A Query Language Combining Object Features and Semantic Events for Surveillance Video Retrieval

Thi-Lan Le[1,2], Monique Thonnat[1], Alain Boucher[3], and François Brémond[1]

[1] ORION, INRIA, 2004 route des Lucioles, B.P. 93, 06902 Sophia Antipolis, France
{Lan.Le_Thi, Monique.Thonnat, Francois.Bremond}@sophia.inria.fr
[2] International Research Center MICA Hanoi University of Technology, Viet Nam
[3] Equipe MSI, Institut de la Francophonie pour l'Informatique, Hanoi, Viet Nam
Alain.Boucher@auf.org

Abstract. In this paper, we propose a novel query language for video indexing and retrieval that (1) enables to make queries both at the image level and at the semantic level (2) enables the users to define their own scenarios based on semantic events and (3) retrieves videos with both exact matching and similarity matching. For a query language, four main issues must be addressed: data modeling, query formulation, query parsing and query matching. In this paper we focus and give contributions on data modeling, query formulation and query matching. We are currently using color histograms and SIFT features at the image level and 10 types of events at the semantic level. We have tested the proposed query language for the retrieval of surveillance videos of a metro station. In our experiments the database contains more than 200 indexed persons and 48 semantic events. The results using different types of queries are promising.

1 Introduction

Video surveillance is producing huge video databases. While there are many works dedicated to object detection, object tracking and event recognition [7], few works have been done for accessing these databases. For surveillance video indexing and retrieval, apart from object tracking, object classification and event recognition, we have 4 main issues: **data modeling**, **query formulation**, **query parsing** and **query matching**. The **data modeling** determines which features are extracted and how they are organized and stored in the database. The **query formulation** specifies the way in which the user expresses his/her query while the **query parsing** [5] specifies the way in which the system analyzes (parses) this query into an internal representation. The aim of **query matching** is to compare elements stored in the database with the query.

The first works dedicated for the surveillance video indexing and video concentrate on data modeling. In [7], IBM smart surveillance engine successfully detects moving objects, tracks multiple objects, classifies objects and events.

However, for retrieving in the database, the queries are done based on only recognized events and metadata. In [6], a data model is built for online video surveillance. Various query types are also presented. In [3], Saykol et al. presented a framework and a visual surveillance querying language (VSQL) for surveillance video retrieval. These previous works have three main drawbacks. Firstly, they work with an assumption that in the indexing phase objects are perfectly tracked and events are perfectly recognized. It is the reason why video retrieving is based on the exact matching of semantic events and metadata. However, object detection and event recognition are not always successful. The video retrieving must work well under imperfect indexing. In this case, the similarity matching on object features is necessary. Secondly, these approaches limit the search space. Because the videos are only indexed by a set of recognized events, the users' queries are restricted to a limited set of predefined events. Thirdly, it is not flexible and does not take into account various users' interests and users' degrees of knowledge. The users could need more or less information according to their interest and could define differently an 'event' in the form of a query in function of their knowledge in this domain.

Our main contributions are designing a video data model and proposing a novel query language. Our data model is different from the model proposed in [6] because it contains object visual features. Our query language overcomes the previous works [7], [6], [3] because it (1) enables users to make queries both at the image level and at the semantic level (2) allows the users to define their own scenarios based on semantic events and (3) retrieves videos with both exact matching and similarity matching.

The rest of this paper is organized as follows: Section 2 describes the proposed approach including data model, query language and query matching. In section 3, we describe some experimental results and their performance evaluation. We conclude this paper in section 4.

2 Proposed Approach

There are two phases in our approach: an indexing phase and a retrieval phase (see Fig.1). The indexing phase has three main tasks: data indexing, feature extraction and event recognition. A data model describes which features and events are used and how they are organized. It is described in detail in the next section. The feature extraction task computes the features defined in the data model on the video data. The event recognition task recognizes events predefined in the data model. The results of these two tasks are used for data indexing.

