Adaptive Metadata Management System for Distributed Video Content Analysis

C. Carincotte[1], X. Desurmont[1], and A. Bastide[2]

[1] Multitel asbl, Rue Pierre et Marie Curie 2, 7000 Mons, Belgium
{carincotte,desurmont}@multitel.be
http://www.multitel.be
[2] ACIC, Incubateur technologique, Boulevard Initialis 7 B2, 7000 Mons, Belgium
bastide@acic-tech.be
http://www.acic.eu

Abstract. Scientific advances in the development of video processing algorithms now allow various distributed and collaborative vision-based applications. However, the lack of recognised standard in this area drives system developers to build specific systems, preventing from e.g. content analysis components upgrade or system reuse in different environments. As a result, the need for a generic, context-independent and adaptive system for storing and managing video analysis results comes out as conspicuous. In order to address this issue, we propose a data schema-independent data warehouse backed by a multiagent system. This system relies on the semantic web knowledge representation format, namely the RDF, to guarantee maximum adaptability and flexibility regarding schema transformation and knowledge retrieval. The storage system itself, namely data warehouse, comes from the state-of-the-art technologies of knowledge management, providing efficient analysis and reporting capabilities within the monitoring system.

1 Introduction

Automatic processing of data coming from video cameras is currently a field of activity stirring up the utmost attention in the pattern recognition community. State-of-the-art advances in this area enable the reliable extraction of huge amounts of low-level features (e.g. colour, texture and shape of mobile objects, object tracks...). The analysis and interpretation of this metadata then allow complex event processing such as event detection and scenario recognition (e.g. face/object recognition, abandoned luggage detection...). Nevertheless, the way the produced knowledge is used and shared in closed circuit television (CCTV) systems calls now for more considerations so as to bridge the gap between specific analysis algorithms and end-users expectations. For instance, a security operator may not only want to be informed in case of important event detection, but also to rapidly and dynamically analyse the produced metadata to understand how/why an unexpected event has been identified. He may also want to analyse the generated metadata over a long period, so as to discover general pattern of events/activities within the monitored architecture. Ultimately, these operations should be achieved without interrupting the ongoing real-time analysis.

J. Blanc-Talon et al. (Eds.): ACIVS 2008, LNCS 5259, pp. 334–345, 2008.

In this context, the need to centralize and benefit from the extracted metadata in an efficient way becomes more and more present. Recent works have explored the use of dedicated data management systems for video monitoring purposes [12,20]. In VSAM project [4], results from each video processing units are archived in an object/event database of known objects, allowing object- and activity-oriented retrieval. In the IBM Smart Surveillance System [17], the metadata produced by the surveillance engines is indexed into predefined tables, providing event-based retrieval. In the SFinX system [16], the trajectory of each object is extracted using *sequence data representation*, and stored in the events database, enabling video retrieval via a variety of attributes. In RETRIEVE project [15], MPEG-7 metadata descriptors are used to describe video content; operators can then search the metadata by any available descriptor. Due to the lack of standards regarding analysis output format and storage, such systems are usually dedicated to specific contexts and data standards, and rely on relational databases structured according to these predefined schemas. As a result, the specification of transmitted data must be *a priori* known, ruling out any *a posteriori* modification of the used schemas and content extraction modules.

In this paper, we present a management system for distributed video processing offering both data schema[1] independence and highly reactive behaviours. The proposed system allows to handle metadata without being aware of its structure before initialization, thus providing an open, generic and flexible monitoring architecture. More precisely, our contribution in this context is twofold:

- With the intention of building a real-time distributed system, we first propose to resort to Agent-Oriented Programing (AOP) to handle metadata transfer, insertion, and retrieval. Indeed, AOP offers a high degree of flexibility, autonomy and proactivity that has already demonstrated its relevance and suitability for sharing knowledge in distributed environments [3]. Moreover, we suggest to exploit the possibilities offered by AOP through a special kind of data-oriented agent providing an efficient handling of metadata within the system.
- Our second contribution is to use data warehouses' concept for the storage task. Data Warehouses (DW) are specific databases which were developed during the last decades to meet a growing demand for information management and analysis that could not be provided by classical databases [11]. Nevertheless, exploiting DW capabilities in a video analysis environment appears to be very interesting [13], especially due to its ability to support complex queries and analysis on the stored data without slowing down the on-going analysis task. In this context, our contribution is to perform the distributed mass storage of metadata using an RDF-based DW, associated to the agent-oriented environment.

