Video Surveillance Framework for Crime Prevention and Event Indexing

Levente Kovács[1], Zoltán Szlávik[1], Csaba Benedek[1], László Havasi[1], István Petrás[1], Dávid Losteiner[1], Ákos Utasi[2], Attila Licsár[2], László Czúni[2], and Tamás Szirányi[1]

[1] Distributed Events Analysis Research Group,
MTA SZTAKI, Hungary
levente.kovacs@sztaki.hu
http://www.sztaki.hu
[1] Dept.of Image Processing and Neurocomputing,
University of Pannonia, Hungary
utasi@uni-pannon.hu
http://www.uni-pannon.hu

Abstract. The paper will present a video surveillance and event detection and annotation framework for semi-supervised surveillance use. The system is intended to be used in automatic mode on camera feeds that are not actively watched by surveillance personnel, raising alarms and enrolling annotation data when unusual events occur. We present the current detector filters, and the easily extendable modular interface. Current filters include local and global unusual motion detectors, left object detector, motion detector, tampering/failure detector, etc. The system stores the events and associated data, which can be organized, searched, annotated and (re)viewed. It has been tested in real life situation for police street surveillance, and we are working towards developing a deployable version.

1 Introduction

While the literature of content based image search contains quite a large number of proposed and in some cases practical systems, the open literature of real time surveillance feed analysis is much more limited. Also, processing surveillance feeds for event detection is mostly done as a post-processing feature, working with archived footage [1,2], not aiding the operators' work in real time. [3] is a method based on tracking moving regions from an aerial view. A system with similar goals is that of [4] with many architectural and procedural differences, but most importantly our system provides a wider range of detectors, multiple event alarms, and an easy modular interface. The presented system is not based on later content based processing and indexing of archived footage, but runs real time filters on live footage and signals unusual events, so as to reduce the need to actively and constantly watch the live feeds. This is important, since in many existing systems a few operators need to survey hundreds of cameras. At the same time, it has the classical archive

functionalities, which makes it also suitable for eventual later content based data mining. The filters we present are all real time, and are based on pixel level approaches complemented with robust statistical evaluation and learning steps. The framework has been developed as part of a project especially started to produce such a system for real life application by the local police stations in the city's districts.

2 System Architecture

In simple terms, the system consists of a user interface and a database backend. The interface runs on a workstation with dual display, which also contains the multihead frame grabber cards that accept the camera feeds. The feeds come from large camera matrix multiplexer stations. The users monitor and use the system through this interface, while the database backend can be anywhere, given the proper internet connections. Figure 1 shows a simplified diagram of the general system architecture, while Figure 2 shows a snapshot of the main interface, handling three feeds.

The main application can handle a maximum of four live camera feeds simultaneously, and each feed can be assigned a chain of selected detection filters (which we call filter chains). The feed handlers, the filter chains, and the filters themselves were all written to be real time, and with SMP and heavy multithreading in mind; all filters and chains run separately and concurrently. Increasing the number of available processors can greatly increase the possibility of using more filters on more feeds. Currently all filters run real time, but a combination of multiple filters on

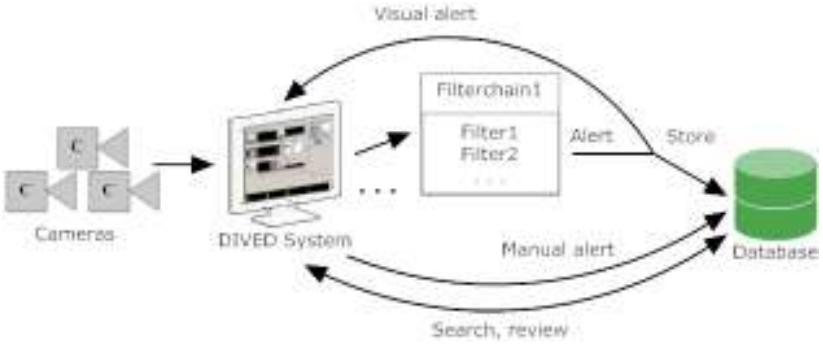

Fig. 1. Main parts of the system architecture.

multiple feeds can quickly run into processing and memory bottlenecks. Thus a careful selection of hardware and filter combinations is necessary.

