ASEV - Automatic Situation Assessment for Event-driven Video Analysis

Michele Fenzi, Jörn Ostermann
Institute for Information Processing (TNT)
Leibniz University Hannover
fenzi,ostermann@tnt.uni-hannover.de

Nico Mentzer, Guillermo Payá-Vayá, Holger Blume
Institute for Microelectronic Systems (IMS)
Leibniz University Hannover
mentzer,blume@ims.uni-hannover.de

Tu Ngoc Nguyen, Thomas Risse
L3S Research Center
Leibniz University Hannover
tunguyen,risse@l3s.de

Abstract

Many complex maneuvers involving aircraft, vehicles and persons are carried out at airport aprons. Manual video surveillance used for safety and security purposes is inefficient and privacy protection must be guaranteed. In this paper, we propose a system named ASEV that automatically assesses situations for airport surveillance. It combines four main components: a low-level image processing unit based on a new hardware implementation to extract features in real time, a high-level image processing unit for scene analysis, a real-time inference engine for scene understanding, and a data protection stage for log encryption. In addition, four often neglected aspects are successfully addressed: two-way communication between system and operator, power consumption, monitored people privacy and operator activity control. Extensive evaluation at a real airport shows that the proposed system improves the operator performance with sound and visual alerts based on the automatic assessment of various events.

1. Introduction

Video surveillance of outdoor airport premises is aimed at safety and security. Different, complex operations are manually monitored: aircraft parking, goods loading and unloading, passengers boarding and deboarding, etc. However, manual video surveillance is notoriously inefficient: operators only watch one screen at a time, events are overlooked as nothing occurs for long periods of time, flight schedules are often not taken into account, etc. Furthermore, monitoring affects the privacy of employees and passengers, all the more when personal information, such as pictures, are retained and stored. Finally, the conflict of interest in the activities of the operator, who is himself an airport employee, must be addressed.

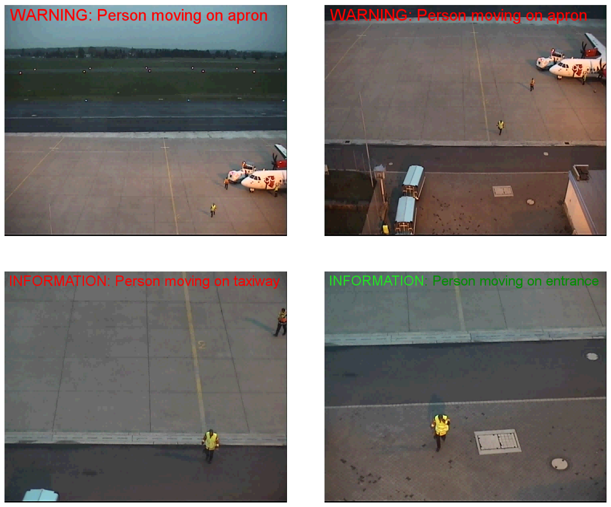

WARNING: Person moving on apron
WARNING: Person moving on apron
INFORMATION: Person moving on taxiway
INFORMATION: Person moving on entrance

Figure 1: An airport employee is detected and tracked in several areas. The system assists the operator with on screen messages assessing the current situation.

Many individual solutions and several systems have been recently devised to introduce automation in airport monitoring [1, 2]. In addition to scene analysis, these systems focus on one or more additional aspects, *e.g.*, improving security against aircraft collisions, detecting servicing activities, etc.

Four aspects of considerable interest in many situations are often neglected in the design of monitoring systems:

- Two-way channel between system and operator.
- Power consumption for distributed systems.
- Privacy of the monitored people.
- Control of operator activities.

In most systems, the operator cannot provide any feedback to the system about the correctness of the received messages [21]. Thus, the system always assesses similar situations in the same fashion. Moreover, distributed systems are commonly chosen if large areas are to be covered. As the processing load is distributed on each of the many sensors, power consumption is a vital aspect to consider. Finally, most systems can neither guarantee the privacy of the monitored, as pictures and videos are permanently stored, nor keep track of the operator activity, in case he tries to manipulate both current and stored information.

In this paper, we propose an effective and flexible system named ASEV that, by combining four different modules, automatically assess events occurring on the apron and successfully addresses the aforementioned issues.