The paper is organised as follow; the application context is firstly presented in Sec. 2. Sec. 3 then focuses on the storage system architecture, from the knowledge flow management in the system to the DW specifications. Sec. 4 finally introduces evaluation results, whereas conclusions and perspectives are drawn in Sec. 5.

[1] A data schema is a description format, typically expressed in terms of constraints on the structure, syntax and content of the description document.

2 Monitoring System Overview

So as to detail the developed system, it is necessary to precise its operational context as well as the functionalities it intends to take on. The considered monitoring environment aims at assessing multimedia knowledge-based content extraction and analysis components developed within the IST CARETAKER project. More precisely, the project focuses on both:

- the reliable extraction of structured knowledge from data acquired over a network of camera/microphones (surveillance network of Roma/Torino metros). This is currently achieved thanks to the investigation of distributed techniques for real-time extraction of semantic metadata from audio/video data streams;
- the relevant exploitation of the extracted information to ease end-user missions (metro monitoring by safety/security operators), which is addressed through event-based retrieval functionalities, and off-line processes of metadata for extraction of longer term patterns of activity.

Fig. 1 present the architecture of the CARETAKER monitoring environment in which both on-line and off-line parts are clearly identifiable.

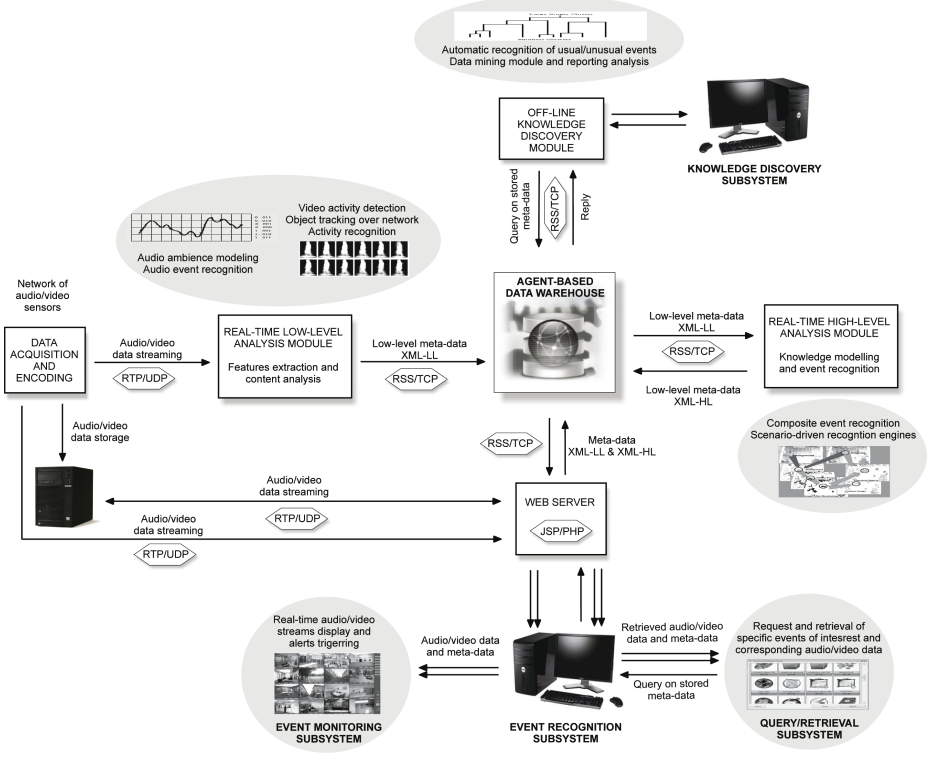


Fig. 1. Design configuration of the audio/video monitoring architecture

Real-time processing. As depicted in Fig. 1, the acquired audio-visual streams are analysed by two different real-time modules. In a first time, raw data coming from the sensors are automatically analysed by a first processing unit responsible for the low-level features extraction (REAL-TIME LOW-LEVEL ANALYSIS MODULE). This layer allows to extract some primitive characteristics from the audio/video raw data such as ambient sounds, mobile objects, crowd density, object trajectories... The low-level semantic descriptors (metadata) resulting from this analysis are then incorporated into the knowledge management system (AGENT-BASED DATA WAREHOUSE). A second layer of higher-level analysis (REAL-TIME HIGH-LEVEL ANALYSIS MODULE) then processes the previously computed metadata, so as to identify events of interest, such as turnstile jumping, left-luggage detection... Resulting high-level metadata is also incorporated in the DW.