Modules/filters can be added easily, either by coding them by using a provided class template as internal filters, or by a provided a library template, as a plugin. Either way, the coder needs only focus on developing the core algorithm, interfacing is seamless.

Fig. 2. Main interface window.

3 Functions, Modules

In this section we describe some of the more important modules/filters currently deployed in the framework. The system also contains classical surveillance functions like image and video archiving, large display of the feeds on a secondary monitor and so on, which we will not detail here. The functions described here are all automatic and consist of a panorama image creator from panning camera feeds, maskable motion detector, camera jump detector for cameras that iterate among different stationary positions, unusual global and local motion detector, fight detector, left object detector, camera fail/tampering detector, annotation, search and review of events. Any order of filter combinations can be assigned to each camera feed separately, and they will run concurrently and independently of each other. All filters run automatically, in real time, and need no manual intervention.

3.1 Panorama/Mosaic Image

The need of constructing and displaying a panorama image of the scene arose since there have been a lot of panning cameras that cover a large field of view. This module allows us to construct and display the full field of view of a camera for the operator, and also to identify the actual camera position. The method continuously registers the incoming frames and builds a mosaic. The properties of these cameras are totally unknown and different on each camera source. There are some articles (e.g. [5]) dealing with moving cameras but they are not based on statistical approach to segment background and foreground. Our approach computes the transformation matrices by using the extracted optical flow vectors. The stable points (good features to track) are determined by the Harris corner [6] detector. The corresponding points between frames are verified with the motion vector of the flow field. Instead of using

RANSAC [7] to compute the homography we implemented a simpler hit-and-miss iterative algorithm: every iteration drops the worst points from the dataset, and the remaining are the base points for the computation of transformation between frames. Figure 3 shows examples of built panoramas.

Fig. 3. Sample panorama images. Red rectangle shows actual camera position.

3.2 Motion Detector

Motion detection and optical flow field extraction steps are required for many of the system's filters, as a base for higher processing stages. But the extracted flows can also be used for other purposes, e.g. the simple task of raising alarms when any type of motion occurs on a surveyed area. In this case, the interface's panorama image pane gives the possibility to mark a certain area of interest with the mouse, and alarms will be raised when motions occur over the masked area. This function can be used to automatically signal e.g. the departure or arrival of a car, open/close of a door/gate, or any activity over a security area (e.g. Figure 4).

Fig. 4. Motion detector on masked area of interest (blue: mask, red: motion).

3.3 Unusual Global Motion Detector

For this filter, the goal was to detect unusual large motion patterns. Intended use cases are, e.g.:

- Someone goes against the traffic in a one way street: long term statistics show one major motion direction, and then a different motion occurs.
- One lane jams in a two way street: long term statistics show two typical motion directions, and then one of them disappears.

- Accident, traffic jam: statistics show intensive various motions, which considerably slows, stops, or drops in variance.

To achieve this goal, in a learning phase long term global statistics are built from direction distributions and typical motion types based on extracted motion fields of the image sequence. Then, in the detection phase, we try to fit the actual motion data to one of the statistics, and raise alarms when no good fit can be found.

The motion field extracted from the image sequence is cut every t_1 second into t_2 long segments, and we take the mean of such segments as a *sample*. If $t_1 < t_2$ then we will get overlapping segments, which will help in smoothing the blockiness at segment borders. We keep collecting the samples for a t_3 time period, which will produce N samples. From the samples directional distributions are constructed, and directional histograms are built from the motion fields. The histograms are quantized into ε degree bins between 0 and 360. These directional histograms will represent the typical motion forms of the scene.

In the *learning phase*, the N samples are classified into k classes by K-means clustering. Distance between the samples is calculated by L_2 norm. K-means needs a k to be given a priori, which we overcome by starting with a large class number, then performing a class consolidation step. If the means A_0, B_0 of classes A and B are closer than n_0, then B is merged into A with a new A_0^* mean. In the end we will have k^* classes, where $k^* < k$.