The retrieval phase is divided into two main tasks: one consists of query formulation and result browsing, the other is composed of query parsing, query matching, and result ranking. In the query formulation task, in order to make the users feel familiar with the query language, we propose a pseudo SQL (Structured Query Language) language. The vocabulary and the syntax are described in the next section. We choose SQL as the basic language because experienced users are normally familiar with this language in traditional databases while novice

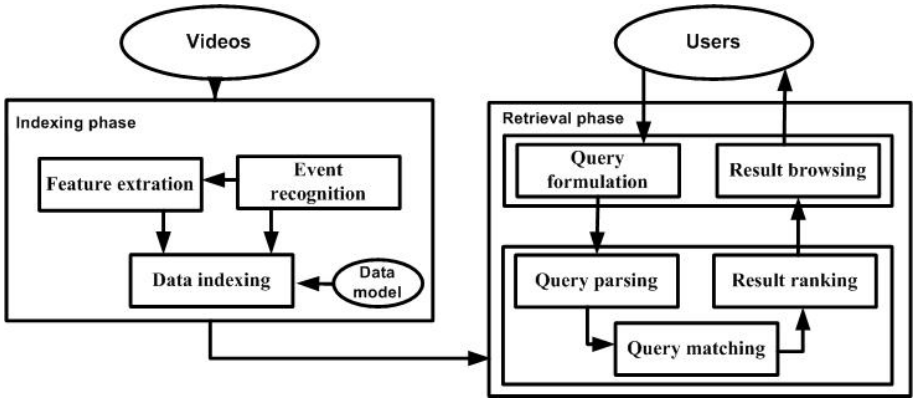


Fig. 1. Global architecture of our approach

users find it easy thanks to an user interface. In the query parsing task, queries built with the proposed language are transmitted to a parser. This parser checks the vocabulary, analyzes the syntax and separates the query into several parts. The query matching task searches in the database the elements that satisfy the query.

2.1 Data model

Our data model contains two main components of interest form the user's point of view: *physical objects* and *events*.

Physical objects: they are the detected objects in the video database. A physical object can be a static object (e.g. contextual object) or a moving object (e.g. a person, a vehicle). Let P be a physical object, P is defined as follows: *P = (Id, [Type], [Name], 2D_positions, 3D_positions, MBRs (Minimum Bounding Box), Features, Time_interval)* where *Id* is the label of the object, the *Type* and *Name* attributes are optional. The *2D_positions*, *3D_positions*, *MBRs* are the sets of 2D positions, 3D positions, MBRs of this object during its lifetime indicated by *Time_interval*. The *Features* currently available in the system are the color histogram and a set of detected keypoints by using SIFT (Scale Invariant Feature Transform) descriptors. We use the SIFT descriptors because the SIFT descriptors are invariant to image scale and rotation and they are shown to provide robust matching across a substantial range of affine distortion change in 3D view point, addition of noise, and change in illumination. Therefore, they helps to match efficiently images. However, SIFT descriptors do not focus on color information like the color histogram. The methods for extracting these features can be found in [4] for the color histogram and in [1] for the SIFT descriptors.

Events: They are the recognized events in the video database. Let E be an event, E is defined as follows: *E = (Id, Name, Confidence_value, Involved_Physical_objects, [Sub_events], Time_interval)*. Where *Id* is the label of

the event and *Name* is the name of the event. The *Confidence_value* specifies the confidence degree of recognized event. Some previous works presented in Section 2 do not take into account this information. In our work, the *Confidence_value* will be used to compute the final distance between video frames and the query. The *Involved_Physical_objects* specifies which physical objects are involved in this event, while the *Sub_events* are optional, and the *Time_interval* indicates the frames in which the event is recognized.

The advantage of our data model is that it is independent of any application and of any feature extraction, learning and event recognition algorithms. Therefore, we can combine results of different algorithms for feature extraction, learning and event recognition (both descriptive and stochastic approaches) and use it for different application domains.

2.2 Proposed language and query syntax

The syntax of a query expressed by our language is the following:

$$SELECT < Output > \ FROM < \ Database > \ WHERE < Condition >$$

Where: **SELECT**, **FROM**, **WHERE** are keywords for a query and they are mandatory. A graphic interface can be developed to generate this syntax but it is out of the scope of this paper.

- **Output** specifies the format of the retrieved results. It can have two values: one is *video_frames* indicating that the retrieved results are video frames in which the <Condition> is satisfied and the other is *number_of_events* indicating that the retrieved result is the number of events in the database satisfying the <Condition>.
- **Database** specifies which parts of the video database are used to check the <Condition>. It can be either * for the whole video database or a list of named subparts. This is interesting for surveillance because the video database can be divided into several parts according to time or location. It allows to accelerate the retrieval phase in the case that the users know already which parts of the video database they are interested in.
- **Condition** specifies the conditions that the retrieved results must satisfy. The users express their requirements by defining this component. The condition may have more than one expression connected together by logic operators (AND, OR, NOT), each expression is started by "(" and ended by ")". This component is the most important component in the language.