Off-line analysis. Regarding the offline part of the architecture, a *knowledge discovery* module (OFF-LINE KNOWLEDGE DISCOVERY MODULE) allows to analyse the stored metadata using clustering/data-mining techniques. This module aims at identifying general trends in the stored metadata, computing statistics (flow of people...) and exploring the relationship between different types of events.

Graphical user interfaces. With respect to both on-line and off-line processing, the extracted information is exploited through two dedicated subsystems. A first EVENT RECOGNITION SUBSYSTEM offers the standard monitoring interface through a web server, triggering alarms corresponding to events detected in real-time. It also allows event driven queries on the DW, and retrieval of related audio/video data. A second KNOWLEDGE DISCOVERY SUBSYSTEM allows users to query combinations of higher-level semantic event and to run off-line algorithms on the stored metadata (e.g. to compute statistics about space usage...).

Metadata structure and exchange. As highlighted in Fig. 1, the system has to handle three kinds of data: results from analysis (performed in low and high-level analysis modules), queries and replies (mediated by the web-server and the knoweldge discovery module). In order to guarantee system consistency and compliance with standards, these three kinds of data share the same markup language, i.e. the eXtensible Markup Language (XML). In order to avoid sending raw XML documents over the network, every transferred data is wrapped in an RSS feed. The data exchange is thus reduced to the handling of an RSS flow.

3 Agent-Based Data Warehousing Architecture

As the way the metadata is generated within the architecture has been delineated, more attention can be given to the knowledge storage itself. The agent-based system architecture, and more especially the knowledge flow management and the data-oriented agents used to handle the metadata within the system are first presented in Sec. 3.1. The knowledge structure used for the storage task, as well as the DW storage specifications, are then addressed in Sec. 3.2.

3.1 Agent-Oriented System Description

As mentioned in Sec. 1, the system we propose is composed of two kinds of agents: *process-oriented* and *data-oriented*. As detailed below, *process-oriented agents* are in charge of the RSS and knowledge flow management while *data-oriented agents* are reactive entities incarnating knowledge pieces.

Process-oriented agents. Fig. 2 sketches the agent system overview and highlights the interactions among process-oriented agents. Note that the RSS flow arrows in the upper part of the figure are the links between the agent-based data warehousing and the rest of the architecture presented in Fig. 1.