During the *detection phase*, a decision step follows, where we try to fit the actual sample to the ξj ($j=1..k^*$) distributions with at most $3\sigma j$ (j^{th} deviation) discrepancy. If it fits none of the distributions, an unusual motion form alarm is raised. Figure 5 shows samples from the operation of this filter.

Fig. 5. Sample frame (top left); flow field and direction histogram (top right: red is the mean motion field, green is the directional histogram of the current sample); bottom: directional classes, with colored circles as the class mean.

3.4 Unusual Local Motion Mask

This filter (based on [8]) also detects unusual motion patterns, but in a local manner: it operates locally, builds different statistics for parts of the frame, can signal unusual patterns at different locations, and can give the mask of the moving object.

We collect eight-bin motion direction histograms for all image pixels. Larger number of bins could enhance the adaptation, but would also increase the uncertainty. We assume that the relative occurrence of motion vectors gives an effective estimate of the empirical probability:

$$P_{Dir} = \|O_{Dir}\| \Big/ \sum_{Dir} \|O_{Dir}\|$$

where $\|O_{Dir}\|$ is the number of observations in one of the predefined direction classes $Dir \in \{i \cdot 45°, i = \overline{0,7}\}$. The probability that an observed vector belongs to an unusually moving object is $P^{(U)}{}_{Dir} = 1 - P_{Dir}$. The obtained local statistics are smoothed by spatial averaging, with a Mean Shift [9] segmentation step of the probabilities. Figure 6 illustrates the estimated and the smoothed motion statistics (using discriminating colors). In the detection phase, we use the segmented probability map for the estimation of anomalous motion:

$$P_{Dir} = S_i\big|_{P_{Dir}} \text{ where}$$

$$S_i \in S = \{S_1, S_2, ...S_N\} \text{ and } S_i = [x_0, y_0, ...x_n, y_n, P_{Dir}]$$

where each segment S_i is a connected component of the image labeled with a probability distribution P_{Dir} obtained by the classification step.

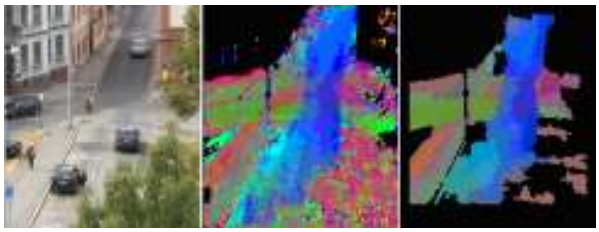

Fig. 6. Sample frame, estimated motion statistics (middle) and typical motion fields after classification (right).

The method is based on statistical processing of raw data without object level understanding, and uses spatio-temporal information for the analysis of motion. We assume that the unusual event happens on at least two consecutive frames, suggesting a Markov Chain property: if we find an anomalously moving pixel and we estimated its motion direction at time *t*, then, projecting back onto the previous frame there should also be - with high probability - a corresponding anomalous pixel. This is formalized as:

$$P^{(U,M)}{}_{x,y,t} = P^{(U)}{}_{Dir,x,y,t} \cdot \max_{x',y' \in R} \{P^{(U)}{}_{Dir,x',y',t-1}\}$$

where R is the *5x5* pixel neighborhood of the motion compensated position (x' and y').

3.5 Fight Detector

The algorithm of this filter detects fighting people (a few people or small group), and raises an alarm when such disorderly motion patterns are detected in the video stream. The tuning of this algorithm is easy and mostly invariant to the characteristics of video (spatial resolution, refresh rate, view parameters, etc.). This filter runs in real time, on live camera feeds. Trajectories are constructed from multiscale Lucas-Kanade-tracked [10] Harris corner points [6] through frames. The main steps are as follows:

1. Stationary points are removed.
2. If the following hold
   - $\min_{Tlength} < N < \max_{Tlength}$
   - $\min_{curved} < curved_k < \max_{curved}$
   - $\min_{C\,\mathrm{var}} < norm\_curved_k < \max_{C\,\mathrm{var}}$ then increase the *alarm* counter
3. If $alarm > \min_{alarm}$ then raise an alarm.

