Event Driven Software Architecture for Multi-camera and Distributed Surveillance Research Systems

Roberto Vezzani, Rita Cucchiara
University of Modena and Reggio Emilia - Italy
roberto.vezzani@unimore.it, rita.cucchiara@unimore.it

Abstract

Surveillance of wide areas with several connected cameras integrated in the same automatic system is no more a chimera, but modular, scalable and flexible architectures are mandatory to manage them. This paper points out the main issues on the development of distributed surveillance systems and proposes an integrated framework particularly suitable for research purposes. As first, exploiting a computer architecture analogy, a three layer tracking system is proposed, which copes with the integration of both overlapping and non overlapping cameras. Then, a static service oriented architecture is adopted to collect and manage the plethora of high level modules, such as face detection and recognition, posture and action classification, and so on. Finally, the overall architecture is controlled by an event driven communication infrastructure, which assures the scalability and the flexibility of the system.

1. Introduction

The explosion of requests of intelligent surveillance systems for different scenarios such as people monitoring in public areas, smart homes, urban traffic control, mobile application, and identity assessment for security and safety, leads research activity to explore many different dimensions in terms of both architectural issues and algorithms for scene recognition and interpretation. Thus, many synergic fields spanning from hardware embedded systems, sensor networks, computer architecture in one side and image processing, computer vision, pattern recognition in the other side are tightly integrated to cope with real-time surveillance applications.

A real world surveillance application, in fact, usually requires a good integration and replication of a large plethora of modules: integration in order to face compound problems and replication to manage more than one video source at the same time. Architectural and development issues must be accounted and are crucial for distributed camera systems.

Leading companies in surveillance have proposed their own integrated frameworks, specially focusing on reliability, extendibility, and scalability aspects. For example, IBM S3 [1] is an open and extensible framework for performing real-time event analysis. New detection capabilities can be included by adding plug-ins to the low level system (*SSE - Smart Surveillance Engine*), while all the fusion tasks are provided at the high level (*MILS - Middleware for Large Scale Surveillance*). Object detection, tracking, and classification are basic SSE technologies. Alert and abnormal event detection activities are instead performed at the MILS level. Similarly, ObjectVideo proposed the VEW system [2], developed starting from the prototype made under the VSAM (Video Surveillance And Monitoring) program [3]. Sarnoff Corporation proposed a fully integrated system called *Sentient Environment*, that can detect and track multiple humans over a wide area using a network of stereo cameras. Each stereo camera works independently proving detection and tracking capabilities, while data fusion is provided by means of a Multi-camera tracker (McTracker) which stores the extracted information on a databased for further query and analysis tasks.

Differently from these commercial frameworks, research laboratories need open and flexible platforms for analyzing, comparing and composing different algorithms and techniques. In addition more algorithms facing the same problem should be implemented for comparison and test. Reconfigurability and flexibility are more important than reliability and scalability aspects.

For this reason, research oriented frameworks often prefer to adopt and expand available software libraries such as OpenCV or Matlab toolboxes. These libs do not contain any facility for module composition and for multiple camera integration and no assumption refers to hardware or system architecture. Instead, we believe that a fruitful research needs an hardware and software infrastructure, flexible to test algorithms and approaches also well designed in their architectural and software engineering aspects to define and develop different platforms for different projects, allowing a reuse ans a fully exploitation of the knowledge.

Thus, in this paper we present an overview of the design aspects of system architecture and computer-based algorithms for multi-camera, distributed camera and heterogeneous sensor based surveillance systems, with specific regards to the problem of people surveillance.

2. Single, Multi and Distributed Camera Systems

Typical video surveillance systems consist at least of a single camera, connected with an embedded or general purpose computer with local or remote storage, display resources and related computer vision software. Independently from the application, some surveillance tasks are almost unavoidable, although their execution order and their mutual feedback can be designed in different ways, accordingly with the architecture of the system and the peculiarities of the application itself. For instance, some systems identify moving objects and then classify them to select only useful targets (e.g., vehicles or people); others, instead, provide target detection at the beginning and then they track selected objects only. A part from some initial image enhancement and image processing steps, in any surveillance system we can identify the following steps, even if their order can be different.