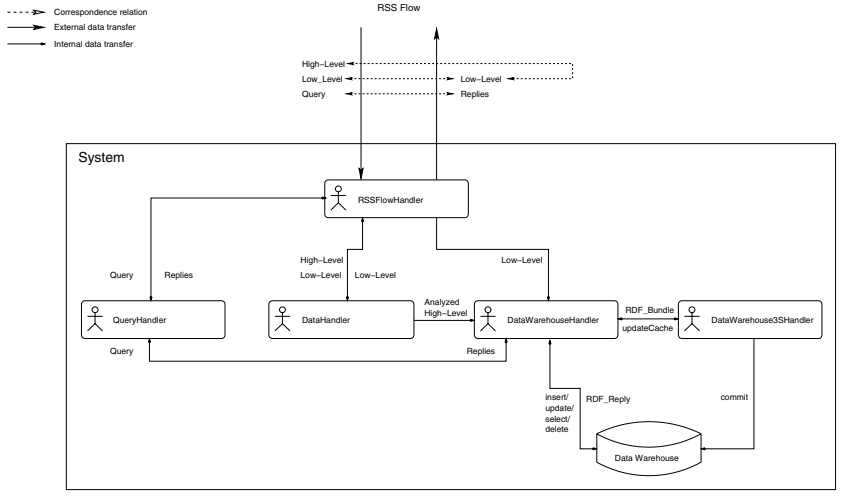


Fig. 2. Agent system architecture overview

As illustrated in Fig. 2, the roles of the different process-oriented agents are the following:

- `RSSFlowHandler` handles the data flow coming from the RSS publishers, and transmits the XML extracted from the RSS feeds to the concerned agents (e.g. unwraps and transmits XML to `DataHandler`, wraps/transmits XML replies from `QueryHandler` to the queries' emitters...).
- `DataHandler` transmits low-level analysis results to the `RSSFlowHandler` to obtain high-level analysis results. It also identifies, from high-level results, semantic links to add to formerly received metadata and transmits them to the `DataWarehouseHandler`.
- `DataWarehouseHandler` saves data in cache, and transmits *commit* requests to `DataWarehouse3SHandler`. It also removes committed data from cache after transaction completion. It is last in charge of running queries from the `QueryHandler` and returning results.

- `DataWarehouse3SHandler` asynchronously saves data on a non-volatile support, and confirms *commit* completion to the `DataWarehouseHandler`.
- `QueryHandler` handles query execution coming from the `RSSFlowHander` by maintaining a pending query queue. It transmits queries to and retrieves replies from the `DataWarehouseHandler`.

Data-oriented agents. In order to allow the efficient handling of data within the system, we developed dedicated data-oriented agents named *autotroph agents* (AA) [10]. These AAs form a set of agents corresponding to XML documents handled by the various components of the system. An AA does not have to be assimilated to a simple XML document containers as it embodies XML documents and enables the following operations: self-replication, i.e replicate its structure and content; self-reification, i.e. retrieve the knowledge stored in the DW and instantiate its structure with it; self-exportation to XML, i.e. generate an XML document corresponding to its structure; self-exportation to RDF/query XML and N-Triples, and optimisation of the graph in order to minimise reification time and storage space; updating its components; modifying its structure from new data; merging with other AAs of possibly different types; and applying a mask on the exported data.

Since the schemas of the XML files coming from low/high-level analysers are not *a priori* known, they have to be specified at system initialisation. The structures of the agents for both level are thus obtained at run-time by interpreting the low/high-level XML schemas[2], and exploiting the AAs adaptive capabilities. More precisely, when a new XML file is received, its type (low/high-level) is recognised and the matching structure is replicated. The structure of the replicated agent is adapted and instantiated with the structure/values enclosed in the XML. An AA is thus a dynamic, and self-managing structure considered as the smallest knowledge unit processable by the system. Therefore, these generic, context-independent and adaptive AAs allow to handle any kinds of XML meta-data in the system, without any restrictions on their format and/or content.

Indeed, AAs are dormant entities not actually aware of their environment evolutions. The events they handle take the form of Java method calls that cannot, rightly to our opinion, be considered as agent-level events. Nevertheless, they do not stop running after the execution of a specific action. They are also goal driven and interact much with their environment, notably during the reification phase. What highly differentiates them from classic structures such as vectors is that they are aware of their own structure and can completely modify and update it. Even though AAs cannot be considered as pure agents, we showed they represent much more independent and smart structures than traditional object approaches. Furthermore they are very similar entities to those addressed by the KidSim [18] and IBM agents [7]. Consequently, even if disputable, we classified them as data-oriented agents.

[2] The XML schemas are designed in *Relax NG* (REgular LAnguage for XML Next Generation), which is an XML schema definition language.

3.2 Data Warehouse Storage

As highlighted previously, a key feature of the monitoring system we propose is that the schema used for formatting the metadata is not *a priori* known, which rules out any *a priori* storage optimization or database schema definition. The XML meta-data handled by the system is thus considered as knowledge we want to structure, manage and store in an optimized and unsupervised way. To achieve such storage task, we resort to *Resource Description Framework* (RDF), a technology used in the semantic web to represent and share knowledge.