where: $\min_{Tlength}$ and $\max_{Tlength}$ are minimal and maximal trajectory lengths, N is the actual trajectory length, $\min_{C\,\mathrm{var}}$ and $\max_{C\,\mathrm{var}}$ are the min. and max. curvature variances, $curved_k$ is the actual trajectory curvature, and

- $norm\_curved_k = curved_k / length_k$ is the normalized curvature measure,
- $curved_k = \sqrt{\sum\left(\left\|m_k - t_k\right\|^2\right)/N}$ is the deviation from the trajectory's mean,
- $length_k = \left\|t_k(0) - t_k(N)\right\|$ is the distance of the first and last trajectory points, and
- *alarm* is the number of trajectories where the above shape constraints are fulfilled simultaneously.

Figure 7 shows two excerpts from real feeds where people were fighting, red overlay showing the frames where the filter raised an alarm signal.

3.6 Left and Removed Object Detector

In conventional video surveillance applications, the aims of background modeling and background subtraction modules are usually limited to moving object detection and analysis. However, relevant information can be exploited by following the changes in the background as well. We implemented a filter, which not only detects objects moving in front of the camera, but it detects changes in the static background and

signals the appearance of new objects (i.e. objects that are brought into the field of view, then left there, see Figure 8, or objects that are taken from the field). The method can be used to observe abandoned or stolen objects, which is an important surveillance task.

a *b*

Fig. 7. Fight detector feeds. Red overlay shows frames where fight alarms were raised.

The proposed method (building on [15]) extends the widely used Gaussian mixture background modeling approach of [11]. Each pixel s is considered as a separate process, which generates an observed pixel value sequence over time (t is the time index):

$$\left\{x^{[1]}(s), x^{[2]}(s), \ldots x^{[t]}(s)\right\}$$

To model the recent history of the pixels, [11] suggested a mixture of K Gaussians distribution:

$$P\left(x^{[t]}(s)\right)=\sum_{k=1}^{K} w_k^{[t]}(s)\cdot\eta\left(x^{[t]}(s),\mu_k^{[t]}(s),\sigma_k^{[t]}(s)\right)$$

where $k=1,\ldots,K$ are *unique* and in time *static* id's of the mixture components, while $\eta(.)$ is a Gaussian density function, with given μ mean and σ deviation. We ignore multi modal background processes, and consider the background Gaussian term to be equivalent to the Gaussian component in the mixture with the largest weight.

The mixture parameters are iteratively refreshed. The weight as updated as follows:

$$w_k^{[t+1]}(s)=(1-\alpha)\cdot w_k^{[t]}(s)+\alpha\cdot M^{[t]}\left(k,x^{[t]}(s)\right)$$

where the following matching operator is used:

$$M^{[t]}\left(k,x^{[t]}(s)\right)=\begin{cases}1 & if\ \left|\dfrac{x^{[t]}(s)-\mu_k^{[t]}(s)}{\sigma_k^{[t]}(s)}\right|<c\\ 0 & otherwise\end{cases}$$

The mean and deviation parameters of the mixture components are similarly updated. Denote component with the maximal weight by:

$$k_{\max}^{[t]}(s)=\underset{k=1,\ldots,K}{\arg\max}\, w_k^{[t]}(s)$$

Hereon, the background model can classify each pixel into three classes: *foreground*, *background* and *changed background* as follows:

- If $M^{[t]}\left(k_{\max}^{[t]}(s),x^{[t]}(s)\right)=0$: pixel s is *foreground* at time t.
- Else, if $k_{\max}^{[t]}(s)\neq k_{\max}^{[t-1]}(s)$: pixel s is *changed background*
- Else: pixel s is *background* at time t.