The first component is a high-level multithreading image processing unit, which employs state-of-the-art feature-based algorithms to perform object tracking, recognition and classification tasks. The second module is a real-time ontology-based inference engine, that analyzes the output of the high-level unit by comparing it to a set of rules. The resulting messages are communicated to the operator, who provides feedback to the system to adjust its performance.

Thirdly, a specialized low-power hardware for SIFT [15] feature extraction enables decentralized image processing, *e.g.*, for an embedded system within a surveillance camera. A customized application-specific instruction-set processor (ASIP) with an instruction-set extension for accelerated computation of SIFT features is emulated on a Xilinx ML605 evaluation board. The ASIP is a Tensilica Xtensa LX4 processor. With this ASIP architecture, the system is still programmable and thus flexible for future algorithmic changes, and still provides the processing performance required. The fourth component addresses data security and privacy. All operators have encryption key pairs assigned by a Public Key Infrastructure (PKI). As all data usage and operator activities are logged and encrypted, only authorized auditors can access them, so that tampering is deterred.

The rest of the paper is organized as follows. Related works are illustrated in Sec. 2, while a system overview and a detailed description of its components are given in Sec. 3 and 4, respectively. Experimental results are provided in Sec. 5, while conclusions are given in Sec. 6.

2. Related Works

While for indoor airport surveillance many publications and systems exist [22, 19], outdoor airport monitoring has received much less attention. Most papers in the literature provide solutions to individual problems and only few describe systems tested at real airports. Individual solutions address specific monitoring tasks, *e.g.*, tail number recognition for aircraft identification [8], planes and aircraft detection [17], or vehicle tracking for activity monitoring [21].

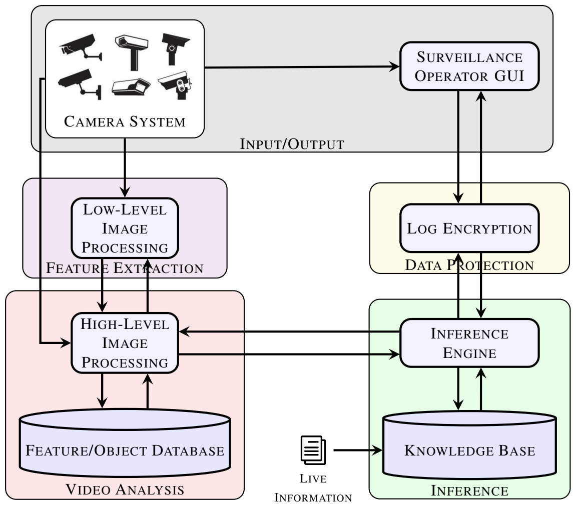


Figure 2: ASEV system overview.

Many systems integrate cameras and other sensors, like thermal sensors [10] or surface movement radars (SMR) [9]. The former employs a network of cameras and infrared sensors to detect and track objects on the apron, while the latter uses cameras as “gap-fillers” to provide motion information for radar-problematic areas. A pure camera-based commercial system is proposed by SAAB [3], where many Pan-Tilt-Zoom (PTZ) cameras in different small airports can be controlled from a centralized Remote Tower Center.

The works that are most similar to ours are related to the project AVITRACK and its follow-up Co-FRIEND [1, 2]. They have a similar operational framework addressing apron surveillance: motion detection, followed by object tracking and classification on the basis of predefined events. Even though both systems show some analogy in the overall design and in the individual components to the one proposed here, they both fail to address the four issues highlighted in Sec. 1. No communication from the operator to the system is envisaged; power consumption is not considered even if they are distributed systems; no measure for privacy and misuse control is taken. Our system not only provides a comparable scene analysis, but it also solves the aforementioned problems.

3. ASEV System Overview

Here, an overview of the system is given to illustrate its design and operation, while implementation details are postponed to the following section. The system is composed of four main operative blocks: feature extraction, video analysis, inference and data protection, as shown in Fig. 2. In addition, an input/output interface is available.