There are two types of expression in the condition component: a declaration expression (α_d) which is mandatory and a constraint expression (α_r) which provides additional conditions. The declaration expression indicates the types of variable while the constraint expression specifies constraints the variable must satisfy.

The syntax for a declaration expression is: $(v : type)$ where v is a variable. It is the place where the user specifies if the retrieval is at the image level, the

semantic level or both levels. The authorized types are : *Image*, *Physical_object* and its subtypes (*Person*, *Group*, *Luggage*) and *Event*.

The syntax for a constraint expression is very rich. The constraint expression can be expressed by using a set of projections, functions, predicates, algebra operators and constants. Currently, the authorized projections are {*'s Id*, *'s Type*, *'s Name*, *'s 2D_positions*, *'s 3D_positions*, *'s MBRs*, *'s Features*, *'s Time_interval*} for physical object and {*'s Id*, *'s Name*, *'s Confidence_value*, *'s Involved_Physical_objects*, *'s Sub_events*, *'s Time_interval*} for event; there are two authorized functions which are *histogram_distance* that returns the distance between color histograms and *number_matched_keypoint* that returns the number of matched keypoints between an example image and an indexed object; there are four authorized predicates which are *color_similarity* and *keypoints_matching* that return true if an image example and an indexed object are similar in term of color histogram or keypoints, *involved_in* which verifies whether one indexed object belongs to an event and *before* which verifies whether an event occurs before another event; the authorized algebra operators are =, <, >, >=, =<, ! =}; the constants can be either numbers or strings.

This language is rich enough to express numerous possible queries. Based on the technique proposed in [5], we implement a parser to check automatically the syntax of each query. This parser automatically analyzes the syntax of the query. The results of this parsing allow to locate which databases will be used to match query, which variables must be set and which results must be returned.

An example expressed by this language at the semantic level is: Find Close_to_Gates events occurring in videos of all databases.

*SELECT e FROM * WHERE ((e: Events) AND (e's Name = "Close_to_Gates"))*

where e is a variable of Events, $e's\ Name$ gets Name of e.

Another example expressed by this language at the image level is: Find indexed persons in the database named Video_Database that are similar to a given image.

SELECT p FROM Video_Database WHERE ((p: Person) AND (i: SubImage) AND (i color_similarity p))

where p is a variable of Person, i is a variable that will be set by an image example, *color_similarity* is predicate that decide whether two images are similar (in color).

2.3 Query matching

For each expression α in the condition field, the evaluation of the expression α is performed by matching the indexed database D and the expression α. The results of this process are a set of physical objects and events extracted from D that satisfy α. Let $\eta_i = \{\varsigma_i, \mu_i, I_i\}$ be the ith result instance of the expression α and η be the set of the result instances of α where ς_i are the physical objects or the events, μ_i is the similarity degree that determines how much ς_i satisfy α in a time interval I_i. Currently, we have defined the similarity degree for the predicates based on the feature similarity (e.g. nearest neighbors for SIFT descriptors and

histogram intersection for color histograms). With the other types the default value of the similarity degree is set to 1.

If the query has more than one expression in the condition, the similarity degrees are computed according to the operator linking these expressions as follows: $\mu = \mu_j.\mu_k$, $\mu = max(\mu_j, \mu_k)$, $\mu = (1-\mu_k)$ for AND, OR, NOT operators where μ_j, μ_k are the similarity degrees of the expressions α_j, α_k respectively.

3 Implementation and Experimental Results

3.1 Video event database

In order to validate our approach, we have used two videos of 10 minutes and 2 hours acquired by two fixed cameras at different positions that record human activities in a metro station. An example of two scenes in two videos is shown in Fig.2. The scene contains a platform and several gates. The automatic object tracking, object classification and event recognition proposed in [5] have been applied to these videos. As results, we have 221 indexed persons (101 for the first video and 120 for the second one) with their labels, 3D positions, 2D positions, MBRs and time intervals. One person is perfectly tracked and recognized if in all frames in which this person appears, this person is detected and has one sole label. In addition, 10 event types have been defined and recognized for each frame (inside Platform, close to Gates i (i from 1 to 9)) for the second video. These events are defined in the language proposed in [5] as follows: *Event(**close_to**,*
PhysicalObjects((p : Person), (eq : Equipment))
Constraints((p distance eq $\leq$ Close_Distance))

where eq is Gate i (i from 1 to 9), Close_Distance is a threshold.

