatives were detected by the KLT based tracker. This leads to a tracker detection rate of 0.97 and a false alarm rate of 0.16. Representative results of the scene tracking module are presented in Figure 7. It can be seen that strong shadows are tracked as part of the objects such as the tanker from Dataset 4 and the transporter from Dataset 5. In Dataset 4 a person (bottom-right of scene) leaves the ground power unit and in Dataset 5 a container is unloaded from the aircraft, both these scenarios leave a ghost track in the previous object position.

The evaluation of the hierarchical object recognition module is described in more detail in [**?**]. It was found that errors occured in the first stage of classification when the bottom-up features were not well detected, therefore the descriptors were no longer representative of the object type. Type categorisation accuracy found to be between 60 and 97% for evaluated test sequences. The type classification was found to be sensitive to the detection result. Sub-Type categorisation accuracy found to be between 61 and 88% for evaluated test sequences. The sub-type classification was found to be sensitive to similarity in vehicle appearance and local maxima in the evaluation score surface.

**Fig. 7** (Left) Results obtained from the scene tracking module showing (Left) Dataset 4 and (Right) Dataset 5. (Right) Result obtained from the data fusion module.

The Data Fusion module is qualitatively evaluated for an extended sequence (S21) of 9100 frames. The data fusion performance is shown in Figure 8 where estimated objects on the ground plane are shown for the test sequence. It is clear to see that the extended data fusion module out-performs a standard (i.e. spatial validation and fusion) data fusion process. This is achieved by extending the validation gate to more features and fusing objects based on the measurement confidence. Many more objects estimated by the extended data fusion are contiguous, with less fragmentation and more robust matching between measurements and existing tracks. For many scenarios the extension of the validation gate provides much greater stability, especially when objects are interacting in close proximity. The use of object confidence in the fusion process also improves the stability of the tracking when objects enter/exit the cameras fields-of-view. It is noted that the track identity can be lost when the object motion is not well modelled by the Kalman filter or when tracks are associated with spurious measurements.

### *5.4.2 Scene Understanding*

To validate globally the video understanding approach (i.e. Scene Tracking, Coherence Maintenance and Scene Understanding), we have tested it on various Airport Activity video sequences. First, we have tested the approach for the recognition of airport activities involving one vehicle, then we have tested it for activities with interactions with several vehicles.