- **Detection**: *the task of exploiting space coherence to extract area of interest typically at image segmentation level.* In stationary camera it is always provided by background suppression, while in more general scenarios with moving cameras or without a reference image, segmentation according with motion fields, color, texture and so on have been adopted. Since the surveillance system detects and can be interested on vehicles, animals in addition to people, hereinafter we more generally refer to them with the term *moving object*.

- **Tracking**: *the task of exploiting time coherency to follow the same object along the time and the space.* The dichotomy of tracking-by-detection or detection-by-tracking is still open [4]. When detection is easily provided in stationary camera, appearance-based tracking models working at pixel level are preferable (e.g., [5, 6]). However, a complete survey on the topic was provided by Yilmaz *et el.* [7].

- **Recognition**: *the task of exploiting model coherency to provide object identification and classification*. This problem was initially underestimated in surveillance since the type of objects was often pre-defined given the application (*e.g.* vehicles in road or people in an office) but it is mandatory in real scenarios.

- **Understanding**: *the task of exploiting models to recognize actions, events and behaviors*. This is the final step, dependent on the surveillance application. New generations of smart surveillance systems must integrate a final step of high level reasoning to assess the situation, and infer possible dangerous or interesting behaviors, action or events.

- **[Data fusion]**: *the task of fusing information coming from different sensor sources to provide better detection and tracking, or to enlarge and enrich the recognition and understanding*. The natural evolution of standard surveillance systems goes in the direction of enlarging the data availability using more cameras in parallel and consequently more processing modules for providing the previously discussed steps. In all these situations, data fusion is mandatory.

2.1. A three layer data fusion

By means of the previously defined steps, each **single camera processing (SCP)** aims at generating and updating a set of *tracks*. We call single camera track (or just *track*) [8] the logical entity which represents a moving object inside the field of view of one camera. The track should be consistently maintained and updated during all the time is visible. The object's current appearance and position, as well as its history, are stored inside the track state.

When more than one single camera processing is active at the same time, more knowledge about the scene can be obtained from the combination of the single outputs. For example, instead of simply merging the outputs, obtaining a collection of tracks coming from different points of view, we can take care of identifying when multiple tracks refer to the same object. This task is called Consistent Labeling [9] and can be performed by means of geometrical and/or visual information [10]. The choice of the consistent labeling technique is stiffly related to the camera set up. In particular, if the fields of view of two or more cameras are overlapping, geometrical constraints can be used and a strong interaction among the involved single camera environments is required to fully exploit the time interval during which the same object goes through the overlapping zone and it is contemporaneously visible by all the cameras. We call **multi-camera processing (MCP)** the processing system devoted to control an overlapping set of cameras and which produces a set of multi-camera tracks (*mtracks*). *Mtracks* shall have a one-to-one correspondence with real observed objects and embed information coming from all their different views (cameras).

A weaker synchronization is required, instead, when the consistent labeling is performed among non overlapping cameras or when we are trying to re-identify the same object once it comes back to the scene after a while. In such a situation, this task is operated by a **distributed camera processing (DCP)** system.

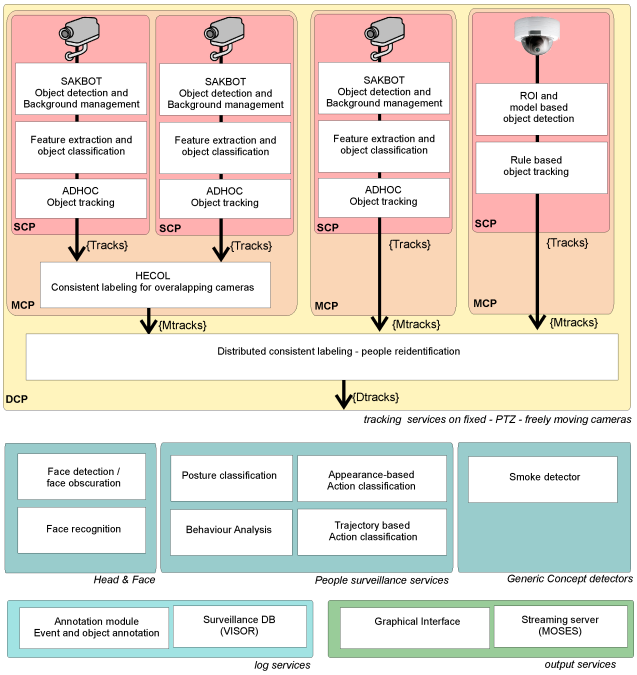


Figure 1. SOA-EDA research platform

Summarizing, the overall surveillance tracking system can be seen as a set of clusters of overlapping cameras. Each node consists of a single camera processing, embedding the traditional single view stack of tasks. Inside each cluster, a strong interaction among nodes guarantee the consistent labeling by means of geometrical and appearance constraints. Information coming from each cluster are then merged and managed by an higher level processing. The resulting framework is a *three-layer architecture* as depicted in the top-most part of Fig.1.