Input/Output - It comprises a camera system and a GUI for the operator. The camera system envisages overview and Pan-Tilt-Zoom (PTZ) cameras, and each overview camera

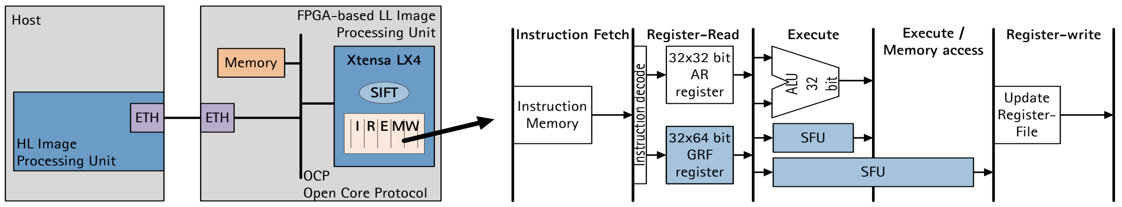


Figure 3: The LL unit consists of the Tensilica Xtensa LX4 ASIP with extended processor pipeline, DDR3 memory for local buffering of image processing results and communication interface. The HL unit sends images to the LL unit via Ethernet, and a feature list is returned. The processor pipeline is extended with a new register file and special functional units (SFU).

is assigned several PTZ cameras in a master-slaves fashion. The GUI allows the operator to switch between camera views, receive messages and send feedback to the system.

Feature Extraction - It contains the low-level (LL) image processing unit, which extracts SIFT features from camera images and transfers them to the HL unit. The former is parametrized by a set of process variables. This additional degree of freedom makes the system usable in many scenarios, as only a further tuning to each application is required.

Video Analysis - It comprises the high-level (HL) image processing unit and the object database.The HL unit analyses the scene by first performing motion detection and eventually, object tracking, recognition and classification on the basis of the features extracted by the LL unit. A database is used to store feature-based descriptions of known objects. At run-time, the operator can add descriptions of new objects to the database for future usage.

Inference - It envisages a real-time ontology-based inference engine and a knowledge base. The inference engine receives information from the HL unit, *e.g.*, object position or identity, and processes it according to a set of hierarchically structured rules contained in the knowledge base. The rules define allowed object behaviors on the basis of object attributes, *e.g.*, access permission, distance between vehicles, maximum speed, etc. Rules can be deleted, added and updated by the operator at run time. According to the resulting danger level, the inference engine sends specific messages to the operator screen.

Data Protection - It employs mechanisms for privacy protection and operator activity control. The need for this component is twofold. Not only must logs remain undisclosed, achieved using state-of-the-art encryption techniques, but operator activity must be traceable in case of tampering, achieved by introducing digital signatures.

In the following, the system operation is described. After the system initialization, where the operator identity is checked and system parameters are set, the HL unit starts performing motion detection. If a moving object is detected, a tracking thread is started on the most suitable PTZ camera associated to the corresponding overview camera. The HL unit computes the predicted object position in the chosen PTZ view, so that the camera focuses and zooms in. The PTZ camera pan, tilt, and zoom parameters are continuously adjusted during tracking. A recognition thread immediately follows the tracking thread. Feature-based recognition is performed by comparing the features extracted by the LL unit to those in the database. If recognition is not successful, a classification thread starts to evaluate the moving region with respect to three categories: aircraft, vehicle or person. The information yielded by the HL unit analysis is sent as streams to the inference module, that is based on a fast pattern matching algorithm. The algorithm takes the HL-streamed data as input, extracts objects from it and matches them to the knowledge base. Afterwards, it performs real-time rule-based reasoning, the results of which are displayed as on screen messages on the operator's monitor, who accepts or rejects them. The messages and the operator activity are encrypted to eventually trace them back.

The multithreading design provides the operator the maximum amount of information available. Tracking supplies data about object position and speed, on which basic inference rules apply. If classification or recognition are successful, the object category or identity is transmitted, refining rules accordingly. A rejection-based feedback mechanism automatically adjusts the inference rules. Moreover, privacy is guaranteed, as features are the only stored information, and system misuses are deterred by securely logging operator's activities.

4. System Components and Algorithms

In the following section, details about the technical implementation of the LL and HL units as well as of the inference engine and the log encryption module are given.