*Event(**inside_zone**,*
PhysicalObjects((p : Person), (z : Zone))
Constraints((p in z))

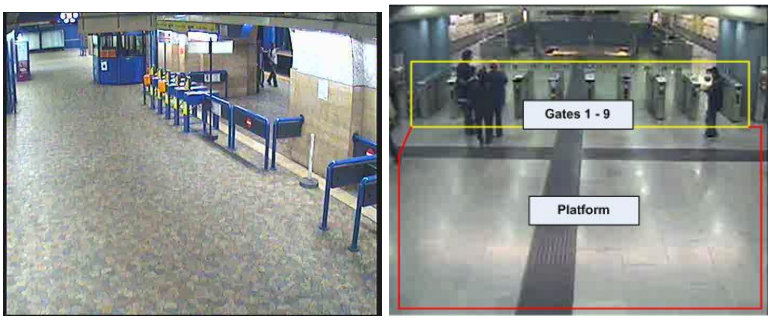
where z is a Platform, *in* is a predicate that checks whether p's center belongs to the polygon z.

The events defined above are recognized for each frame, so we merge the recognized instances which have the same name and involve the same detected person into a merged event. So, we have 29 merged events of inside_zone_Platform (from 24296 recognized instances), 19 merged events of close_to_Gate1 (from 919 instances recognized) from the second video.

3.2 Experimental results and performance evaluation

In order to evaluate the retrieval performance, we use the average normalized rank proposed in [2]. We do not use the measures of true positives, true negatives, false positives and false negatives like in classification problem because for the retrieval problem, the retrieval algorithm is good if it return relevant results first.

$$\widetilde{Rank} = \frac{1}{NN_{rel}}\left(\sum_{i=1}^{N_{rel}}(R_i) - \frac{N_{rel}(N_{rel}+1)}{2}\right) \quad (1)$$



(a) (b)

Fig. 2. Two scenes describe human activities in some metro stations(a) in the first video. (b) in the second video.

where N_{rel} is the number of relevant result for a particular query, N is the size of the tested set, and R_i is the rank of the ith relevant result. $\widetilde{Rank}$ is zero if all N_{rel} are returned first. The $\widetilde{Rank}$ measure lies in the range 0 (good retrieval) to 1 (bad retrieval), with 0.5 corresponding to a random retrieval.

Experiment 1: The goal of this experiment is to check whether our proposed language enables users to retrieve effectively persons in the database even though they are not successfully detected and tracked. For this experiment, the query is: Find in the second video the video frames having persons that are similar (in term of keypoint matching) to the person in this example image. This query is expressed as follows:

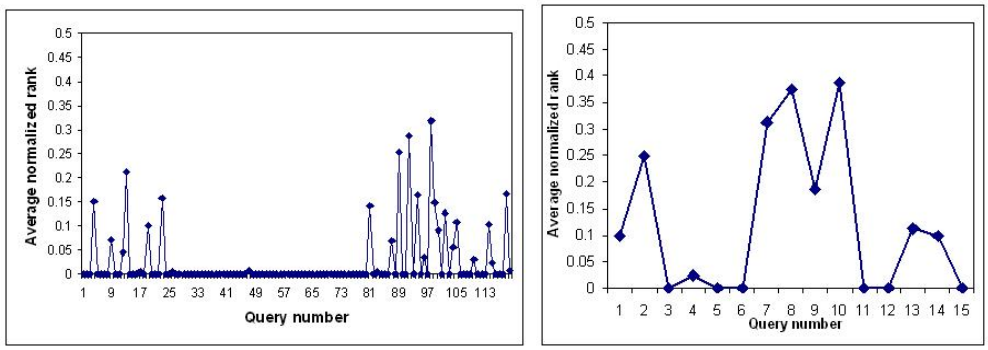
SELECT video_frames FROM Video2 WHERE ((i: Image) AND (o: Person) AND (i keypoints_matching o))

where keypoints_matching is a predefined predicate of our language, i is an example image.

Figure 3.a shows the average normalized rank for this experiment. There are 120 indexed persons in the second video. The retrieval performance is measured over all 120 images using each in turn as a query. Each image query has from 3 to 5 relevant images. The small obtained overage normalized ranks (the maximum value is 0.3) show the proposed approach retrieve successfully the indexed objects even when they are imperfectly indexed.