2.2. Multi-camera and distributed camera systems: a computer architecture analogy

Multi and distributed camera systems have a strong analogy with parallel computer architectures.

Multi-camera (or multi-view) systems are composed by many processes concurrently working on their own local memory data and must be synchronized to produce a shared visual data used by higher level processes, in a producer-consumer paradigm. This model has a direct correspondence with the shared-memory architecture model (multiprocessor or multicore), and inherits all the characteristics and requirements well known in tightly coupled multiprocessors. Low Level processes, viewed as threads working on the same processor or on different cores, execute initial steps of surveillance and reconstruct a shared scene and shared visual data where higher level processes can work. There are the same pros and cons of shared memory architecture: basically, by having all data in a shared space, higher level threads can exploit all information at pixel level or in an intermediate level (e.g. the moving blobs). On the other hand, the architecture is less scalable and the capability is bounded by the capacity of the system to work in a multi-thread fashion with many parallel streams.

Distributed Surveillance Systems are based on multiple camera and processing modules, without the need of overlapping field of views to fuse surveillance data and information in an enlarged surveillance space. They are constituted of fixed, moving, PTZ cameras connected in a loosely-coupled architecture. This architecture is potentially more flexible since the nodes can have a more or less powerful processing capability. A node can be either a smart camera or a classical single stationary surveillance system, either a multi-camera system with a possibly infinite scalability. Moreover, heterogeneous sensors can be also easily integrated. Here the challenges are similar to the ones of parallel distributed architectures, and the most profitable trade-off between local and remote computation should be defined. In addition to the computational power, the network bandwidth may be the obvious bottlenecks of the system, so that normally the exchanged messages regard high level knowledge only (e.g., a people ID or textual data) and, only if necessary, the complete pixel-level data.

Algorithms for distributed environments are different to multi-camera ones: for instance the problem of maintaining the same identity of the individual along the time, previously called consistent labeling, here is normally named re-identification and is typically addressed with similarity search as in the multimedia retrieval.

3. Design issues for surveillance systems

In addition to the tracking, a complete surveillance system should include several high-level modules coping the recognition and understanding points. Face detection, face recognition, action analysis, event detection, and so on are some examples of functionalities of automatic people surveillance systems. Other modules can be added depending on the application and/or the research environment. Some of these tasks should or can be performed on a subset of the installed cameras only, or with a particular time schedule. Moreover, the same task (e.g., face detection) can be carried on using different techniques. A wide surveillance system should be able to take into account all these requirements. To this aim, we propose a Service-Oriented Architecture to model each system functionality, with an Event Driven communication schema supporting this plethora of services.

3.1. Static Service Oriented Architecture

As proposed by Detmold *et al*[11], a Service Oriented Architecture (SOA) approach [12] can be efficiently used

for distributed video surveillance systems. Similarly to [11], we have implemented each high-level functionality as an individual and isolated service, even if no standard description languages have been adopted to define the services and the orchestration, using a static SOA approach [13, 14]. Static SOA environments are similar to traditional systems, in which "services" acts similarly to "components". In addition, dynamic composition is possible (at run-time it is possible to configure the set of working services), but dynamic discovery is limited to a predefined service pool. Surveillance services comprise unassociated, loosely coupled units of functionality that have no calls to each other embedded in them. The main goal is the reconfigurability of the system, leading to low marginal costs of creating the n-th application.

3.2. Event Driven communication Architecture

The modular architecture described in section 3.1 has been developed to keep each module isolated from the others. Anyway, a communication architecture is required to manage the overall system. To this aim we propose an event driven communication architecture (EDA) [15]. EDA is a well known software pattern; events and their production, detection, consumption of, and reaction are the core of such applications. It is usually applied as a synchronization method among loosely coupled software components and services. It could be used also in tightly coupled architecture in a distributed-shared memory paradigm. An event-driven system typically consists of agents (event generators) and sinks (event consumers). Sinks have the responsibility of applying a reaction as soon as an event is raised by an agent. The event based communication paradigm is used in a wide range of applications and it regulates the behavior and the interactions among items in nature as well.