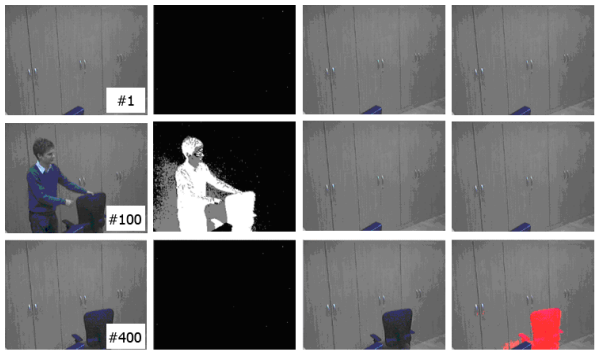

Fig. 8. Left object sample, columns from left to right: input frame, extracted motion and object/shadow segmentation, learned background, frames with new object alerts (red shows the newly left object)

3.7 Camera Fail/Tampering Detector

This filter detects when a periodic camera movement (i.e. automatic panning camera) is interrupted by an operator. This is an important case, since the system should run on unsupervised feeds and stop when an operator is active. To detect the panning period length, a method like [12] is used; a similarity plot is constructed, and then projected on a 45 degree line. The actual period length can be calculated as the distance between consecutive peaks (Figure 9).

Detection of the intervention is an online process and must be recognized immediately. Building on the Lucas-Kanade [10] algorithm, the method continuously registers the frames and calculates the optical flow between them.

Fig. 9. The projected similarity plot (left) showing the projected peaks (right).

After statistical analysis two hysteresis thresholds are calculated. If the value calculated from the current optical flow is outside the thresholds an intervention is signaled. This indicates that the operator e.g. stopped the panning, zoomed to a region of interest, or stopped the movement.

Sudden significant changes in illumination are also detected by another filter, which is useful to signal when e.g. the image gets too dark or too bright (relatively), lights are switched on or off in indoor views, the camera gets covered.

3.8 Annotation

The application's interface provides the possibility of annotating frames from different feeds. This is possible by selecting a certain feed and the annotation tool, and then putting in some text (Figure 10). This annotation will be stored in the database, as a viewable, searchable entry (Figure 11). This way it is possible to add comments to certain events/scenes to provide an easier way to review and search stored details.

3.9 Archiving

Without going into details, it should be mentioned that the framework also has the traditional capabilities of archiving frames and videos besides events. Frame and video archiving can be done automatically in the background, for all or selected feeds, with custom frame rates. In the case of events at least a frame will be stored along the event in the database, which will become handy when running queries among the stored alarm data, showing not just textual and filter data, but also a browseable associated frame series.

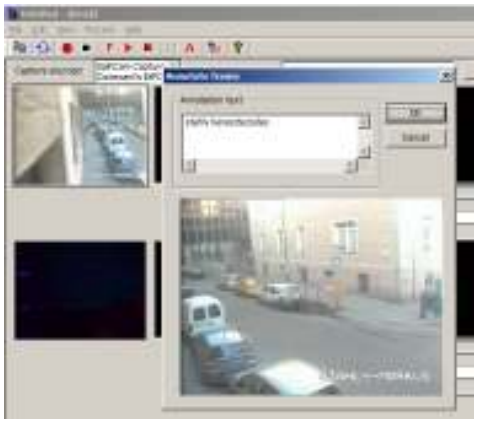

Fig. 10. The annotation dialog.

Forrás	Időbélyeg	Típus	Leírás	Annotáció
Feed 0	2008-05-29-15:41:34:421	2	mozgás jelzés	
Feed 0	2008-05-07-15:31:36:468	3	közvetlen riasztás	közvetlen riasztás
Feed 1	2008-05-07-15:31:26:218	1	annotáció	mozgás
Feed 0	2008-05-07-15:31:17:359	2	mozgás jelzés	
Feed 1	2008-05-07-15:31:16:796	0	mozgáspálya2 r...	
Feed 1	2008-05-07-15:31:15:359	3	közvetlen riasztás	közvetlen riasztás
Feed 1	2008-05-07-15:31:14:78	0	mozgáspálya2 r...	
Feed 1	2008-05-07-15:31:13:343	3	közvetlen riasztás	közvetlen riasztás
Feed 1	2008-05-07-15:31:10:546	0	mozgáspálya2 r...	

Fig. 11. A sample of searched events, annotation being a separate field among the data.