4.1. Low-Level Image Processing Unit

Given the necessity for accelerated SIFT processing and for platform flexibility as well as strict constraints on power consumption, an application-specific instruction-set processor (ASIP) offers the best trade-off. Thanks to the resulting acceleration, specialized feature-based tasks can achieve the required processing performance. The ASIP approach also allows for future algorithmic changes via software.

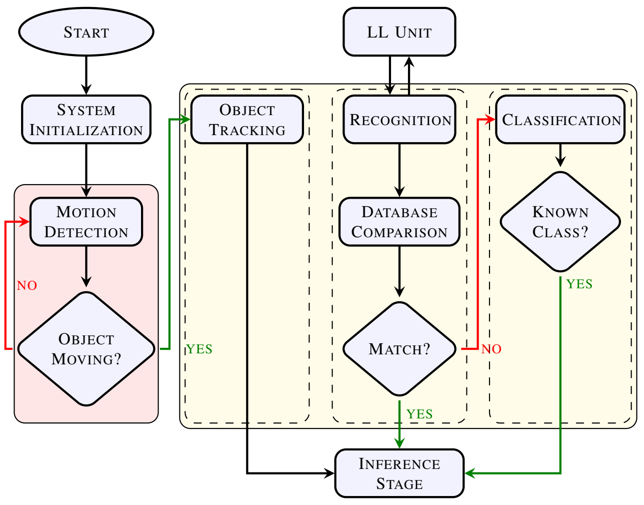


Figure 4: HL unit multithreading framework

Application-specific instruction-set processors, especially the Tensilica Xtensa processor, have shown their capability in many digital image processing applications before [7]. The low power consumption combined with the high flexibility of a programmable processor and the possibility to accelerate specific processing bottlenecks of algorithms provide a promising base for computer vision applications [18].

The LL unit consists of the commercially available Xtensa LX4 ASIP by Tensilica, 512 MB DDR3 memory and an Ethernet interface, as shown in Fig. 3. The base Xtensa LX4 core can be configured in multiple processor characteristics and is extendable by Tensilica Instruction Extensions (TIEs) to accelerate processing steps. In this case, the processor pipeline is extended with a new register file and special functional units. The emulation platform is a Xilinx ML605 evaluation board with a Virtex 6 FPGA [23].

To achieve the required processing performance, a new register file is defined for parallel image processing. Basic image processing and special SIFT processing steps are accelerated by customized special functional units. A Gaussian filtering, which is a symmetric and separable FIR filter, is accelerated, including a mode for choosing the kernel size. Furthermore, arithmetic functions with high computational costs, *e.g.*, arctangent, integer square root or sine/cosine, are approximated and accelerated with special functional units to meet the strict processing constraints.

4.2. High-Level Image Processing Unit

As shown in Fig. 4, the HL unit implements several tasks performing scene analysis in a multithreading fashion. Each block in Fig. 4 is described in the following.

System Initialization - It is composed of:

Camera calibration - The overview camera is fully calibrated with respect to a fixed world coordinate frame. The operator performs calibration using a simple GUI tool and airport maps with precise apron measurements. As the scene geometry can be approximated as two-dimensional, the world coordinate frame and the image frame are related by a homography $H_{o\to w}$. Then, each PTZ camera is automatically registered to its corresponding overview camera via a homography $H_{p\to o}$, so that all positions in all views can be expressed in one global coordinate frame.

Database Loading - Feature-based object descriptions are loaded for recognition purposes. At run time, the operator can add descriptions of new objects currently on the scene.

Motion Detection Thread - Motion detection is performed on the overview camera images, and is always in execution so that every event occurring on the scene can fire a tracking thread. Motion detection is performed through a Gaussian mixture-based segmentation algorithm [25]. The algorithm output is a binary image to which dilation and erosion operators are applied for noise reduction. The binary image is then analyzed by a blob detector that identifies the connected components whose size is larger than a fixed threshold. The threshold is inclusive in order to accommodate for small objects, like vehicles and persons. For each connected component, the overview camera assigns a tracking task to the most suitable PTZ camera on the basis of the calibration information.