Chandy *et. al.* in [16] report definitions and some issues for the development of event-driven systems. Using the proposed terminology, we describe the developed surveillance framework.

Events. Our architecture follows the "sense-and-respond" schema. The set of input cameras, together with low level processing units, are the sensing parts inside our "universe". Then, the other processing modules usually operate as *processing agents* and *responders* at the same time, depending on their function and their role. Each time a processing module detects a *significant state change* in the analyzed data it will generate an *event*. In particular, the tracking modules will generate events such as *NewTrack*, *TrackExits*, *TrackUpdate*, and so on. Higher level modules can operate as a responders since they will capture events from lower level modules, and can perform processing tasks and to generate their own events. For example, people detectors will be triggered by *NewTrack*, and *TrackUpdate* events, generating *PersonDetected* events whenever a person is found. This event is then an input for face detection modules: each time they receive such an event they perform face detection on the person appearance, and generates events such as *FaceDetected*. Some modules, such as the annotation modules, operates as responders only. Many events are obtained by combining other base events or my means of temporal processing. Thus we can consider them as Complex Events and our architecture as a Complex Event Processing (CEP) [15].

Information flow. The connections between modules are fully configurable, allowing each module to subscribe to events of any other module. In particular, two different connection types are created. Directly from the work flow of traditional video surveillance applications a tree structure can be created. Starting from the video sources as leaves up to the global application as root, each module has a specific parent, which is automatically subscribed to its children's events. This structure follows the "has-a" component relations. In addition to the default parent, other modules can subscribe to a particular agent's events. For example, the annotation module collects events coming from different sources, also from components belonging to other branches of the tree. In Fig. 2 the network of event subscriptions is depicted. Continuous arrows represent the parental tree, while dashed ones indicate additional subscriptions.

Modes of Interaction between Components. Three Modes of Interaction between Components have been defined: *Schedule*, *Pull*, *Push* [16]. Our framework exploits all of them:

- *Schedule*: the normal activity of the framework is regulated by a *heart-beat* clock which trigger the normal processing activity of each module. For example, each SCP modules performs the object detection and tracking steps, other modules may update internal counters only, and so on; during these "normal" activities each module can detect state changes and accordingly generates events.

- *Push*: the flow of events, instead, follows the Push interaction, since they arise when the source module detects a state change and want to propagate it to responsible listeners.

- *Pull*: specially for optimization reasons, not all the state changes are propagated by means of events; thus, sometimes some modules need to sample the state of other modules, applying a push interaction.

4. A SOA-EDA research platform

The proposed paradigms and techniques have been integrated in a C++ video surveillance Library. In addition to base classes for image processing, motion detection, object tracking, as well as video source management, video output, and graphical interface, the library supports the definition of isolated services as well as an event driven orchestration among services. In Fig. 1 a schema of the overall frame-

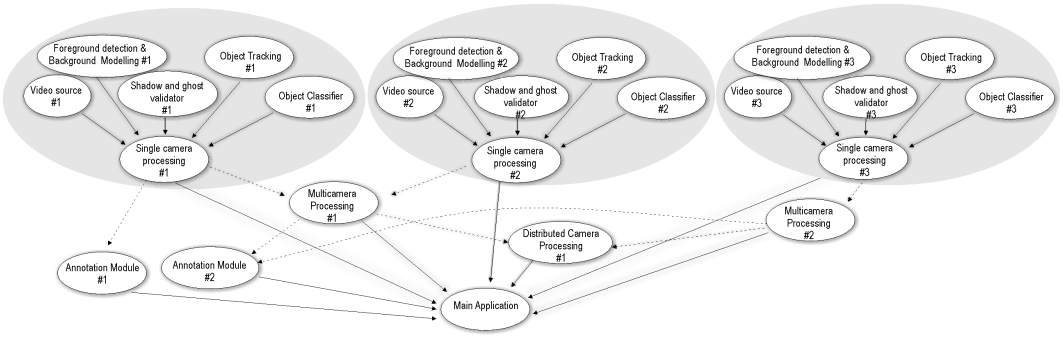


Figure 2. Event Flow for a three-camera system, with two of them overlapping: continuous arrows represent the parental tree, while dashed ones indicate additional subscriptions.

work is reported. In this section a brief description of each implemented service is given, with particular emphasis on the three-layer tracking system, which is the core service of each surveillance application.

