Dynamic Video Surveillance Systems Guided by Domain Ontologies

Juan C. SanMiguel and José M. Martínez

Video Processing and Understanding Lab, Universidad Autónoma de Madrid, E-28049 Madrid, Spain
{JuanCarlos.SanMiguel, JoseM.Martinez}@uam.es

Keywords: Domain knowledge, System knowledge, ontologies, dynamic framework, video surveillance.

Abstract

In this paper we describe how the knowledge related to a specific domain and the available visual analysis tools can be used to create dynamic visual analysis systems for video surveillance. Firstly, the knowledge is described in terms of application domain (types of objects, events... that can appear in such domain) and system capabilities (algorithms, detection procedures...) by using an existing ontology. Secondly, the ontology is integrated into a framework to create the visual analysis systems for each domain by inspecting the relations between the entities defined in the domain and system knowledge. Additionally, when necessary, analysis tools could be added or removed on-line. Experiments/Application of the framework show that the proposed approach for creating dynamic visual analysis systems is suitable for analyzing different video surveillance domains without decreasing the overall performance in terms of computational time or detection accuracy.

1 Introduction

Nowadays, advanced video surveillance systems are expected to work in dynamic and different environments or domains (but related) allowing the on-line addition or removal, when necessary, of services and analysis capabilities. During the last years, the description of the knowledge relevant to each domain and the development of knowledge-based automatic video analysis tools have been transformed into topics of greatest interest in the video-surveillance community to analyze different environments [1,2,3]. In this context, ontologies have been used to represent the relevant prior information of each domain with the objective of adding expressiveness and reasoning capabilities to the system that uses them, and making the encoded knowledge usable by people and automatic systems. In video-surveillance applications, the use of ontologies involves two basic problems: how to build and represent the entities of the ontology and how to use the ontology to enhance the analysis performed. Section 2 discusses the existing approaches.

In this paper, we propose to use the knowledge related to the domain and system to create dynamic visual analysis systems. Firstly, an existing ontology description focused on event detection for video-surveillance [4] is used to define the relevant information for each domain and the analysis capabilities. Secondly, the ontology is integrated into a framework to create the visual analysis systems for each domain. Then, the ontology is used to create the visual analysis system by inspecting the relations between the entities defined in the ontology. It consists in a selection of the most appropriate visual analysis tools to analyze the domain and the determination of their execution order. Additionally, new analysis capabilities or existing ones can be added or removed respectively by applying similar operations. The proposed approach is demonstrated in the Underground video-surveillance domain showing its powerfulness for creation of dynamic visual analysis systems.

The paper is structured as follows: section 2 overviews the related state-of-the art, section 3 describes the ontology entities used, section 4 describes the proposed framework that creates and updates the visual analysis systems, section 5 describes the domain-based creation/update process and section 6 describes its application for the Underground video-surveillance domain. Finally section 7 raises some conclusions and proposes lines of future work.

2 Related work

The use of ontologies in video analysis systems involves two basic problems: how to use the ontology to represent the domain knowledge and to improve the analysis performed.

The first problem, also known as *knowledge sharing and representation*, has been the most widely explored by the video research community. The common idea is to define a basic structure for the analysis problem (e.g., event detection) identifying its key concepts (e.g., objects and events). Then, it is extended with the prior knowledge of each domain. For instance, [5] proposed an ontology representation language (VERL) and markup language (VEML) as an initiative for dynamizing the use of ontologies for describing and annotating events in video analysis. Similarly, [7] proposed an ontology to describe physical objects and video events that can be observed in the scene by defining a class hierarchy of objects/events. More recently, [13] has proposed a framework to integrate the analysis and evaluation of video-surveillance systems by using an event detection ontology. On the other hand, there are some approaches more focused on the description of each domain instead than on the overall structure of the ontology for the Video-surveillance domain [3,8,14,15], the Meeting domain [16] and the Soccer domain [11]. OWL is the most popular language between all the