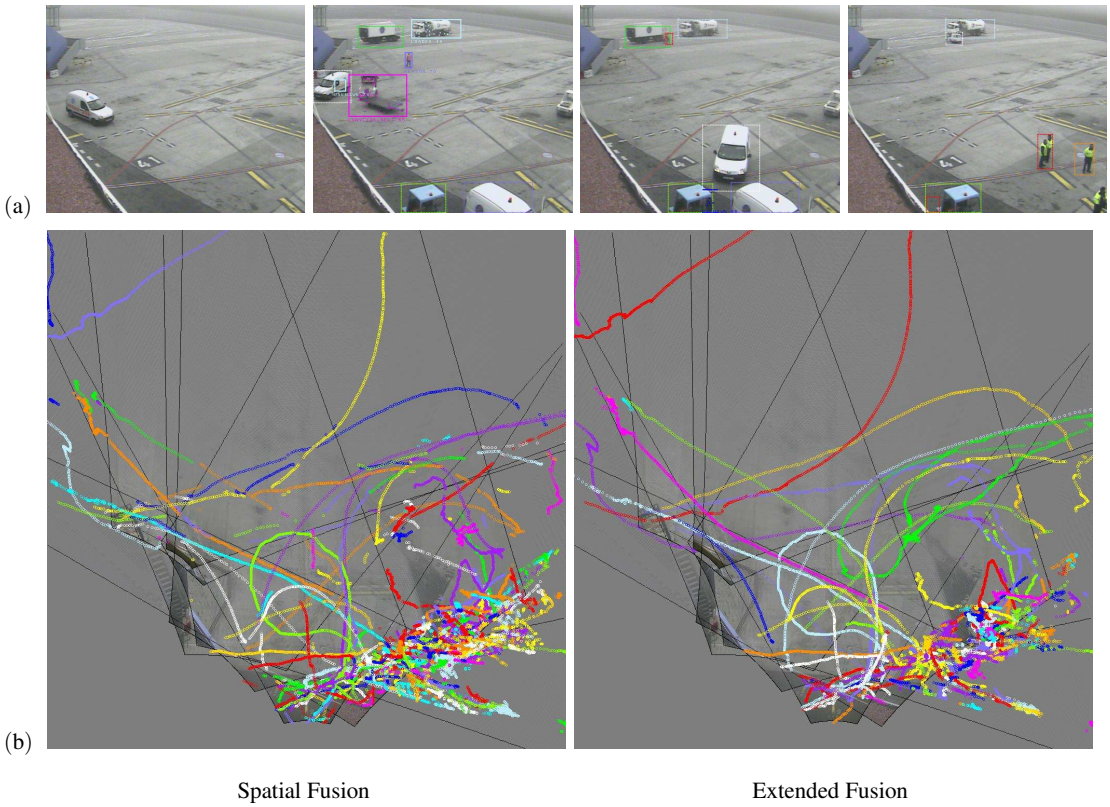



**Fig. 8** Results of the data fusion module showing tracked object locations on the ground-plane for two representative data sets. The track colour is derived from the object ID, limited to eight colours for visualisation. (a) Per-camera object tracking results for sequence S21 - All cameras frames 0, 6000, 7000, 9000. (b) Objects tracked by the data fusion module with (Extended Fusion) and without (Spatial Fusion) the extended validation gate and confidence based fusion.

*Single Vehicle Activity Recognition.*

We have tested the recognition of activities involving a single vehicle and/or a single person on twenty two video sequences for a total duration of about four hours and half. More precisely, the following activities have been modeled and recognised :

- "Aircraft Arrival Preparation" : containing 8 video events, describing the Ground Power Unit vehicle arrival and the deposit by a worker of the chucks and cons before the aircraft arrival. This operation has been recognised on 10 video sequences with duration of about ten minutes.
- "Refuelling Operation", 8 video events, involving the Tanker vehicle and one person refuelling the aircraft. This operation has been recognised on seven sequences of twenty minutes duration.
- "Aircraft Arrival", 3 video events, describing the Aircraft arriving at its parking position. This operation has been recognised on two sequences of about ten minutes.

- "Tow Tractor Arrival", 3 video events, describing the arrival of the Tow Tractor vehicle before towing the aircraft before its departure. This operation has been recognised on two sequences of about five minutes.
- "Conveyor Arrival", 3 video events, describing the Conveyor Belt vehicle arrival before the backdoor baggage unloading. This operation has been recognised on one sequence.

*Multi Vehicle Activity Recognition.*

We have tested the recognition of activities involving several vehicles and people on two video sequences for a total duration of twenty minutes. More precisely, the following activities have been modeled and recognised :

- "Unloading global operation", involving the Loader vehicle (3 video events), the Transporter vehicle (3 video events), and three zones of interest. This operation has been recognised on two sequences of about ten minutes.
- "Unloading detailed operation", involving the Loader vehicle (3 video events), the Transporter vehicle (3 video events), one person, one container (2 video events) and four zones of interest. This operation has been recognised on two sequences of about ten minutes.

So the system we propose currently recognises 58 video events : 21 general video events, 25 single vehicle video events (8 involving the GPU, 8 involving the Tanker, 3 involving the Aircraft, 3 involving the Tow Tractor, 3 involving the Conveyor), and 12 multi vehicle video events involving both the Loader and the Transporter. This experimentation has been done during regular operation conditions on test sets representative of usual servicing operations selected by end users.