Stored knowledge schema. Main advantages of RDF are that stored knowledge is incrementaly built and that no schema need to be *a priori* determined before being instantiated. Another interesting feature of RDF is that duplicated literals in XML files (which are very frequent) are not duplicated in the DW. Last, syntactically different XMLs can express the same semantic information; indeed, while XML's primary function is to describe data, RDF's primary role is to describe knowledge. Consequently, using RDF to perform the storage task yields to a storage-independent data structure providing a more sophisticated representation of the knowledge than classical raw XML storage.

RDF graphs are defined as sets of triples, also called *statements*, of the form $\langle$ *subject predicate object* $\rangle$. The predicate is a directed arc going from the subject to the object. It can be seen as a property linking the object to its subject. For example, one way to represent the notion "Blob has id 5" in RDF is as a triple made of the subject "Blob", the predicate "has id", and the object "5". More formally [8], if we define U as an infinite set of RDF URI references, $B = \{N_i : i \in \mathbb{N}\}$ as a set of RDF blank nodes and L as an infinite set of RDF literals (e.g. string, int...), a triple $(s, p, o) \in (U \cup B) \times U \times (U \cup B \cup L)$ where s is a subject, p a predicate and o an object is called an RDF triple and a graph $G \subseteq (U \cup B) \times U \times (U \cup B \cup L)$ is an RDF graph. However, RDF graph-based approach does not prevent deep tree structures. To avoid following long paths in XML files, some optimisations on the structure of the schemas have to be performed. In this context, we propose to reduce the RDF graph to a depth of four nodes, no matter its initial schema. As highlighted in Sec. 4, this schema restructuring allows to significantly reduce knowledge retrieval costs.

3store. The storage of RDF graphs is commonly performed in triple stores. For this purpose, we rely on 3store [9] which is a MySQL backed RDF triple store implemented in C. The main advantages of 3store are that it performs a reduced set of inferences, benefits from optimisations realized in MySQL, fully supports the SPARQL (SPARQL Protocol And RDF Query Language) query language and appears to better support mass storage.

Cache. Aware of the huge amount of disk access to perform in such a real-time system, keeping as much information as possible in main memory is of the utmost importance. The cache memory we use is implemented by means of a red/black tree guaranteeing query, insertion, and deletion time of $O(log\ n)$. Each node in

the tree is an AA whose access, comparison, and deletion key is its unique id determined by a 64-bit integer.

For brevity reasons, the selection process for the most suitable engine for RDF physical storage, the concerns and specifications of the RDF-graph depth reduction, the reification process associated to XML document retrieval, the query/reply mechanisms implemented, the cache dimensioning and purging strategies, as well as technical motivations related to the agent-oriented data warehousing in its whole are not presented here. Interested readers should refer to [10] for more detailed information.

4 Performance Evaluation

In this section, we present results obtained while measuring the system performance through the technical/computational point of view; adaptiveness capabilities are also highlighted with respect to the current achievements in the project. Note that the Video Content Analysis (VCA) used for the evaluation are not themselves evaluated in this section, but in the referred papers.

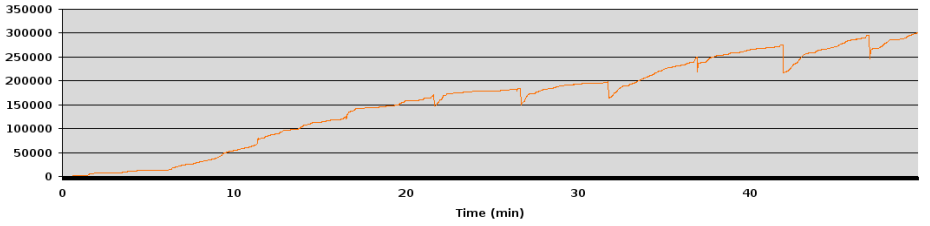
4.1 Storage/Retrieval Performance

In a first part, a performance evaluation was performed using low-level VCA aiming at detecting moving objects (people) in the video streams. Due to the status of the high-level stubs (under development), XML files produced were sent over the system to a high-level simulator, which was randomly sending XML results back at a real-scale rate (from 5 to 25 results per second). While this configuration does not fully correspond to the final one, it already allows to validate the real-time aspect and the storage capabilities of the system, as well as the scalability of the architecture.