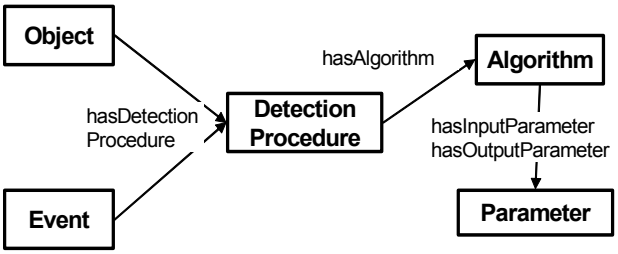

Figure 1. Entity relations exploited in the proposed work.

studied approaches. More recently, the integration of domain and system knowledge have been proposed in order to provide a detailed description of the relationships between the domain entities and their detection algorithms for object detection [10,11] or event detection [12]. However, there are some approaches that use ontologies without a detailed definition of events and objects. For instance, the annotation of high-level video-surveillance concepts is proposed in [17].

The second problem, also known as *knowledge-based analysis*, has been similarly approached by the video research community. Some specific rules are introduced to infer/derive new knowledge due to the limitation of ontologies to describe other forms of knowledge (e.g., event detection). The defined rules are linked with the ontology concepts in order to increase the system capabilities. Description Logic (DL) [1,11], Fuzzy Logic [2], F-logic [10] and Semantic Web Rule Language (SWRL) [9] are the rule languages most frequently used. However, there some approaches that define their own proprietary approaches to describe the logical constraints applied [4,7,18]. Knowledge-based analysis by employing rule languages has been used for two objectives: complex event detection and the deployment of automatic systems. Complex event detection approaches are not discussed in this paper because they are out of the scope of the paper. The development of fully automatic analysis systems assisted by ontologies has been used to analyze different domains that share a common knowledge basis. To our knowledge few analysis systems with a generic architecture have been proposed to automatically analyze different domains. The detection of objects [10,11] and events by using a ontology-based Dynamic Bayesian Network [12] are the main contributions in this area. The main drawback of these approaches is the absence of a detailed description of the relation between domain knowledge and available analysis tools making their use limited for automatic analysis of different domains. Additionally, domain and system engineering parts of these approaches are very dependent causing their design more difficult for the analysis of different domains. In conclusion, most of the approaches in the reviewed literature use the ontology for describing the prior knowledge related to the domain modelled. OWL is the most common used language. It can be explained due to the high amount of editors available. Deployment of automatic analysis systems require the addition of rule languages in order to include the necessary expressiveness as the ontology concepts can not encode some types of prior knowledge. Existing approaches do not provide an automatic solution to the problem due to they do not describe in detail the relation between domain knowledge and available analysis tools.

3 Ontology entities related with this work

In order to realize the knowledge-based determination of the workflow, we have selected a state-of-art ontology description that is able to define domain and system knowledge [4]. It is structured into an upper ontology (basic classes and their specializations) that describes the structure of each knowledge type (domain and system) and the domain-specific knowledge defined to model each domain/system. The most relevant entities of the ontology that are exploited in the work proposed in this paper are:

- Entity *Object*: represents the objects in the scene. Each *Object* instance is related to detection procedures instances by the *hasDetectionProcedure* property.
- Entity *Event*: represents any event to be detected in the video content. Each *Event* instance is related to appropriate detection procedures instances by the *hasDetectionProcedure* property.
- Entity *Algorithm*: represents the available visual analysis tools in the system. They are used in the detection procedures defined for the detection of objects and events. Each *Algorithm* instance is related to input/output parameter instances by the *hasInputParameter*/ *hasOutputParameter* property.
- Entity *DetectionProcedure*: represents the available processing schemes in the system for detecting the concepts described in the ontology. Each *DetectionProcedure* instance is related to appropriate algorithms instances by the *hasAlgorithm* property.
- Entity *Parameter*: represents the different inputs and outputs of the algorithms available in the system. It is sub-classed according to the available algorithms.