The recognition of these video events and their complexity demonstrate both the system **effectiveness** and **flexibility**. Thanks to this project, we have shown that automatic video system can monitor an airport apron during normal utilisation conditions.

## 6 Conclusion and Future Work

We have proposed a video understanding system which perfoms a real-time activity recognition on video sequences following three stages : tracking, coherence maintenance and understanding. The main challenges are to provide a robust tracking process to be able to recognise events in outdoor and in real application con-

ditions, to allow the monitoring of a large scene through a camera network, and to automatically recognise complex events involving several actors interacting with each others.

The proposed system has been evaluated with an Airport Activity Monitoring application during normal utilisation conditions, demonstrating both the effectiveness and robustness of the approach. The Scene Tracking results have shown its reliability on a large number of video sequences. Remaining tracking errors are handled by the coherence maintenance stage. The modeling of airport servicing operations has shown the flexibility and the expressiveness of the video event representation formalism. Finally the Scene Understanding results have shown its ability to deal with complex activity, recognising more than fifty types of events. Thus, the proposed system has successfully handled both Cognitive Vision and Airport Activity Monitoring challenges. Next steps will consist in assessing in live conditions the system in the airport to validate its robustness whatever the weather conditions are. Future work will also tackle the automatic setup of the system to obtain a convenient solution which can be easy installed in a large variety of critical areas, such as other airports, but also train stations, harbours and every place where automated visual surveillance can provide more security.

## References

1. Aguilera, J., Wildernauer, H., Kampel, M., Borg, M., Thirde, D., Ferryman, J.: Evaluation of motion segmentation quality for aircraft activity surveillances. In: Proc. Joint IEEE Int. Workshop on VS-PETS, Beijing (2005)
2. Allen, J.: Maintaining knowledge about temporal intervals. In: Communications of the ACM, pp. 823–843 (1983)
3. Avanzi, A., Bremond, F., Thonnat, M.: Tracking multiple individuals for video communication. In: Proc. of IEEE International Conference on Image Processing (2001)
4. Bar-Shalom, Y., Li, X.: Multitarget-Multisensor Tracking: Principles and Techniques. YBS Publishing (1995)
5. Black, J., Ellis, T.: Multi Camera Image Measurement and Correspondence. In: Measurement - Journal of the International Measurement Confederation, vol. 35 num 1, pp. 61–71 (2002)
6. Chleq, N., Thonnat, M.: Real-time image sequence interpretation for video-surveillance applications. In: Proc. IEEE International conference on Image Processing, vol. Vol 2, pp. 801–804 (1996)
7. Collins, R., Lipton, A., 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. In: Tech. Report CMU-RI-TR-00-12 (2002)
8. Cupillard, F., Avanzi, A., Bremond, F., Thonnat, M.: Video understanding for metro surveillance. In: Proc. of IEEE ICNSC 2004 in the special session on Intelligent Transportation Systems (2004)
9. Ferryman, J.M., Worrall, A.D., Maybank, S.J.: Learning enhanced 3d models for vehicle tracking. In: Proc. of the British Machine Vision Conference (1998)