Experiment 2: This experiment aims at pointing out the advantage of our language. It allows to retrieve interesting events with a detailed description. For instance, the event *close_to_Gate1*(p) indicates that person p is close to the Gate 1. In the case of many *close_to_Gate1* recognized instances, the user may be interested in only *close_to_Gate1* frames containing a person that is similar to a given example. The user can ask a query as follows:

SELECT video_frames FROM Video2 WHERE ((i: Image) AND (o: Person) AND (e: Event) AND (e's Name = close_to_Gate1) AND (o involved_in e) AND (i keypoints_matching o))



(a) (b)

Fig. 3. (a) The average normalized rank for experiment 1 over 120 queries. (b) The average normalized rank for experiment 2 over 15 queries. The value 0 of average normalized ranks corresponds to good retrieval, value 1 corresponds to bad retrieval and 0.5 corresponds to random retrieval.

where i is an image example, *'s Name* is a predefined projection of event's name, *involved_in* is a predefined predicate that determines whether one person is involved in an event and *keypoints_matching* is a predefined predicate described in section 2.

To answer this query, the persons involved in all *close_to_Gate1* events are used for *keypoints_matching* with the given example. The returned results for each query is a list of *close_to_Gate1* events ranked by the number of matched keypoints between the involved persons and the given example. In 19 *close_to_Gate1* events of the second video, there are several events concerning one person. We have chosen 15 example images. Each image turns as input for the query. Figure 3.b gives the obtained average normalized rank over 15 image examples and 19 events of *close_to_Gate1* in the second video. The ground truth is made by hand for these 15 queries. A returned result is considered relevant if it is a *close_to_Gate1* event whose the involved persons show the same person as in the given image example. **Experiment 3**: The objective of this experiment is to measure the capacity of this language to define new events from the recognized ones. From two recognized events in the database: inside_zone_Platform and close_to_Gate1, the user may write a query such as: Find the frames in which one person is going from the Platform to Gate1.

This query is expressed as follows:

SELECT video_frames FROM Video2 WHERE ((o_1: Person) AND (e_1: Event) AND (e_2: Event) AND (o_1 involved_in e_1) AND (o_1 involved_in e_2) AND (e_1's Name = inside_zone_Platform) AND (e_2's Name = close_to_Gate1) AND (e_1 before e_2))

This query is automatically analyzed by a parser presented in the section 2.

Because of imperfect indexing, one person in the real world may be indexed as different persons within the database. Thanks to similarity matching returned

results must consider all *inside_zone_Platform* and *close_to_Gate1* events containing indexed persons that are similar (an exact matching that matches the indexed persons by their labels would have returned for this query incomplete results and sometimes empty ones as shown in Figure 4.a). With our technique to answer this query, the system first matches the involved persons in both *inside_zone_Platform* and *close_to_Gate1* by keypoint matching. For each person involved in *close_to_Gate1* event, it computes the number of matched keypoints between this person and the persons involved in the *inside_zone_Platform* event. A set of persons ordered by their matched keypoints are returned. The *inside_zone_Platform* events containing these persons become candidate for retrieval results. Then, these events are used to check whether they satisfy the *before* constraint with the *close_to_Gate1 events*. The *before* constraint performs based on the starting frames and the ending frames of *inside_zone_Platform* and *close_to_Gate1* events.

For each *close_to_Gate1* event, the retrieval result is a list of *inside_zone_Platform* events that satisfy the *before* constraint with *close_to_Gate1* event and that are ranked by the number of matched keypoint between their involved persons and the person involved in *close_to_Gate1* event. The returned result is considered relevant if it contains an *inside_zone_Platform* event that satisfies the *before* constraint and if their involved persons show the same person in the real world.

The average normalized rank for this experiment is given in Fig.4.b over 19 events of close_to_Gate1.

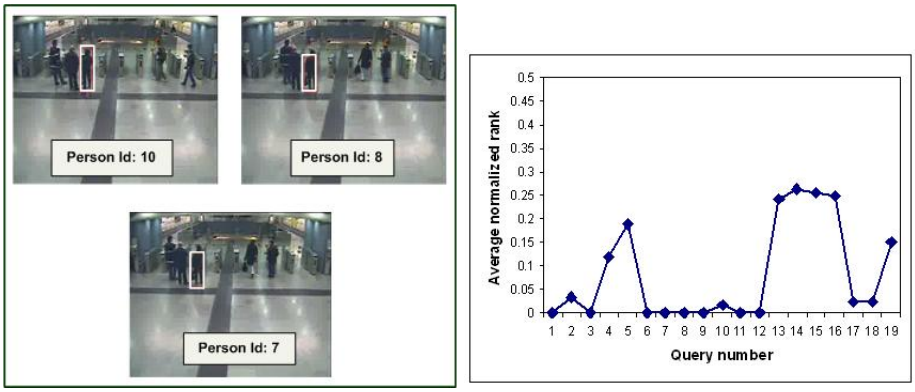
This experiment shows the capacity of this language to define new event from the recognized ones with satisfying results (average normalized ranks of all 19 events are smaller than 0.3).