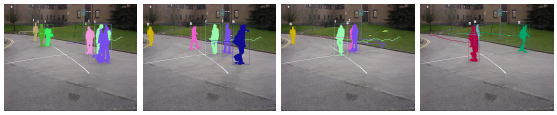
4.1. The three tracking layers

In this section a summary description of the implemented three tracking layers (namely, the SCP, the MCP, and the DCP layer) is given.

SCP - Single Camera Tracking. In this layer, we assume that the models of the target objects and their motion are unknown, so as to achieve maximum application independence. In the absence of any *a priori* knowledge about target and environment, the most widely adopted approach for moving object detection with **fixed camera** is based on *background subtraction* [17, 18]. It is well known that background subtraction carries two problems: the first is that the model should reflect the real background as accurately as possible, to allow the system accurate shape detection of moving objects. The second problem is that the background model should immediately reflect sudden scene changes such as the start or stop of objects, so as to allow detection of only the actual moving objects with high reactivity (the "transient background" case). If the background model is neither accurate nor reactive, background subtraction causes the detection of false objects, often referred to as "ghosts" [17]. In addition, moving object segmentation with background suppression is affected by the problem of *shadows*. Indeed, we would like the moving object detection to not classify shadows as belonging to foreground objects, since the appearance and geometrical properties of the object can be distorted and delocalized, which in turn affects many subsequent tasks. Moreover, the probability of object under-segmentation (where more than one object is detected as a single object) increases due to connectivity via shadows between different objects. We implemented in our framework the Statistical And Knowledge-Based ObjecT detection algorithm, called **Sakbot**, proposed by *Prati et al.* in [19]. Sakbot exploits statistics and knowledge of the segmented objects to improve both background modeling and moving object detection and integrates an effective shadow removal algorithm. A background model based on Mixture of Gaussians [18] has been implemented and can be selected in the place of Sakbot. Inside each SCP, we implemented and integrated some different tracking algorithms. Among them, we included the appearance-based tracking algorithm proposed in [6], which is characterized by some peculiarities that makes it particularly suitable for video surveillance applications. In Fig. 3 we have collected some snapshots of the system output, proving that the SCP tracking allows to correctly manage mutual occlusions and foremost object recognition. Finally, since the low level processing is model free, it can be used to track vehicles or other objects too, such as the van in the distance in Fig. 3 (c).

MCP - Multi-camera tracking. As previously defined in section 2.1, the multi-camera engine is devoted to control a cluster of overlapping cameras, performing a consistent labeling task over the sets of single camera tracks. To this aim, we implemented the approach proposed in [10] which belongs to the class of geometry-based techniques. Each time a new track is detected in one camera C^i in the overlapping area, its support point is projected in the other cameras C^j by means of homography transformations look-

(a) V5-fr.703 (b) V6-fr.278 (c) V7-fr.200 (d) V7-fr.279



(e) V8-fr.32 (f) V8-fr.154 (g) V8-fr.164 (h) V8-fr.257

Figure 3. Qualitative results of the implemented SCP tracking on PETS2009 dataset.

ing for corresponding tracks. The match is then reinforced by means of epipolar constraints in a Bayesian framework.

DSP - Distributed camera tracking. Let us suppose to have a wide space monitored by a set of non overlapping fixed cameras (or non overlapping multiple camera systems). If the fields of view of the cameras are close enough, it is plausible that a person exiting a camera field of view will soon appear on the field of view of a neighbor camera. Moreover, if the illumination conditions and the camera type are not so different, a person will generate similar color histograms on different views. To obtain a distributed consistent labeling, we are interested on comparing two different MTracks in order to detect if they belong to the same moving object. Let us introduce the global *Compatibility Score* between two tracks, i.e., a value which indicates if the two tracks can be effectively generated by the same real object. Compatibility Score is defined in a negative way: a good Compatibility Score does not mean that the two objects are the same, but that nothing has been detected to say they are not. Temporal, geometrical, and appearance constraints are both considered. For example, we can state that two tracks are not related to the same person if they appear at the same time on different non overlapping cameras, or if they appear in two different cameras but the time delay is not enough to cover the distance; similarly, differences detected on clothes colors, or the real height, decreases the Compatibility Score.