Figure 1 depicts the exploited relations between the ontology entities relevant to the proposed work.

4 Framework overview

A complete framework has been designed to create/update the visual analysis system for each domain. It performs two main tasks: ontology interpretation and video analysis. It is able to process ontologies in order to create/update the visual analysis system determining its analysis tools (detection procedures and algorithms) and their execution order for detecting the relevant domain concepts defined.

The proposed framework is composed by different functional modules (see Figure 2). The *Ontology Database Module (ODM)* is in charge of providing the available domain ontologies and it is composed by a server and a database with the available domain ontologies. The *Interpretation and Management Module (IMM)* is in charge of interpreting the selected domain ontology, then combine it with the defined rules (for analysis or guidance) and to request the execution of algorithms (to the *Algorithm Server Module*). The *Algorithm Server Module (ASM)* is in charge of providing the processing capabilities to the entire framework. It makes the visual analysis tools usable through a server. The *.Algorithm Database Module (ADM)* indexes the available visual analysis tools and stores their compiled versions in order to provide the processing capabilities. Its structure is similar to

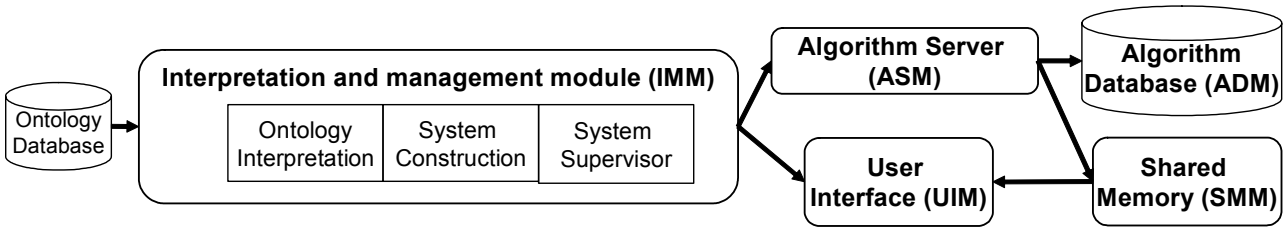

Figure 2. Architecture of the System proposed for the creation and update of visual analysis frameworks.

the *Algorithm* entity described in the ontology. The *Shared Memory Module (SMM)* is in charge of maintaining the necessary memory resources for the creation, execution and destruction of the algorithms. Additionally, it is connected with the *UIM* for display purposes. Finally, the *User Interface Module (UIM)* is in charge of the interaction related with the final user or consumer of the analyzed content providing input from the user (e.g., domain to analyze) and output to the user (e.g., video descriptions…)..

The sequence of operations performed by the framework in order to analyze video sequences from a specific domain is:

1. *Initialization*. The *UIM* gets the necessary data for the analysis (e.g., domain to analyze, operation mode...) and configures the *IMM*. Then, the *IMM* requests to the *OSM* the corresponding domain ontology.
2. *Visual analysis framework creation.*
   (a) The *IMM* requests to the *ASM* the analysis tools available for the selected domain by using the data indexed in the *ADM*. Finally, the instances of the existing visual analysis tools are created and linked appropriately.
   (b) The *IMM* interprets the ontology in order to calculate the necessary resources (parameters) and allocate memory for them in the *ASM*. Instances of the parameters are created and linked with the *Algorithm* instances.
   (c) The *IMM* interprets the ontology in order to select the necessary visual analysis tools between the available ones and their computation order. Then, this information is sent to the *SMM* (via the *ASM*) for the algorithm resources creation. This process is described in section 5.
3. *Analysis*. Finally, the *IMM* begins the sequential execution of the visual analysis tools via execution requests to the *ASM*. The analysis is performed until the video file has been finished or the system is turn off (for live on-line video analysis). During the analysis, the update of the analysis system (addition or removal) is performed by applying the ontology inspection process proposed in section 5.3.