As shown in Fig.3 and Fig.4, most of the average normalized ranks of the three experiments are small but there are some cases where these measures are quite high (maximum value is respectively 0.3818, 0.3875, 0.262 for the experiment 1, the experiment 2 and the experiment 3) because of the error in object tracking. Therefore, in addition of keypoints and color histogram we intend to use more features or to use a combination of features to solve these problems.

The experiments 1, 2 and 3 show that the proposed approach overcomes the first and the second drawback presented in the introduction. We give a query to explain how the proposed approach can do to overcome the third drawback. In the first video, we do not have the results of event recognition. Users may define then Close to Gates event by stating query:

SELECT video_frames FROM Video1 WHERE ((o: Person) AND (z: Gates) AND (o distance z < threshold))

One person is close to gates if the distance between this person and gates is smaller a given threshold. The distance is computed based on 3D position of the persons and gates. This query takes into account users interest by using a threshold. User can set threshold a value as he/she wants. By setting two different values for the threshold, 100 and 150, we have two different results. The first result returns 10 indexed persons with 320 recognized instance of the Close to



(a) (b)

Fig. 4. (a)Three indexed persons with labels 10, 8, 7 describe the same person in the real world. These indexed persons belong to close_to_Gate1 and inside_zone_Platform events that satisfy *before* constraint. The exact matching based on persons' label gives only one result of person label 10 while our similarity matching gives three persons label 10, 8, 7 with respectively 1, 2, 5 of rank (b) The average normalized rank for experiment 3 over 19 close_to_Gate1 events. The value 0 of average normalized ranks corresponds to good retrieval, value 1 corresponds to bad retrieval and 0.5 corresponds to random retrieval.

Gates event. The second one returns 20 indexed persons with 727 recognized instances.

4 Conclusions

In this paper, we have proposed an approach for video indexing and retrieval for surveillance based on a query language. This new language enables both image and semantic queries and similarity matching. The obtained results for three experiments show that: combining the image level and the semantic level (in experiment 2 and 3) and similarity matching (in experiment 1, 2 and 3) manage imperfect object tracking and imperfect event recognition. New events defined by the user from the recognized ones have been successfully retrieved (in the experiment 3).

Currently, similarity matching has been limited to the color histogram and the keypoints. We plan to study and use more features to enrich the proposed language. In addition, the users may want to make a complex query containing several subqueries. How to combine the results from these subqueries is an issue that we plan to study in the future.

References

1. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, Vol. 60 No. 2, 2004, 91-110.

2. Müller, H., Marchand-Maillet, S., Pun, T.: The truth about corel - evaluation in image retrieval. In Proc. Of. CIVR, London, July 2002, 28-49.
3. Saykol, E., Güdükbay, U., Ulusoy, O.: A database model for querying visual surveillance by integrating semantic and low-level features. In Proc. of 11th International Workshop on Multimedia Information Systems (MIS05)), Vol. 3665, Sorrento, Italy, September 19-21 2005, 163-176.
4. Swain, M. J., Ballard, D.H.: Color indexing. International Journal of Computer Vision, Vol. 7, No. 1, 1991, 11-32.
5. Vu, V.T., Brémond, F., Thonnat, M.: Automatic video interpretation: A novel algorithm for temporal scenario recognition. In Proc. of International Joint Conference on Artificial Intelligence (IJCAI03), Acapulco, Mexico, August 9-15 2003, 1295-1302.
6. Durak, N., Yazici, A., George, R.: Online Surveillance Video Archive System. In Proc. of International Multimedia Modeling Conference, Singapore, January 2007, 376–385.
7. Hampapur, A., Brown, L., Connell, J., Ekin, A., Haas, N., Lu, M. Merki, H., Pankanti, S., Senior, A., Shu, C., Tian, Y. L.: Smart Video Surveillance: Exploring the concept of multiscale spatiotemporal tracking. In IEEE Signal Processing Magazine, Vol. 22, Issue 2, March 2005, 38–51.