Object Tracking Thread - The PTZ camera performs tracking on the region hypothesis provided by the overview camera. A KLT tracker is used to track the target object in consecutive video frames [20]. An estimation of the object direction based on the tracking information is used to continuously adjust the PTZ camera, which returns the updated pan p and tilt t parameters. Prediction is computed every two frames to keep camera motion to a minimum, as mechanical movements introduce delay. The size of the moving object is used to adjust the PTZ zoom parameter, changing the camera focal length to $f' = zf$, where z is the zoom factor and f the initial focal length. Accordingly, the homography to the overview camera changes to

$$H'_{p\to o} = H_{p\to o} K R_p R_t K_z^{-1}, \tag{1}$$

where K_z is the intrinsic camera matrix with focal length f', R_p and R_t are the rotation matrices defined by p and t. The position of the tracked object as well as its speed and direction are sent to the inference stage. For a more detailed scene analysis, a recognition thread is started.

Object Recognition Thread - Recognition is performed according to the feature-based paradigm. A set of SIFT features is extracted by the LL unit from the tracked region and compared to object descriptions in the database. The object with the highest number of geometrically consistent correspondences assigns its label to the moving region. To cope with pose changes, each object has several descriptions indexed by the view. If recognition is successful, the information about the object identity is sent to the inference stage, otherwise a classification thread is started.

Object Classification Thread - In case the object is not identified, the moving region is analyzed with a Deformable Part Model-based (DPM) classifier [11]. The classifier has been trained with three different categories: airplanes, vehicles and persons. In case of success, the object class is sent to the inference stage. It is important to remark that the object classifier provides only the label "Person", and not the precise person identity. Therefore, personal privacy is enforced without affecting surveillance performance.

4.3. Inference

The inference engine is based on a pattern matching algorithm that determines which system rules apply. RETE is a state-of-the-art system family for implementing inference engines [12]. A major challenge in this context is that the inference algorithm needs to process event-based data in real time. This can be obtained via *stream* or *incremental reasoning*. Thereby, we leverage a popular fast RETE-based implementation, BaseVISOr [4], and set it on top of a *semantic* knowledge base, empowered by an airport-domain ontology. The main difference between a static and a dynamic knowledge base for event streams is that in the streaming world, facts have an expiration date. As systems are constantly fed with new facts, old and no longer reality-representative facts need to be identified and removed. A trivial solution is to attribute each fact an expiration date, which gets updated if the same object is encountered again, otherwise the object expires and is automatically removed.

4.4. Log Encryption

To protect messages and trace operator activities, each operator is given a key pair by the PKI. The private key is linked to a personal ID and password, so that the operator can prove his/her identity at system start.

Logging envisages three operations: digital signature for operator tracing, encryption for message secrecy, integrity protection for manipulation deterrence. First, a signed log is obtained as the concatenation of the plain log and its SHA-256 digest, which is encrypted with the operator's private RSA key. Then, an encrypted signed log results from the concatenation of the AES-encrypted signed log and the AES key, which is itself encrypted with the public RSA key of the PKI. Finally, the encrypted signed log is concatenated to previous logs, and a further SHA-256 digest is computed.

5. Experimental Results

The experimental section presents both component and system evaluation. The LL unit and the inference engine are compared against standard implementations and benchmarks in order to highlight their individual contributions.

Function	Base processor	Our processor	# Invocations
Arctangent	500	29	1400
Integer square root	1534	22	1400
Sine	6532	5	2
Gaussian Filtering	161,492,072	1,562,757	
SIFT Feat. Extraction		1,023,910,174	

Table 1: The number of cycles for selected arithmetic functions with the corresponding average invocation for one detected feature during SIFT processing. The Gaussian filtering is simulated with a kernel size of 15×15 pixels. The image size is 640×480 pixels.

The ASEV system was evaluated at a real airport during a one-month test run. Extensive data has been collected and a thorough evaluation is presented in the following.

5.1. Low Level Image Processing Unit Evaluation

To evaluate the LL unit performance, the cycle count of the original SIFT C reference implementation [14] is compared to the cycle count of the accelerated processor. To be sure that the implemented approximations provide sufficient accuracy, the resulting descriptors are compared with the reference descriptors of the same images.