The main advantage of this framework is the integration of ontology and video analysis tools and its scalability and flexibility. Any domain described with the used ontology can be analyzed with the proposed framework.

5 Ontology-based analysis system creation

In order to provide an automatic mechanism for the creation and update of the visual analysis system for different domains, we propose to use the properties of the defined ontology entities to determine the best visual analysis tools to apply and their associated execution order. This is performed by exploiting the properties of the *Algorithm*, *DetectionProcedure* and *Parameter* instances created and it is divided in two stages: visual analysis tools selection and execution order determination.

5.1 Visual analysis tools selection

Visual analysis tool selection is directly performed by inspecting the properties of the sub-entities of the *Event* and *Object* entities defined for each domain. This phase should be considered as the integration of domain knowledge and system knowledge. This selection is automatically computed each time the framework is requested to analyze a specific domain and it is based on rules that exploit the transitivity properties between the entities defined in the ontology. These rules define the mapping between the visual analysis tools and the relevant entities to be detected in the modelled domain. Among the different choices available in the literature, we have decided to use F-Logic due to the rules are easy to describe and understand using this language. Firstly, we have defined the rules to select all the necessary detection procedures to analyze a specific domain. They are represented as follows:

1. IF an Event E_1 has objects $O=\{O_1,...,O_N\}$ as a part of its description AND objects $O=\{O_1,...,O_N\}$ have detection procedures $DP_O=\{DP_{O1},...,DP_{ON}\}$ respectively THEN E_1 *hasDetectionProcedure* $DP_{E_1}=\{DP_{O1},...,DP_{ON}\}$
2. IF an Event E_1 has sub-events $SE=\{SE_1,...,SE_N\}$ as a part of its description AND sub-events $SE=\{SE_1,...,SE_N\}$ have detection procedures $DP_{SE}=\{DP_{SE1},...,DP_{SEN}\}$ respectively THEN E_1 *hasDetectionProcedure* $DP_{SE}=\{DP_{SE1},...,DP_{SEN}\}$
3. IF an Object O_1 has sub-objects $SO=\{SO_1,...,SO_N\}$ as a part of its description AND sub-objects $SO=\{SO_1,...,SO_N\}$ have detection Procedures $DP_{SO}=\{DP_{SO1},...,DP_{SON}\}$ respectively THEN O_1 *hasDetectionProcedure* $DP_{O_1}=\{DP_{SO1},...,DP_{SON}\}$

Then, another rule is defined in order to select the *Algorithm* entities to apply from the selected detection procedures. This rule is applied in pairs to all the selected detection procedures as follows:

4. IF a Detection Procedure DP_1 has algorithms A_1, A_2 and A_3 as a part of its description AND a Detection Procedure DP_2 has algorithms A_3, A_4 and A_5 as a part of its description THEN the algorithms to use are A_1, A_2, A_3, A_4 and A_5.

Finally, the selected Algorithm instances conform the set of visual analysis tools to be executed and their properties are used to compute their execution order

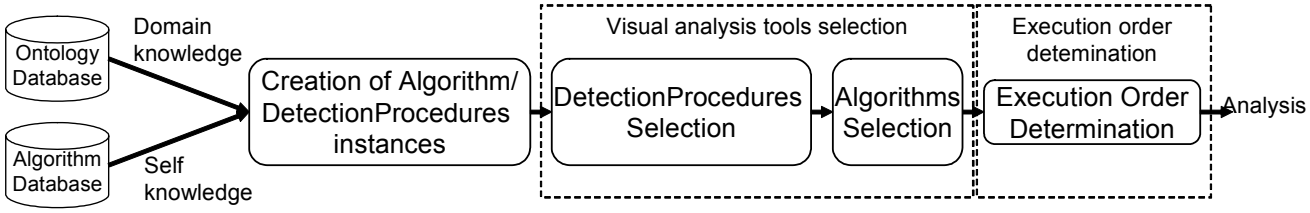

Figure 3. Procedure applied to select the appropriate visual analysis tools and execution order for each domain.