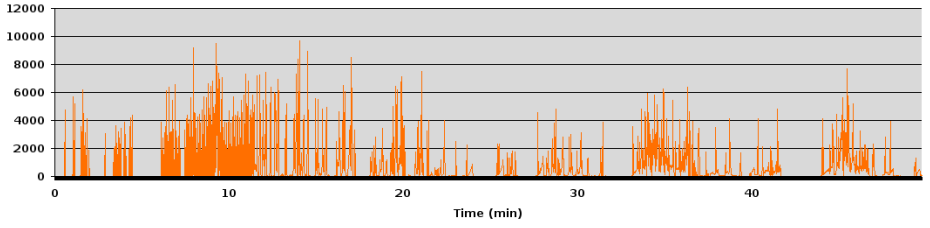
For this first set of experiments, different moments of the day with varying crowd density levels (e.g. rush hour, evening...) from Roma metro CCTV system were used; the standard for formatting metadata was a subset of the INRIA (surveillance-oriented) ontology [2]. Tab. 1 summarises the main results obtained after 50 minutes of video processing at 5 frames per second. Queries were based on selected values appearing in analysis results (e.g. people ids, time

Table 1. Storage/retrieval performance

System features	Value		AA capabilities	Value
DW size	291 MB			
Query (from cache)	3 ms		Agent replication	0 ms
Query (from 3store)	50 ms		Agent initialisation	5 ms
Reification (from cache)	0 ms		Agent → XML	5 ms
Reification (from 3store)	40 ms		Agent → RDF	5 ms
Insertion (in cache)	0 ms			
Insertion (in 3store)	70 ms			



(a) Evolution of data warehouse's size over time.



(b) Evolution of data warehouse's insert number over time.

Fig. 3. Statistics of data warehouse's storage

intervals...). Values related to query, reification and insertion are average values for a single XML document.

As highlighted in Tab. 1, the system performance is markedly improved when using the cache memory. Moreover, this evaluation allows to measure the interest of the RDF optimisation; the reification time from 3store with a non-optimised RDF structure (not reported in Tab. 1) was about 1600 ms, whereas the one with our optimised RDF structure (depth limited to four nodes) is about 40 ms, i.e. 40 times faster. Other basic operations, i.e. replication, initialisation, and conversions, take only some milliseconds to complete, which ensures to fullfill the real-time constraints. Fig. 3 provides corresponding statistics from the DW part.

During the trial, the system was almost continuously receiving meta-data coming from the content analysis process; as shown in Fig. 3-(a), the memory usage of the DW (DW disk storage) has thus grown throughout the trial, exhibiting a ramp-up behavior. Moreover, analysis of Fig. 3-(b) shows that data rows were inserted in large bursts separated by idle time. This observation confirms that the system is caching received data, then writing them periodically to the database. While such process may not be the most optimal in terms of DW throughput, it allows to control the front-end system workload, which is of primary importance. So as to confirm these observations, a similar experiment was performed for a longer period (around seven hours); corresponding statistics are provided in Fig. 4.