Cycle count reduction is the main goal to accelerate SIFT processing on the LL unit. In Table 1, cycle counts of the base processor and our extended processor are compared and the average number of function invocations for one detected feature is shown. Thanks to the processor extension, sine computation requires now 5 cycles instead of 6532, which means a speed-up factor of 1306. The arctangent and integer square root function are invoked 1400 times on average for one detected keypoint. With a reduction of the cycle count from 500 to 29 for one arctangent computation and from 1534 to 22 for one integer square root computation, the cycle count is reduced by 2.7 million cycles per feature just for those two computations. Compared to the SIFT C reference implementation on a basic Xtensa LX4 core, the number of total processing cycles, and thus the processing speed, is reduced by a factor of over 160 for the extended processor, still providing the original feature quality. As the LL unit is a programmable platform, full software flexibility is available, so that future SIFT software updates can be applied without changing the complete LL unit.

For a 45nm standard CMOS technology, the Tensilica tool estimated a power consumption of approximately 57.84 mW at 100 MHz and a silicon area of 1.3 mm^2. To the authors' best knowledge, this work is the first full ASIP implementation for SIFT feature extraction, without any other dedicated hardware accelerator.

5.2. Inference Engine Evaluation

To evaluate our inference engine in the streaming context, an evaluation framework is needed. We used the novel benchmark proposed in [16], that is a stream-based extension of LUBM [13]. Briefly, the benchmark provides a generated university knowledge base and dynamically changes a portion of the facts after every semester. The tested system needs to detect this inconsistency, remove the expired facts and infer new results. We consider it fair to evaluate our inference engine with this benchmark, as it assesses the intrinsic engine performance, which is uncorrelated to the application at hand. We compared our choice, BaseVISOr, with three state-of-the-art systems: Jess [5], Pellet [6] on top of Jena as well as OWLAPI. Loading time and query response time are used as main metrics.

First, we compared our choice with respect to the other RETE-based system, Jess. The experiment is carried out with the *Query 14* for SLUBM(1,0,5) (1 university over 5 semesters). Regarding loading time, BaseVISOr is faster by a factor of 37, 11.33 s *vs* 419.2 s, while for query response time BaseVISOr is faster by a factor of 3000, 0.14 s *vs.* 414.41 s. BaseVISOr is much faster because it exploits a triple-based data structure with binary predicates, thus simplifying the heavy work for pattern matching. To relate this experiment to our application, we have to think that inference time is approximately proportional to the amount of facts, which is in the order of 10^5 for this experiment and around 10^3 for our application, with approximately 10% of dynamic facts in both contexts. Figure 5 depicts the query response time for BaseVISOr, Pellet+OWLAPI and Pellet+Jena for SLUBM(10,0,5). BaseVISOr is slower at first, but becomes significantly faster with time, executing less computations and becoming more efficient, which is an essential characteristic of a stream-based reasoning system. Overall, we can conclude that our choice for BaseVISOr (RETE + optimization) guarantees a superior performance with respect to other state-of-the-art engines.

5.3. Overall System Evaluation

Here, the results of a one-month test run of the ASEV system at a regional airport are given. Results are presented for the motion detection, tracking and classification tasks.

To evaluate motion detection, Precision and Recall are considered. Precision, defined as $\frac{TP}{TP+FP}$, reaches a value of 60%, while Recall, defined as $\frac{TP}{TP+FN}$, reaches 95%. Given the system architecture and the multithreading design, False Positives are a minor issue, as they will be later discarded by the tracking, recognition or classification thread. On the contrary, Recall is an extremely important value for security, as False Negatives can be very dangerous. Just consider the potential danger of a full-loaded refueler moving undetected during passenger boarding. Our system proves to be almost free from False Negatives.

To assess tracking, three standard measures are considered: mostly tracked (MT) trajectories, mostly lost (ML) trajectories, and partially tracked (PT) trajectories [24]. MT

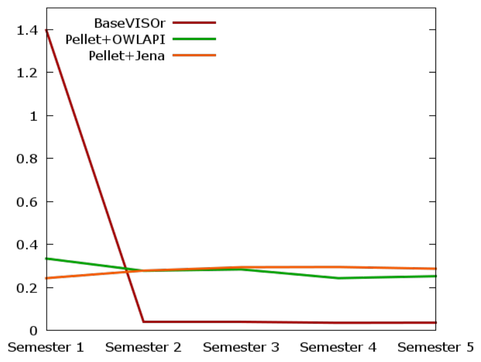


Figure 5: The performance of different systems for SLUBM(10,0,5)