5.2 Execution order determination

After the selection of the *Algorithm* instances to be applied during the analysis of the domain and system knowledge, a determination of the execution order execution is performed in order to determine the sequential order application of each algorithm. Its computation is performed by inspecting the properties *hasInputParameter* and *hasOutputParameter* of each instance. The basic idea is to define a set of input parameters, select the algorithms that can be used with this input set and study the possible relations between the selected algorithms. Priorities and sub-priorities are assigned depending on their relations. Then, the set of input parameters is extended with the output parameters of the selected algorithms and the procedure is repeated with the new input set. This process is performed from the minimum set of inputs, composed by the input image (named *FRAME-RGB* in the ontology), until the list of *Algorithm* instances selected is finished. The full execution order determination procedure is described in Algorithm 1.

5.3 On-line system update (insertion/removal)

The on-line insertion and removal of new analysis tools into the system is performed by the user or other applications via the *UIM* module. Its main advantage is that the tools are not destroyed and only their execution order is changed. The insertion operation is performed by adding the new data (system and domain knowledge), creating the instances corresponding to the new data and computing the execution order of each new visual analysis tool added. Finally, the execution order of existing analysis tools applied after the new insertion (with higher execution order) is increased. Removal operation is performed by deleting the corresponding instances and computing the execution order of all remaining tools (creating a new visual analysis system).

6 Application of the framework

In order to show the applicability of the proposed approach, we have tested it in Underground video-surveillance domain by defining the necessary knowledge, creating and updating the visual analysis system. The update operations are based on the prior information about the two used datasets: PETS2006 dataset (available at http://pets2006.net/) and AVSS2007 dataset (available at http://www.avss2007.org/). The results obtained for these operations are depicted in Figure 4. Moreover, a computational cost comparison with a similar one (from now on base system) without dynamic capabilities and non ontology-based has been performed.

Algorithm 1. Execution order determination

Definition. $\mathbb{P}$ and $\mathbb{A}$ are two sets that represent all the parameter/algorithm instances respectively defined in the ontology. p_j and a_i describe a Parameter or an Algorithm respectively. S, A_I and A_F are three sets that contain selected algorithm instances and I represents a set that contains selected parameter instances. The execution order is represented by o. Additionally, some useful functions are defined:

- $Input(a_i) = \{p_j \subset \mathbb{P} / a_i \; hasInputParameter \; p_j\}$
- $Output(a_i) = \{p_j \subset \mathbb{P} / a_i \; hasOutputParameter \; p_j\}$
- $AssignOrder\ (o, a_i)$ assigns the execution order o to the algorithm a_i

Stages of the algorithm.

1. Set $S = \{\varnothing\}$, $A_F = \{\varnothing\}$ and $o = 1$
2. Set $I = \{FRAME - RGB\}$ (raw image as initial input)
3. Iteratively apply until $S \equiv \mathbb{A}$
   a) $A_I = \{a_i \subset \mathbb{A} / Input(a_i) \equiv I\}$
   b) IF $a_i \subset A_I$ has not input parameters from other $a_j \subset A_I (i \neq j)$ THEN $A_F = A_F \cup A_I$
   c) IF $card(A_F) = 1$ THEN $AssignOrder\ (o, A_F)$ ELSE use rules for collision*
   d) Set $o = o + 1$
   e) Set $S = S \cup A_F$ and $A_F = \{\varnothing\}$
   f) Set $I = \{FRAME - RGB, Output(S)\}$

***RULES FOR COLLISION**

They are defined to determine the sub-order of algorithms with the same execution order. Firstly, the algorithm type is identified:

- *Filtering algorithms:* $a_i \subset \mathbb{A} / Input(a_i) \equiv Output(a_i)$
- *Processing algorithms type 1:* $a_i \subset \mathbb{A} / Input(a_i) \neq Output(a_i)$ AND $Output(a_i) \subset Input(a_i)$
- *Processing algorithms type 2:* $a_i \subset \mathbb{A} / Input(a_i) \neq Output(a_i)$ AND $Output(a_i) \not\subset Input(a_i)$