3.10 Alerts, Visualization, Retrieval

When a filter in any of the filter chains signals an alert, a message with the alert details and filter data at the time of the alert is sent to the main framework, which creates a database entry with the details of the alert and the filter data, also storing a captured frame. These data will be searchable, by location, time interval, type of alerts, annotation texts, and sample images. Also, the alarm also shows a visual alert on the main user interface, in multiple ways, to be easily noticeable. One way of visual alerting is shown on Figure 12: each feed has an associated event histogram graph, which shows the last events, color coded for different filters. Clicking on a

graph point brings up a browseable view dialog, showing the alert details (Figure 13). Also, searching, viewing, and browsing through the stored events is possible in the search and query dialog (Figure 14).

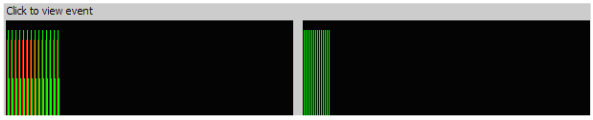

Fig. 12. Sample alarm graphs, which show a visual representation of the alerts. Clicking on a point in the graph pop up a dialog with alert information (Fig. 13).

Fig. 13. Dialog with alert information and the belonging stored image. Clicking left/right makes browsing around the event possible.

Fig. 14. Query/search dialog for reviewing stored event/alarm data and viewing associated details.

Besides textual, time interval, alert type – and so on – searches, a basic content based search option is also available. It provides the possibility of selecting an area of interest on an alert frame, and search for similar frames in the database (Figure 17).

The search engine uses local maxima to calculate the distance between scale invariant features (SIFT) [13]. The main advantage of this approach is that there is no need to search for the two best matchings to compute the ration and the distance. The idea came from the analysis of matched SIFT descriptors. All descriptors never fit each other completely (Figure 15); they are just very similar to each other (if there is a match at all). E.g. rotation causes changes in the feature vector because of the discrete

transformation but it will not change the local maxima (*locmax*). In our attempt, when using the local maxima only 3 neighboring values are checked so the 128 long descriptor may contain not more than 128/3=42 *locmax* positions. In practice this number spans from 15 to 32.

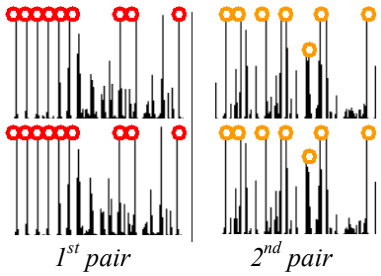

Fig. 15. Standard way for pairing descriptors (for two sample image pairs). Local maximas take the same positions.

Because of the possible difference between the *locmax* vector lengths, we used the DTW (Dynamic Time Warping) algorithm [14] as a distance measure. Using only the positions is not enough for the correct distance as the structure of the descriptor is determined by both the position and the weight. Before the DTW, we calculate the distance between positions vectors, into the distance matrix *D*:

$$D(i,j) = \left| p1(i) - p2(j) \right|^p$$

where *p1* and *p2* are position vectors and *p* is the power factor. Then, the matrix is corrected using the normalized weights:

$$D(i,j) = D(i,j) \cdot (1 + \left| w1(i) - w2(j) \right|)$$

where *w1* and *w2* come from SIFT. The DTW works the classical way, only with a different input: instead of vectors, the compensated *D* matrix is used. The resulting distance is used to compare two *locmax* descriptors:

$$Dist = DTW(D)/k$$

Dimension reduction. The main problem with searching by SIFT descriptors is the high dimensionality of the feature space, which is also why fast tree structures (e.g. KD-Tree) cannot be used. Our proposed approach uses only the dominant values of *locmax* descriptors. In this case, the significant part of the distance between two features is estimated from the most important *locmax* values instead of the Euclidean distance of the whole feature vector. We have found empirically, that the optimal dimensionality is 7. The importance of these reduced descriptors is that there is no further need for supervised learning in a dimension reduction technique (e.g. covariance matrix). The proposed descriptor can be computed by a *locmax* search and a sort algorithm on *locmax* values. A sample matching using such values is shown in Figure 16.