We can see in Fig. 4 that the ramp-up behavior observed during the first hour of the experiment is clearly lessened over the time. Indeed, this point highlights the fact the stored knowledge is incrementaly built; more precisely, when most of the possible RDF node' values and their corresponding hashed value (i.e. all possible XML elements, attributes and corresponding values) have been stored

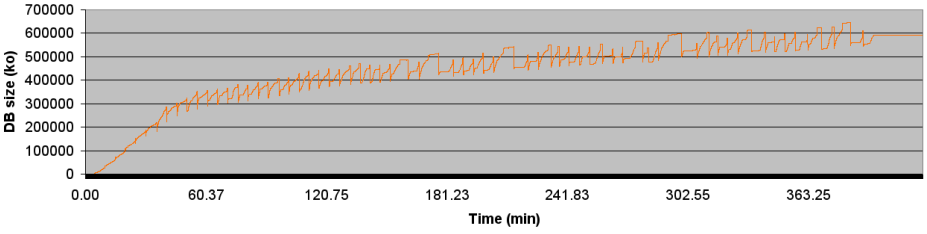


Fig. 4. Evolution of data warehouse's size over seven hours

once, only the triple they are related to has to be stored. In other words, once a defined position of a bounding box has been stored, next XMLs enclosing this position will just refer to the firstly stored position, which represent a significant storage cost reduction and point out the interest of using RDF for the storage task. Furthermore, another noticeable point is the sawtooth wave form of the DW size evolution; this can be easily explained taking into account the memory allocation at the database level; indeed, when additional memory allocation is needed, the DW content is compressed; this phenomena yields to a significant decrease of the DW size at each memory reallocation.

Regarding the DW size itself, the entire trial yield to a 700 Mo database for approximatively seven hours of processing; while this could first appear inhibitive, it worth noting that the used schema was voluntarily very verbose, and that the quantity of stored data was overly huge (and useless), so as to assess the system scalability for longer and more relevant metadata mass storage.

4.2 Adaptiveness Evaluation

Within the context of the project, the system adaptiveness was then validated by building the real-scale configuration corresponding to Fig. 1. The used architecture contained five modules including a module for audio/video data acquisition and streaming. These five modules and the system itself were mapped on a Local Area Network (LAN) of a dozen of computers.

In a first time, the module responsible for the sensing and streaming of the data coming from the audio/video sensors was set up. A first VCA module was used to segment the video frames into a static background and moving foreground objects [6]. A second one was then activited to track detected objects over time [1] (see Fig. 5-(a)). A "left-luggage" module was last applied, to detect objects appearing in the scene and identifying whenever they remain stationary for a certain duration [19] (see Fig. 5-(b)). Regarding the audio part, an audio analysis was employed to detect abnormal audio events [5].

During the architecture building, the *Relax NG* schemas characterizing the low/high level analysis have been constinuously updated to support newly added modules in the architecture. At each module adding, the storage system was restarted using the newly updated schemas. This building allowed to assess the efficient storage and fast retrieval capabilities of the system.

(a) Long-term tracking (Roma site). (b) Left luggage detection (Torino site).

Fig. 5. Examples of VCA modules integrated within the system architecture

5 Conclusion

In this work, an innovative system for video analysis results handling in distributed environments has been presented. The main contribution lies in the use of an agent-based data warehouse to manage the audio/video analysis metadata. More precisely, the proposed system is context-independent and allows to handle metadata without being aware if its structure before initialisation, thus providing an open, generic and flexible monitoring system. Dedicated kind of agents called *autotroph agents* have been also proposed to improve the way the knowledge is handled within the system. Moreover, RDF graphs were used to efficiently manage and store the huge amount of data available. Several optimisations dealing with RDF graph depth and cache memory usage were also implemented to enhance the system reactivity. Local tests conducted with real metadata and in a real-scale configuration demonstrated the efficiency and adaptiveness of the proposed system.

Future works will include real-scale evaluations within a complete monitoring environment, i.e. including different analysis processes running in parallel and GUI subsystems accessing/querying the system at the same time. The exploitation of the stored knowledge will also be addressed by linking the storage system with advanced algorithms of knowledge modelling and discovery available in the project consortium [14].

Acknowledgements

The work presented here is partially supported by the European Commission under the 6th Framework Program through the IST FP6-027231 CARETAKER project. For further information about the CARETAKER project, please visit `http://www.ist-caretaker.org/`.

Part of this work has also been achieved in the framework of the ITEA SERKET project (label #04005), funded by the Belgian-Walloon DGTRE.