Then, three rules are applied to determine the sub-order:

- Rule 1: Filtering algorithms are applied after each input or output modification (the execution order o is increased each time the algorithm is assigned)
- Rule 2: Processing algorithms type 1 are analyzed in first position to assign their execution order (high priority)
- Rule 3: Processing algorithms type 1 are analyzed in second position to assign their execution order (low priority)

6.1 Domain knowledge

For the *Object* entity, the following objects have been modelled in the domain ontology: person, group of persons, metro train, portable object, door, window, wall and zone. Contextual objects (door, window, wall and zone) are provided as prior information by manually annotating them in the first video frame. For the *Event* entity, we have described a limited number of the events of interest in the domain. The simple and single events modelled are: *EnterZone*, *LeaveZone*, *GetObject* and *PutObject*. The complex and single events modelled are: *BecomedStationaryObject*, *AbandonedObject* and *StolenObject*. For the *Parameter* entity, all the available algorithms in the database are inspected and their corresponding inputs/outputs are defined in the domain ontology. Then, their respective instances are created. For the *Algorithm* entity, we have described the common algorithms used in this domain. They are: foreground analysis (divided into pre/post-processing, shadow and foreground detection methods), tracking analysis (divided into connected components, Kalman and Particle filter for blobs or specific objects), blob classification (people and train detection methods) and event analysis routines (divided into trajectory-based and single image-based). Finally, instances of the available algorithms are created and linked with the respective instances of the *Parameters* entities by using the *hasInputParameter*/*hasOutputParameter* properties. For the *DetectionProcedure* entity, the entities to describe the detection procedures of the domain ontology entities (*Object* and *Event)* are defined. Finally, their respective instances are created and linked with the *Algorithm* instances by using the *hasAlgorithm* property. Moreover, the *DetectionProcedure* type is also assigned in the corresponding *Object* and *Event* entities by using the *hasDetectionProcedure* property.

6.2 Visual analysis system creation

Firstly, the *hasDetectionProcedure* property of the Object and Event entities defined in the domain ontology is examined by using the first three rules defined in sub-section 5.1. Then, all *DetectionProcedures* are listed and the repeated ones are eliminated. Finally, algorithm selection is easily performed by applying the fourth rule defined in sub-section 5.1 to all the *DetectionProcedures* listed. As a result of this procedure, the following algorithms are selected: foreground detection, shadow elimination, noise elimination, blob tracking, people detection and the corresponding algorithms to detect the events modelled in the domain.

6.3 Execution order determination

To compute the order execution of the *Algorithm* entities, the *hasInputParameter* and *hasOutputParameter* properties are examined as described in sub-section 5.2. Firstly, the selected algorithms that can be applied using the raw frame (input *FRAME-RGB*) are examined. As a result, foreground detection is selected as the first algorithm to apply. Then, shadow detection and noise removal are selected in the second phase. Rules for collision are applied to determine their execution order. The noise removal and shadow detection algorithms are identified as filtering and type 1 algorithms respectively. Then, noise removal is firstly applied (Rule 1) in order to filter the input, then shadow detection is applied (Rule 2). Finally, noise removal is again applied in order to filter the shadow analysis (Rule 1). A third phase selects the blob tracking and the classification methods (people, group and vehicle). These algorithms are identified as type 2 so their execution order is the same (they can be executed in any sub-order). A fourth phase selects event detection algorithms corresponding to simple events. These algorithms are identified as type 2 so their execution order is the same. Finally, event detection routines for complex events are selected with the same execution order (algorithms type 2). Figure 4 depicts the final results obtained for this domain.

6.4 On-line system update (insertion/removal)

In order to simulate an on-line system update, we have decided to apply one insertion and one removal into the proposed framework. Firstly, a removal of the train detection method is performed for the sequences without trains (PETS2006 dataset). The related information in the ontology (the train *Object*) is eliminated and the system is updated. The insertion is performed as described in section 5.3 by adding a Group detection algorithm and linking it with the Group *Object* for the AVSS2007 test sequences.