4 Conclusions, Applications

We presented an automatic surveillance system, which is intended to be an aid for surveillance operators who handle hundreds of feeds, thus being physically unable to watch all feeds at once. We presented some of our current filters that are the base of the unusual event signaling and review framework. The actual version has been tested at local police surveillance stations, and we are working towards creating a finalized deployable version. The system is being developed to be as much modular and extendable as possible, with easy integration with existing systems.

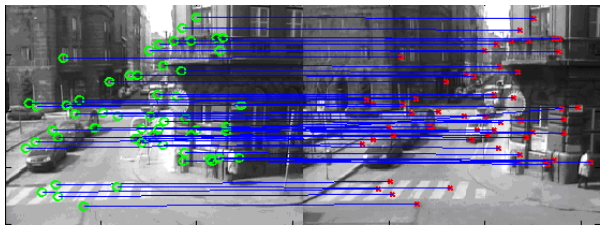

Fig. 16. Sample showing correct matching of locmax SIFT features.

Fig. 17. Browsing by selecting region of interest: select a region, search, browse through results, then view associated alert details by double clicking.

Acknowledgments. This work has been supported by MUSCLE (FP6-507752) and JUMAS (FP7-214306) projects.

References

1. A. Hampapur, L. Brown, R. Feris, A. Senior, S. Chiao-Fe, Y. Tian, Y. Zhai and Max Lu, “Searching surveillance video”, In Proc. of AVSS 2007, pp. 75-80.
2. J. Meessen, M. Coulanges, X. Desurmont and J.F. Delaigle, "Content-Based Retrieval of Video Surveillance Scenes," Multimedia Content Representation, Classification and Security, 2006.
3. G. Medioni, I. Cohen, F. Bremond, S. Hongeng, R. Nevatia, “Event detection and analysis from video streams”, IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 23, no. 8, pp. 873 – 889, 2001.
4. A. Adam, E. Rivlin, I. Shimshoni and D. Reinitz, “Robust Real-Time Unusual Event Detection using Multiple Fixed-Location Monitors”, IEEE Trans. on Pattern Analysis and Machine Intelligence , vol. 30, no. 3, pp. 555-560, 2008.
5. R. Cucchiara, A. Prati and R. Vezzani, "Advanced Video Surveillance with Pan Tilt Zoom Cameras" In Proc. of Workshop on Visual Surveillance (VS) at ECCV 2006.
6. C. Harris and M.J. Stephens, “A combined corner and edge detector”. Alvey Vision Conference, pp. 147–152, 1988.
7. M.A. Fischler and R.C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". Comm. of the ACM 24, pp. 381–395, 1981.
8. Á. Utasi and L. Czúni, "Anomaly Detection with Low-level Processes in Videos", The 3rd International Conference on Computer Vision Theory and Applications, pp. 678-681, VISAPP 2008.
9. D. Comaniciu and P. Meer, "Mean Shift: A Robust Approach Toward Feature Space Analysis", IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (5), pp. 603-619, 2002.
10. B.D. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision”. In Proc. of Imaging understanding workshop, pp 121-130, 1981.
11. C. Stauffer and W. E. L. Grimson, “Adaptive background mixture models for real time tracking,” In Proc. IEEE Conf. Computer Vision and Pattern Recognition, vol. 1, pp. 22–29, 1999.
12. R. Cutler and L. Davis, "Robust Real-Time Periodic Motion Detection, Analysis, and Applications," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 22 No. 8, pp.781-796, 2000.
13. D.G. Lowe, "Object recognition from local scale-invariant features," In Proc. of ICCV, 1999, pp. 1150-1157.
14. C. S. Myers and L. R. Rabiner, “A comparative study of several dynamic time-warping algorithms for connected word recognition”, The Bell System Technical Journal, 60(7):1389-1409, 1981.
15. Cs. Benedek and T. Szirányi:, “Bayesian Foreground and Shadow Detection in Uncertain Frame Rate Surveillance Videos”, IEEE Trans. on Image Processing, vol. 17, no. 4. pp. 608-621, 2008.