We modeled the Compatibility Score by means of two terms. One is based on color histogram and the other one on the mutual camera positioning. To this aim we exploit computer graphics techniques as proposed in [20]: interactions between different scenes and the movements of avatars are usually managed with graph models by the computer graphic and virtual reality communities. All the possible avatar positions (or rooms) are represented by nodes and the connecting arcs refers to allowed paths. The sequence of visited nodes is called *pathnodes*. A weight can be associated to each arc in order to give some measures on it, such as the duration, the likelihood to be chosen with respect to other paths, and so on. We empirically set the arc weights, but a learning phase can be established to automatically compute them.

According with the allowed paths, the overall area can be divided into three region types: A. Visible regions, corresponding to camera's fields of view; B. Non visible regions, i.e., transition areas between two or more visible regions; C. Forbidden or exit regions. In particular we are interested to identify the boundaries of these regions. People inside visible regions can be detected and tracked by the corresponding single camera system; once they exit that region to enter in a non visible region, they will likely appear again in another visible region. The weight associated to each arc is considered for the Compatibility Score computation. People exiting the overall scene are no more considered by the system, and their Compatibility Score with subsequent tracks are null.

4.2. High level surveillance services

The tracking module is only part of a complete surveillance system. Other items should be added in order to automatically extract useful information and events. For example, we have focused our activity to people surveillance, which calls for tasks such as face detection and recognition, posture and behavior classification, event detection, and so on. These high level modules are not directly included in the different layers of the tracking system, but are kept independent. In Fig. 1 a schema of the framework is reported and in Table 1 the main services implemented in the prototype are described. In the table, some of the input events (which trigger the module execution on the specified data) and the output events (which indicate the service outputs) are indicated. In Fig. 4 some of the main people surveillance services are illustrated together with the most important events which trigger the service activities. The framework also integrates the OpenCV library, allowing the development of services based on already implemented functionalities such as face detection. Finally, the proposed framework is also enriched with a unpretentious graphical interface, which allows to monitor the output of each service as well as intermediate results exploiting the same event driven communication architecture. The application is realized as an MDI framework and the services are dynamically added at runtime. A comprehensive preference windows lets to control all the system thresholds and parameters. Fig. 5 shows a snapshot of the working framework; on the left, inspection windows for the intermediate and final output of the services; on the right, the thresholds and parameters dialog window.

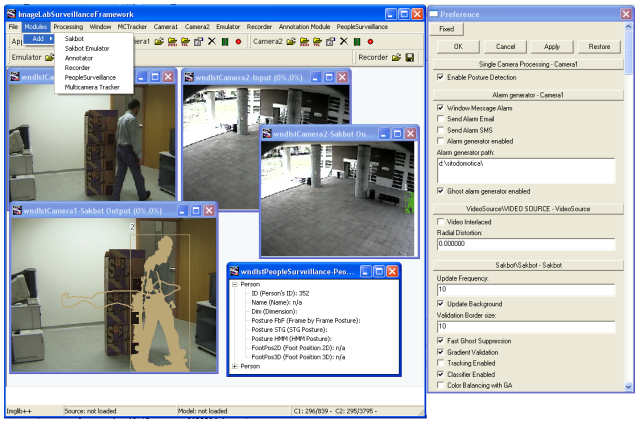


Figure 5. A snapshot of the graphical user interface

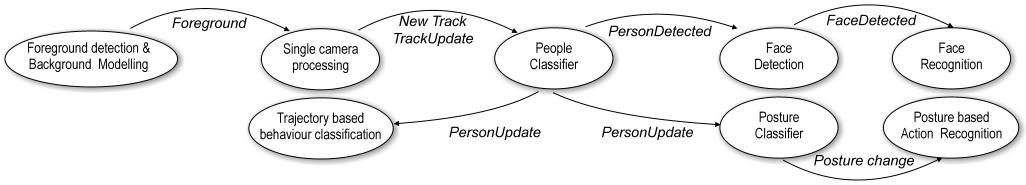


Figure 4. Main services and synchronization events of a typical people surveillance applications