6.5 Computational cost comparative evaluation

A comparison has been performed in order to test the performance of the proposed approach. The base system has been designed "by hand" with the same analysis capabilities being its domain specialization and no possible reconfiguration its main drawbacks. Due to both systems have identical analysis capabilities, the comparative evaluation is only performed in terms of computational cost (event detection results are the same). Both system have been implemented in C++ using the OpenCV library (t http://opencvlibrary.sourceforge.net/) using the OWL Protegé API to manipulate ontologies (available at http://protege.stanford.edu/). Tests were executed on a Pentium IV (2.8GHz) and 1GB RAM (for base framework) and two similar PCs connected through a GigE network (for the proposed framework). The comparative results are summarized in Table 1. It can be observed that the creation/update time is higher in the proposed framework than in the base one due to the proposed system apply the process described in section 5 for each system creation/update. It also depends on the amount of information encoded in the ontology. Additionally, the analysis/display of each frame is slightly increased in the proposed system due to its distributed configuration. It can be considered as inappreciable if the same computer is used.

System	Framework creation/update	Avg per frame	Display
Base	10/0	32,5	9
Proposed	300/30	33,5	9,3
Difference	+3000/300%	+2,98%	+3,2%

Table 1. Computational time comparative results (ms)

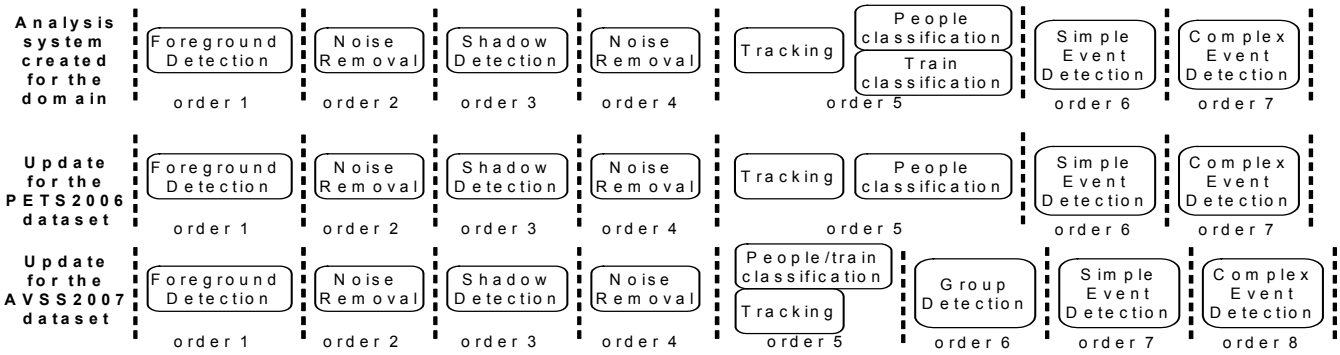

Figure 4. Visual analysis system created for the Video Underground surveillance domain and updated for the selected datasets.

7 Conclusions and Future work

In this paper, we have presented how the formalization of knowledge relevant to video analysis in a specific domain can be used to create dynamic visual analysis frameworks. The framework creation/update is performed by analyzing the relations between the entities defined for each domain ontology and the visual analysis tools available. This process is focused on two aspects: selection of the algorithms to apply and determination of their execution order. Experimental results show that the proposed system operates at the same performance level as the base system adding a low time delay (~3%) in the processing of each frame. The main advantage of the proposed approach is its adaptive capability to analyze different domains and its on-line reconfiguration. Moreover, the design of such kind of frameworks is separated in two parts: domain-knowledge-related and algorithmic-related parts. Domain experts and algorithm designers can focus their efforts in the development of more accurate models or algorithms respectively.