10. Georis, B., Maziere, M., Bremond, F., Thonnat, M.: A video interpretation platform applied to bank agency monitoring. In: Proc. of IDSS'04 - 2nd Workshop on Intelligent Distributed Surveillance Systems (2004)
11. Ghallab., M.: On chronicles: Representation, on-line recognition and learning. In: Proc. of 5th International Conference on Principles of Knowledge Representation and Reasoning (KR'96) (1996)
12. Horprasert, T., Harwood, D., Davis, L.: A statistical approach for real-time robust background subtraction and shadow detection. In: IEEE ICCV'99 FRAME-RATE Workshop, Kerkyra (1999)
13. Howell, A., Buxton, H.: Active vision techniques for visually mediated interaction. In: Image and Vision Computing (2002)
14. Khoudour, L., Hindmarsh, J., Aubert, D., Velastin, S., Heath, C.: Enhancing security management in public transport using automatic incident detection. In: L. Sucharov and C. Brebbia, editors, Urban Transport VII : Proc. of the 7th International Conference on Urban Transport of the Environment for the 21st Century. WIT Press., pp. 619–628 (2001)
15. Lo, B., Velastin, S.: Automatic congestion detection system for underground platforms. In: International Symposium on Intelligent Multimedia, Video and Speech Processing, pp. 823–843 (2001)
16. Piater, J., Richetto, S., Crowley, J.: Event-based activity analysis in live video using a generic object tracker. In: Proc. of the 3rd IEEE Workshop on Performance Evaluation of Tracking and Surveillance (2002)
17. Pinhanez, C., Bobick., A.: Human action detection using pnf propagation of temporal constraints. In: M.I.T. Media Laboratory Perceptual Section Report, vol. No. 423 (1997)
18. R. Gerber, H.N., Schreiber., H.: Deriving textual descriptions of road traffic queues from video sequences. In: The 15-th European Conference on Artificial Intelligence (ECAI'2002) (2002)
19. Rota, N., Thonnat, M.: Activity recognition from video sequences using declarative models. In: Proc. of 14th European Conference on Artificial Intelligence (ECAI 2000) (2000)
20. S. Hongeng, F.B., Nevatia, R.: Representation and optimal recognition of human activities. In: In IEEE Proc. of Computer Vision and Pattern Recognition (2000)
21. Shet, V., Harwood, D., David, L.: Vidmap : Video monitoring of activity with prolog. In: Proc. of IEEE International Conference on Advanced Video and Signal based Surveillance, pp. 224–229 (2005)
22. Shi, J., Tomasi, C.: Good features to track. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition, pp. 593–600 (1994)
23. Sullivan, G.D.: Visual interpretation of known objects in constrained scenes. In: Phil. Trans. R. Soc. Lon., vol. B, 337, pp. 361–370 (1992)
24. Thirde, D., Borg, M., J.Aguilera, Ferryman, J., Baker, K., Kampel, M.: Evaluation of object tracking for aircraft activity surveillance. In: Proc. Joint IEEE Int. Workshop on VS-PETS. Beijing (2005)
25. Thirde, D., Borg, M., Valentin, V., Barthélémy, L., Aguilera, J., Fernandez, G., Ferryman, J., cois Brémond, F., Thonnat, M., Kampel, M.: People and vehicle tracking for visual surveillance. In: Sixth IEEE International Workshop on Visual Surveillance 2006. Graz, Austria (2006)
26. Thirde, D., Borg, M., Valentin, V., Fusier, F., J.Aguilera, Ferryman, J., Brémond, F., Thonnat, M., M.Kampel: Visual surveillance for aircraft activity monitoring. In: Proc. Joint IEEE Int. Workshop on VS-PETS. Beijing (2005)

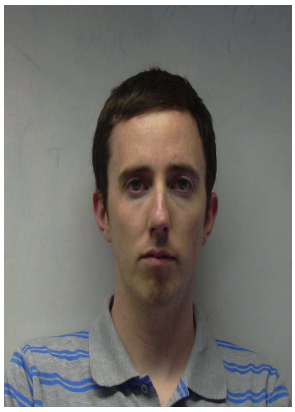
27. Tsai, R.: An efficient and accurate camera calibration technique for 3d machine vision. In: Proc. CVPR, pp. 323–344 (1986)
28. Vu, V., Bremond, F., Thonnat, M.: Automatic video interpretation: A novel algorithm for temporal scenario recognition. In: Proc. of the Eighteenth International Joint Conference on Artificial Intelligence, pp. 1295–1300 (2003)
29. Wren, C.R., Azarbayejani, A., Darrell, T., Pentland, A.: Pfinder: Real-time tracking of the human body. In: IEEE Transactions on PAMI, vol. 19 num 7, pp. 780–785 (1997)
30. Xu, G., Zhang, Z.: Epipolar Geometry in Stereo, Motion and Object Recognition: A Unified Approach. Kluwer Academic Publ. (1996)

**Valéry Valentin** Valéry Valentin is an R&D Engineer in the Orion team at INRIA Sophia Antipolis since . He obtained his Master degree in Image-Vision in 2001 at STIC Doctoral School, Univ. of Nice-Sophia Antipolis.