Service	Description	Input Events	Output Events
People Classifier	The HoG based people classifier [21] was implemented as a service to detect people among the set of tracks, whenever they appear in the scene	*NewTrack*, *TrackUpdate*	*PersonDetected*, *PersonUpdate*
Face detector	We integrated in the framework two different face detectors: the well known Viola&Jones of the OpenCV library and the face detection library by Kienzle *et al.* [22]	*NewTrack*, *PersonDetected*	*FaceDetected*
Posture Classifier	The frame-by-frame posture of each person can be classified by means of the visual appearance. The implemented posture classification is based on projection histograms and select the most likely posture among Standing, Sitting, Crouching and Laying Down [23]	*PersonDetected*, *PersonUpdate*	*PersonePosture*, *PostureChange*
Appearance based action recognition	The action in progress is detected using features extracted from the appearance data. Walking, waving, pointing are some examples of the considered actions. Two different approaches have been selected and implemented: the first is based on Hidden Markov Models [24] and the second on action signature [25]	*PersonDetected*, *PersonUpdate*, *PostureChange*,	*PeopleAction*
Trajectory-based action recognition	People trajectories (i.e., frame by frame positions of the monitored tracks) embed information about the people behavior; in particular they can be used to detect abnormal paths which can be related to suspicious events. A trajectory classifier has been implemented in our system following the algorithm described in [26]	*PersonUpdate*, *TrackUpdate*	*PeopleAction*
Smoke detector	The smoke detection algorithm proposed in [27] has been selected. The color properties of the object are analyzed accordingly to a smoke reference color model to detect if color changes in the scene are due to a natural variation or not. The input image is then divided in blocks of fixed sized and each block is evaluated separately. Finally a Bayesian approach detect whether a foreground object is smoke	*NewTrack*	*SmokeDetected*
Video Streaming	We integrated the MOSES (MObile Streaming for vidEo Surveillance) streaming system proposed by Gualdi *et al.* [28], that supports video streaming in different conditions, aiming at low-latency transmission over limited-bandwidth networks. Additionally, the video stream is provided with a sufficient quality to be correctly analyzed by both human-based or computer-based video surveillance layers.	*NewFrame*	-

Table 1. List of the main surveillance services implemented in the framework; for each service, the main input and output events are listed

5. Conclusions and future work

In the paper, a complete surveillance framework for research purposes is described. The main novelties are a three layer tracking architecture which allows the implementation of single, multi and/or distributed camera systems. The plethora of high level surveillance modules have been structured as a set of services, integrated in the same system by means of an event driven communication model. The surveillance services, the visual interface and the event infrastructure have been implemented in a C++ library, which supports a lot of different applications. A graphical interface for the configuration and the generation of specific applications at run-time is under construction.

The framework is currently under development and improvement within the project *THIS*, with the support of the Prevention, Preparedness and Consequence Management of Terrorism and other Security-related Risks Programme European Commission - Directorate-General Justice, Freedom and Security.

References

[1] Y.L. Tian, L.M. Brown, A. Hampapur, M. Lu, A. Senior, and C.F. Shu, "Ibm smart surveillance system (s3): event based video surveillance system with an open and extensible framework," *Machine Vision and Applications*, vol. 19, no. 5-6, pp. xx–yy, October 2008. 1

[2] N. Haering, P. L. Venetianer, and A. Lipton, "The evolution of video surveillance: an overview," *Mach. Vision Appl.*, vol. 19, no. 5-6, pp. 279–290, 2008. 1

[3] R. Collins, A. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, and O. Hasegawa, "A system for video surveillance and monitoring," Tech. Rep. CMU-RI-TR-00-12, Robotics Institute, Pittsburgh, PA, May 2000. 1

[4] M. Andriluka, S. Roth, and B. Schiele, "People-tracking-by-detection and people-detection-by-tracking," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2008*, 2008, pp. 1–8. 2

[5] T. Zhao and R. Nevatia, "Tracking multiple humans in complex situations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1208–1221, 2004. 2

[6] R. Vezzani and R. Cucchiara, "Ad-hoc: Appearance driven human tracking with occlusion handling," in *1st Int'l Workshop on Tracking Humans for the Evaluation of their Motion in Image Sequences*, Leeds, UK, Sept. 2008. 2, 5

[7] Alper Yilmaz, Omar Javed, and Mubarak Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, no. 4, pp. 13, 2006. 2