Future work includes an extension of the ontology in order to support complex processing schemes like feedback strategies (in this work only sequential mode is supported), to incorporate user preferences into the framework creation and to include failure component management policies or different distributed configurations will be explored.

Acknowledgments

This work was partially supported by the Spanish Administration agency CDTI (CENIT-VISION 2007-1007), by the Spanish Government (TEC2007- 65400 SemanticVideo), by the Comunidad de Madrid (S-050/TIC-0223 - ProMultiDis), by Cátedra Infoglobal-UAM for "Nuevas Tecnologías de video aplicadas a la seguridad", by the Consejería de Educación of the Comunidad de Madrid and by The European Social Fund.

References

[1] N. Maillot, M. Thonnat, and A. Boucher, "*Towards ontology based cognitive vision*", MVA, 16(1):33-40, 2004.

[2] C. Town, "*Ontological inference for image and video analysis*", MVA, 17(2): 94–115, 2006.

[3] L. Snidaro, M. Belluz, and G. Foresti, "*Domain knowledge for surveillance applications*," Proc. of FUSION, pp. 1–6, 2007.

[4] J. SanMiguel, J. Martinez, and A. Garcia, "*An ontology for event detection and its application in surveillance video*," Proc. of IEEE AVSS, pp. 220-225, 2009.

[5] A. Francois, R. Nevatia, J. Hobbs, R. Bolles, and J. Smith, "*Verl: an ontology framework for representing and annotating video events,*" IEEE Trans. on Multimedia, 12(4):6–86, 2005.

[6] N. Voisine, S. Dasiopoulou, F. Precioso, V. Mezaris, I. Kompatsiaris, and M. Strintzis, "*A genetic algorithm-based approach to knowledge assisted video analysis*," Proc. of IEEE ICIP, 3:441–444, 2005.

[7] F. Bremond, N. Maillot, M. Thonnat, and V. Vu, "*Ontologies for video events*," Technical report no 5189, INRIA, 2004.

[8] C. Fernández and J. Gonzàlez, "*Ontology for semantic integration in a cognitive surveillance system*," Proc. of IEEE SAMT., pp. 206–263, 2007.

[9] L. Snidaro, M. Belluz, and G. Foresti, "*Representing and recognizing complex events in surveillance applications*," Proc. of IEEE AVSS, pp. 493-498, 2007.

[10] S. Dasiopoulou, V. Mezaris, I. Kompatsiaris, P. V.-K., and M. Strintzis, "*Knowledge-assisted semantic video object detection*," IEEE TCSVT, 15(10):1210–1224, 2005.

[11] L. Bai, S. Lao, G. Jones, and A. Smeaton, "*Video semantic content analysis based on ontology*," Proc. of IMVIP., pp. 117–124, 2007.

[12] C. Town, "*Ontology-driven bayesian networks for dynamic scene understanding*" Proc. of IEEE CVPR, pp. 116–123, 2004.

[13] R. Vezzani and R. Cucchiara, "*Visor: Video surveillance on-line repository for annotation retrieval*," Proc. of IEEE ICME, pp. 1281–1284, 2008.

[14] L. Xin and T. Tan, "*Ontology-based hierarchical conceptual model for semantic representation of events in dynamic scenes*," Proc. of IEEE PETS, pp. 57–64, 2005.

[15] U. Akdemir, P. Turaga, and R. Chellappa, "*An ontology based approach for activity recognition from video*," Proc. of ACM-MM, pp. 709–712, 2008.

[16] A. Hakeem and M. Shah, "*Ontology and taxonomy collaborated framework for meeting classification*," Proc. of the IEEE ICPR, pp. 219–222, 2004.

[17] B. Vrusias, D. Makris, J. Renno, "*A framework for ontology enriched semantic annotation of cctv video*" Proc. of IEEE WIAMIS, pp. 5–8, 2007.

[18] K. Kaneiwa, M. Iwazume, and K. Fukuda, "*An upper ontology for event classifications and relations*," Proc. of AI, pp. 394–403, 2007.