		Classifier Output			
		Person	**Aircraft**	**Vehicle**	**No Object**
Actual Object	**Person**	0.725	0.034	0.0	0.241
	Aircraft	0.10	0.70	0.0	0.20
	Vehicle	0.25	0.0	0.50	0.25
	No Object	0.05	0.07	0.0	0.88

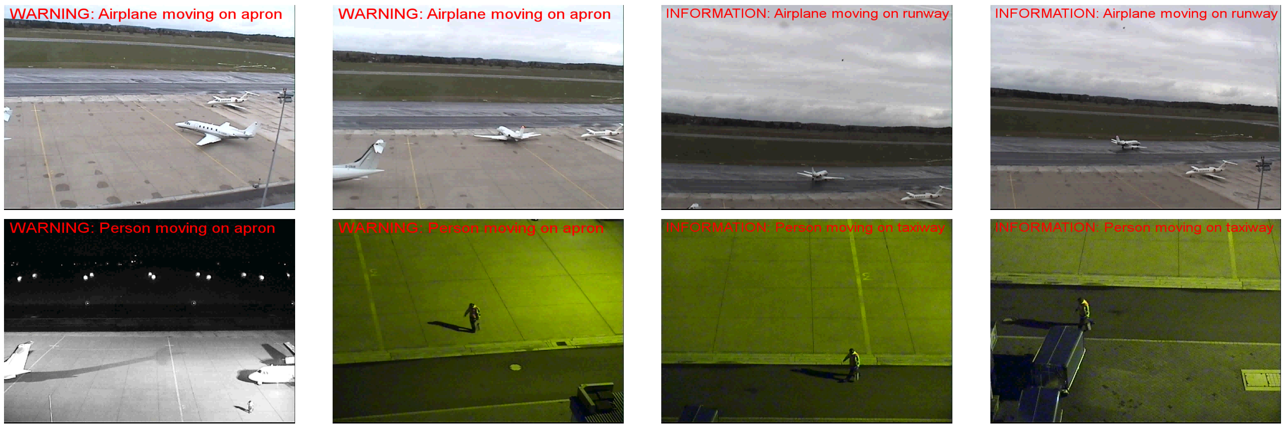
Table 2: Confusion table of the classifier

are defined as those trajectories that are successfully tracked for more than 80%, ML as those that are successfully tracked for less than 20%, PT as the remaining trajectories, *i.e*, $1 - MT - ML$. The system reaches a performance of 70% for MT, 25% for PT and 5% for ML, respectively. This indicates that the system succeeds in following most of the objects most of the time. In addition, most tracks are lost only before the object exits the scene, *e.g.*, when an aircraft is taking off or passengers approaches the entrance.

Regarding classification performance, the confusion matrix for the three categories is given in Table 2. A further explanation is needed for the "No Object" entry. This addresses the case in which the motion detector fires a tracking thread, and eventually a classification thread, but nothing is actually on the scene. The DPM classifier outputs also a score[1] that can be used to filter weak results. The threshold was experimentally set to -1. Regarding the three categories, the classifier performance is remarkably good for airplanes and persons, with a 70% and 72.5% of correct classification. The lower performance for vehicles is motivated by the great intra-class variability of the training set that includes very different vehicles, *e.g.*, refuelers, luggage carts, pushback tugs, etc. This can be coped with by exploiting the system capability of adding new object descriptions to the database at run time, so that newly stored objects will be likely found during the next recognition thread.

In Fig. 6, visual results of the system performance are provided. In the two sequences, a moving airplane is detected and tracked on the apron and on the runway, and airport employee is detected and tracked on the apron and

[1]Please refer to the DPM implementation for the score meaning


WARNING: Airplane moving on apron
WARNING: Airplane moving on apron
INFORMATION: Airplane moving on runway
INFORMATION: Airplane moving on runway
WARNING: Person moving on apron
WARNING: Person moving on apron
INFORMATION: Person moving on taxiway
INFORMATION: Person moving on taxiway
Figure 6: Airplane tracking on apron and runway (top), person tracking on apron and taxiway (bottom). Different messages describing each situation in terms of object, area and danger level are shown. Moreover, in the first sequence a still object is not detected, while the second sequence shows the system robustness to low illumination.

on the taxiway, respectively. The ASEV system correctly classifies the situation, in terms of object and location, and the inference engine provides messages with varying importance according to the knowledge base.