[8] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting moving objects, ghosts and shadows in video streams," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1337–1342, Oct. 2003. 2

[9] S. Khan and M. Shah, "Consistent labeling of tracked objects in multiple cameras with overlapping fields of view," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1355–1360, Oct. 2003. 2

[10] S. Calderara, R. Cucchiara, and A. Prati, "Bayesian-competitive consistent labeling for people surveillance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 354–360, Feb. 2008. 2, 5

[11] Henry Detmold, Anton van den Hengel, Anthony Dick, Katrina Falkner, David S. Munro, and Ron Morrison, "Middleware for distributed video surveillance," *IEEE Distributed Systems Online*, vol. 9, 2008. 3, 4

[12] Thomas Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005. 3

[13] W.-T. Tsai, Xiao Wei, Zhibin Cao, Raymond Paul, Yinong Chen, and Jingjing Xu, "Process specification and modeling language for service-oriented software development," in *Proc. of 11th IEEE International Workshop on Future Trends of Distributed Computing Systems. FTDCS '07*, march 2007, pp. 181 –188. 4

[14] M.B. Blake, "Decomposing composition: Service-oriented software engineers," *Software, IEEE*, vol. 24, no. 6, pp. 68 –77, nov.-dec. 2007. 4

[15] D. C. Luckham, *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001. 4

[16] K. M. Chandy, M. Charpentier, and A. Capponi, "Towards a theory of events," in *DEBS '07: Proc. of the 2007 inaugural int'l conference on Distributed event-based systems*, New York, NY, USA, 2007, pp. 180–187, ACM. 4

[17] I. Haritaoglu, D. Harwood, and L.S. Davis, "W4: real-time surveillance of people and their activities," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 809–830, Aug. 2000. 5

[18] C. Stauffer and W.E.L. Grimson, "Adaptive background mixture models for real-time tracking," in *Proceedings of IEEE Int'l Conference on Computer Vision and Pattern Recognition*, 1999, pp. 246–252. 5

[19] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting moving objects, ghosts and shadows in video streams," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1337–1342, Oct. 2003. 5

[20] R. Vezzani, D. Baltieri, and R. Cucchiara, "Pathnodes integration of standalone particle filters for people tracking on distributed surveillance systems," in *Proceedings of 25th International Conference on Image Analysis and Processing (ICIAP2009)*, Vietri sul Mare, Salerno, Italy, Sept. 2009. 6

[21] Navneet Dalal and Bill Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR05)*, Washington, DC, USA, 2005, pp. 886–893, IEEE Computer Society. 7

[22] W. Kienzle, G. Bakir, M. Franz, and B. Scholkopf, "Face detection - efficient and rank deficient," *Advances in Neural Information Processing Systems*, vol. 17, pp. 673–680, 2005. 7

[23] R. Cucchiara, C. Grana, A. Prati, and R. Vezzani, "Probabilistic posture classification for human behaviour analysis," *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, vol. 35, no. 1, pp. 42–54, Jan. 2005. 7

[24] R. Vezzani, M. Piccardi, and R. Cucchiara, "An efficient bayesian framework for on-line action recognition," in *Proceedings of the IEEE International Conference on Image Processing*, Cairo, Egypt, Nov. 2009. 7

[25] S. Calderara, R. Cucchiara, and A. Prati, "Action signature: a novel holistic representation for action recognition," in *5th IEEE International Conference On Advanced Video and Signal Based Surveillance (AVSS2008)*, Santa Fe, New Mexico, Sept. 2008. 7

[26] S. Calderara, A. Prati, and R. Cucchiara, "Learning people trajectories using semi-directional statistics," in *Proceedings of IEEE International Conference on Advanced Video and Signal Based Surveillance (IEEE AVSS 2009)*, Genova, Italy, Sept. 2009. 7

[27] P. Piccinini, S. Calderara, and R. Cucchiara, "Reliable smoke detection system in the domains of image energy and color," in *6th International Conference on Computer Vision Systems, Vision for Cognitive Systems*, 2008. 7

[28] G. Gualdi, A. Prati, and R. Cucchiara, "Video streaming for mobile video surveillance," *IEEE Transactions on Multimedia*, vol. 10, no. 6, pp. 1142–1154, Oct. 2008. 7