References

1. Avanzi, A., Bremond, F., Thonnat, M.: Tracking multiple individuals for video communication. In: Proc. of IEEE Int. Conf. on Image Processing (2001)
2. Bremond, F., Maillot, N., Thonnat, M., Vu, T.: Ontologies For Video Event. Tech. report 5189, INRIA Sophia Antipolis, France (2004)
3. Chen, H., Finin, T., Joshi, A., Kagal, L., Perich, F., Chakraborty, D.: Intelligent Agents Meet the Semantic Web in Smart Spaces. IEEE Internet Computing (2004)
4. Collins, R., Lipton, A., Kanade, T., Fujiyoshi, H., Duggins, D., Tsin, Y., Tolliver, D., Enomoto, N., Hasegawa, O.: A System for Video Surveillance And Monitoring. Tech. report CMU-RI-TR-00-12, Robotics Institute, Carnegie Mellon University (2000)
5. Couvreur, L., Bettens, F., Hancq, J., Mancas, M.: Normalized Auditory Attention Levels for Automatic Audio Surveillance. In: Int. Conf. on Safety and Security Engineering (2007)
6. Desurmont, X., Messen, J., Parisot, C., Delaigle, J.F.: A step to cognitive vision systems for common videosurveillance. In: Int. Workshop on Image Analysis for Multimedia Interactive Services (2005)
7. Gilbert, D., Aparicio, M., Atkinson, B., Brady, S., Ciccarino, J., Grosof, B., Orsquo-Connor, P., Osisek, D., Pritko, S., Spagna, R., Wilson, L.: IBM intelligent agent strategy. White paper, IBM Corporation (1995)
8. Gutierrez, C., Hurtado, C., Mendelzon, A.O.: Foundations of semantic web databases. In: ACM Symp. on Principles of Database Systems, pp. 95–106 (2004)
9. Harris, S., Gibbins, N.: 3store: Efficient bulk RDF storage. In: Proc. of the Int. Workshop on Practical and Scalable Semantic Systems, pp. 1–15 (2003)
10. Hubaux, A.: On the Use Of Semantic Web Agents in Video Analysis Sharing. FUNDP Namur, Belgium (2007)
11. Inmon, W.H.: Building the Data Warehouse. Wiley Computer Publishing, Chichester (2002)
12. Karunanidhi, A., Doermann, D.: Survey of Content Based Access To Surveillance Video. Tech. report, Laboratory for Language and Media Processing (2002)
13. Lienard, B., Desurmont, X., Barrie, B., Delaigle, J.-F.: Real-time high-level video understanding using data warehouse. In: SPIE Symp. on Electronic Imaging (2006)
14. Patino, J.L., Benhadda, H., Bremond, C.E., Thonnat, M.F.: Video-data modelling and discovery. In: IET Int. Conf. on Visual Information Engineering (2007)
15. Perrott, A.J., Lindsay, A.T., Parkes, A.P.: Real-time multimedia tagging and content-based retrieval for CCTV surveillance systems. In: Proc. of SPIE Internet Multimedia Management Systems, vol. 4862, pp. 40–49 (2002)
16. Rangaswami, R., Dimitrijevic, Z., Kakligian, K., Chang, E., Wang, Y.-F.: The SFinX Video Surveillance System. In: IEEE Conf. on Multimedia and Expo. (2004)
17. Shu, C.F., Hampapur, A., Lu, M., Brown, L., Connell, J., Senior, A., Tian, Y.: IBM Smart Surveillance System (S3): a open and extensible framework for event based surveillance. In: IEEE Conf. on Advanced Video and Signal Based Surveillance (2005)
18. Smith, D.C., Cypher, A., Spohrer, J.: Kidsim: programming agents without a programming language. Commun. ACM 37(7), 54–67 (1994)
19. Smith, K., Quelhas, P., Gatica-Perez, D.: Detecting Abandoned Luggage Items in a Public Space. In: IEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance (2006)
20. Wijnhoven, R.G.J., Jaspers, E.G.T., de With, P.H.N.: Flexible Surveillance System Architecture for Prototyping Video Content Analysis Algorithms. In: Proc. of the SPIE Conf. on Real-Time Imaging (2006)