6. Conclusions

We proposed a system named ASEV for outdoor airport surveillance that combines four modules: an innovative low-level and a state-of-the-art high-level image processing units, a real-time inference engine and an encryption module. In addition to a full scene analysis, the proposed solution successfully addresses four often neglected aspects: two-way communication between system and operator, power consumption, privacy of the monitored people, and activity control of the operator. A one-month test at a real airport proved that the system assists the operator with sound and visual alerts, that result from an automatic assessment of a large set of different events on the apron.

Acknowledgement Research was conducted inside the BMBF-funded project (nr. 13N10718) ASEV.

References

[1] `http://ec.europa.eu/research/transport/projects/items/_avitrack____automated_surveillance_of_airport_aprons_en.htm`.

[2] `www.co-friend.net`.

[3] `http://www.saabgroup.com/Civil-security/Air-Transportation-and-Airport-Security/Air-Traffic-Management-Solutions/Remote-Tower/`.

[4] `http://www.vistology.com/basevisor`.

[5] `http://www.jessrules.com`.

[6] `http://clarkparsia.com/pellet`.

[7] C. Banz, C. Dolar, F. Cholewa, and H. Blume. Instruction Set Extension for high Throughput Disparity Estimation in Stereo Image Processing. In *ASAP*, 2011.

[8] J. A. Besada, J. M. Molina, J. García, A. Berlanga, and J. I. Portillo. Aircraft Identification Integrated into an Airport Surface Surveillance Video System. *MVA*, 2004.

[9] K. Dimitropoulos, N. Grammalidis, D. Simitopoulos, N. Pavlidou, and M. G. Strintzis. Aircraft Detection and Tracking Using Intelligent Cameras. In *ICIP*, 2005.

[10] G. Dumont, F. Berthiaume, L. St-Laurent, B. Debaque, and D. Prévost. AWARE: A Video Monitoring Library Applied to the Air Traffic Control Context. In *AVSS*, 2013.

[11] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object Detection with Discriminatively Trained Part-Based Models. *TPAMI*, 2010.

[12] C. Forgy. Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. *AI*, 1982.

[13] Y. Guo, Z. Pan, and J. Heflin. LUBM: A benchmark for OWL knowledge base systems. *JWS*, 2005.

[14] R. Hess. An open-source sift library. In *ICM*, 2010.

[15] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 2004.

[16] T. N. Nguyen and W. Siberski. SLUBM: An extended LUBM Benchmark for Stream Reasoning. In *ISWC*, 2013.

[17] F. Robert-Inacio, G. Bussone, S.and Chassaing, and Y. Maziere. Object Detection and Identification Applied to Planes and Aircraft for Airport Surveillance. *IVCNZ*, 2008.

[18] G. Schewior, H. Flatt, C. Dolar, C. Banz, and H. Blume. A Hardware Accelerated Configurable ASIP Architecture for Embedded Real-Time Video-based Driver Assistance Applications. In *ICSAMOS*, 2011.

[19] D. Schreiber, A. Kriechbaum, and M. Rauter. A multisensor surveillance system for automated border control (egate). In *AVSS*, 2013.

[20] J. Shi and C. Tomasi. Good Features to Track. In *CVPR*, 1994.

[21] D. Thirde, M. Borg, J. Aguilera, J. Ferryman, K. Baker, and M. Kampel. Evaluation of Object Tracking for Aircraft Activity Surveillance. *VS-PETS*, 2005.

[22] Z. Wu and R. Radke. Real-time airport security checkpoint surveillance using a camera network. In *CVPRW*, 2011.

[23] Xilinx. Virtex-6 FPGA ML605 Evaluation Kit. Technical report, Xilinx, Inc., 2012.

[24] B. Yang and R. Nevatia. An online learned crf model for multi-target tracking. *CVPR*, 2012.

[25] Z. Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *ICPR*, 2004.