**François Brémond** François Brémond is a researcher in the ORION team at INRIA Sophia Antipolis. He obtained his Master degree in 1992 at ENS Lyon. He has conducted research works in video understanding since 1993 both at Sophia-Antipolis and at USC (University of Southern California), LA. In 1997 he obtained his PhD degree at INRIA in video understanding and pursued his research work as a post doctorate at USC on the interpretation of videos taken from UAV (Unmanned Airborne Vehicle) in DARPA project VSAM (Visual Surveillance and Activity Monitoring). He also has participated to three European projects (PASSWORD, ADVISOR, AVITRACK), one DARPA project, five industrial research contracts and several international cooperations (USA, Taiwan, UK, Belgium) in video understanding.

**Monique Thonnat** Monique Thonnat is a senior research scientist at INRIA (Director of Research 1rst class). She is the head of Orion a research group on cognitive vision at INRIA in Sophia Antipolis, France. She received in 1982 a PhD degree in Optics and Signal Processing from University of Marseille. In 1983 she joined INRIA in Sophia Antipolis. In 1991 she became Director of Research and in 1995 she created Orion, a multi-disciplinary research team at the frontier of computer vision, knowledge-based systems, and software engineering. Monique Thonnat is author or co-author of more than 100 scientific papers published in international journals or conferences; she has supervised 17 PhD theses. Her more recent research activities involve the conception of new techniques for the reuse of programs and on image understanding techniques for the interpretation of video sequences.

**James Ferryman** Dr Ferrymans research interests include model-based methods for people and traffic surveillance, human-computer interaction, robotics and autonomous systems, and ”smart” cameras. Dr Ferryman was investigator on two EC Framework V proposals: ADVISOR (IST-1999-11287) on people tracking in metro stations, and ANFAS on modeling flood risks (IST-1999-11676), and the EU Framework VI Aero project AVITRACK which focused on the automated visual surveillance of airport aprons. Dr Ferryman is co-chair of the IEEE International Workshops on Performance Evaluation of Tracking and Surveillance in 2000-2004, and is a reviewer for the EU Sixth Framework IST Programme. He is currently a co-investgator of the UK EPSRC project REASON on the robust robust monitoring of people in public spaces, the UK EPSRC network ViTAB (Video-based Threat Assessment and Biometrics) and is a Principal Investigator for the EU FP6 Aero project: SAFEE which addresses on-board security.



**David Thirde** David Thirde received his B.Eng. in Electrical and Electronic Engineering from Loughborough University in 2000. He is undertaking a Ph.D. degree with the Digital Imaging Research Centre at Kingst on University, where he was funded until 2003. From 2003 to 2004 he worked as a researcher on the EU project INMOVE. Upon completion of this project he moved to the Computational Vision Group at the University of Reading where he has worked on the EU projects AVITRACK and ISCAPS. His research interests include high-level interpretation of video and autonomous surveillance systems.

**Mark Borg** Mark Borg received a B.Sc. in Mathematics and Computer Science from the University of Malta in October 1995, and an M.Sc. in Engineering and Information Sciences from the University of Reading in 2003. In 2004, Mark joined the Computational Vision Group of the University of Reading where he worked as a research assistant in the area of automated visual surveillance and tracking, in particu lar participating in the EU AVITRACK project. Starting in 2006, Mark returned back to industry and is currently working in the R&D group of Crimsonwing developing new e-commerce solutions, as well as providing freelance consultancy services.