We are IntechOpen,
the world's leading publisher of
Open Access books
Built by scientists, for scientists

6,900
Open access books available

186,000
International authors and editors

200M
Downloads

154
Countries delivered to

Our authors are among the
TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS
BOOK CITATION INDEX
INDEXED

WEB OF SCIENCE™
Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

9

Multi-Stage Video Analysis Framework

Andrzej Czyżewski, Grzegorz Szwoch, Piotr Dalka, Piotr Szczuko,
Andrzej Ciarkowski, Damian Ellwart, Tomasz Merta,
Kuba Łopatka, Łukasz Kulasek and Jędrzej Wolski
*Gdansk University of Technology, Multimedia Systems Department,
Poland*

1. Introduction

Video monitoring systems are a necessity in the modern times. Although some people object the idea of 'being watched', surveillance systems actually improve the level of public security, allowing the system operators to detect threats and the security forces to react in time. Surveillance systems evolved in the recent years from simple CCTV systems into complex structures, containing numerous cameras and advanced monitoring centers, equipped with sophisticated hardware and software. However, the future of surveillance systems belongs to automatic tools that assist the system operator and notice him on the detected security threats. This is important, because in complex systems consisting of tens or hundreds of cameras, the operator is not able to notice all the events.

In the last few years many publications regarding automatic video content analysis have been presented. However, these systems are usually focused on a single type of human or vehicle activity. No complex approach to the problem of automatic video surveillance system has been proposed so far. In order to address this problem, the authors designed a framework that analyses camera images on multiple levels, from basic detection of moving objects to advanced object recognition and automatic detection of important events. The proposed system has a flexible structure, with functional modules that may be selected so that the system suits the need of a particular application. These modules are based on algorithms proposed by various authors, adapted to the needs of the presented framework and enhanced by the authors in order to provide an efficient solution for automatic detection of important security threats in video monitoring systems.

The chapter is organized as follows. Section 2 presents the general structure of the proposed framework and a method of data exchange between system elements. Section 3 is describing the low-level analysis modules for detection and tracking of moving objects. In Section 4 we present the object classification module. Sections 5 and 6 describe specialized modules for detection and recognition of faces and license plates, respectively. In section 7 we discuss how video analysis results provided by other modules may be used for automatic detection of events related to possible security threats. The chapter ends with conclusions and discussion of future framework development.

2. Framework structure

The system for intelligent video analysis has a distributed architecture (Fig. 1), consisting of multiple node stations, one central station and operator stations. Node stations are placed in the monitored area, close to video acquisition sensors (i.e. cameras). They are responsible for camera management and for automatic analysis of images from all cameras in their vicinity. Each node station contains a small-factor PC running Linux operating system and equipped with video analysis software. The computer is enclosed in a weather-proof casing which makes possible to mount a node station outdoors. Results of video analysis are sent from node stations to the central station for storing, evaluating and notifying operators. Central station is also responsible for aggregation results coming from multiple node station in order to detect large-scale, global threats. Such configuration makes possible to use wide-band, short-distance cable connections between cameras and node stations to transfer high-quality video streams and wireless communication medium to send results of analysis to the central station. The system operator has access to the analysis results and camera images from the whole system through the operator station which consists of a monitor set, controllers and computers with the specialized software. Depending on the network throughput available, there is also a possibility to view live video streams from any camera in the system from an operator station.

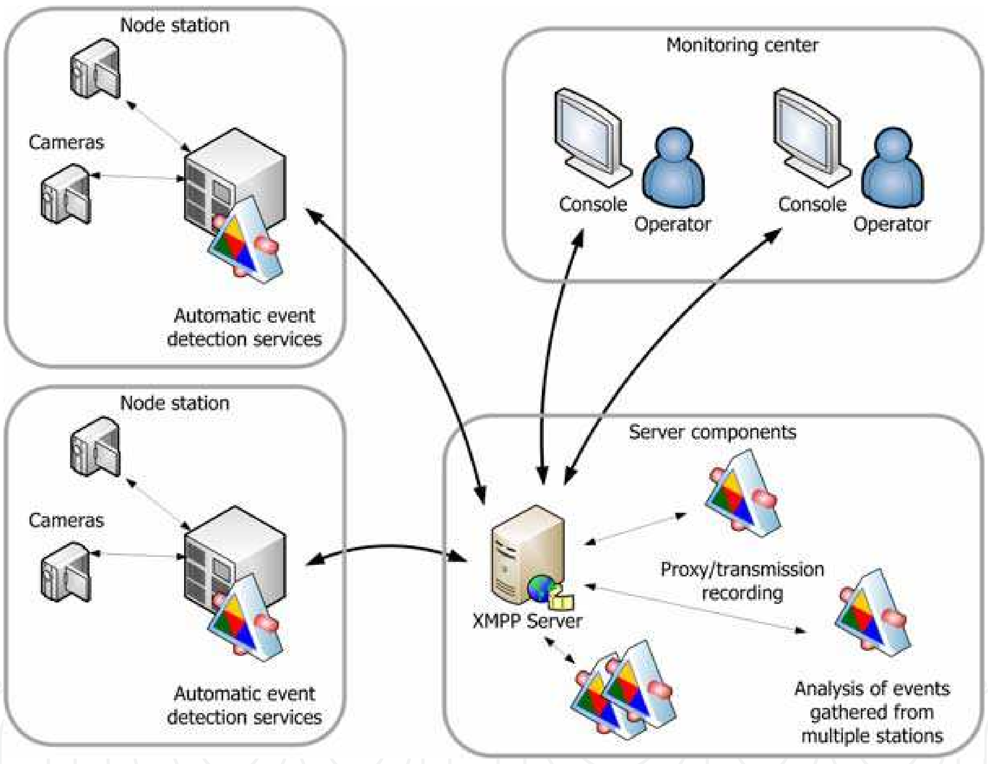

Node station
Cameras
Automatic event detection services
Node station
Cameras
Automatic event detection services
Monitoring center
Console
Operator
Console
Operator
Server components
XMPP Server
Proxy/transmission recording
Analysis of events gathered from multiple stations

Fig. 1. Video analysis system architecture

An important aspect of the proposed system is the issue of data transmission. A layered approach was devised, based on TCP/IP protocols suite, which enables the sensitive data to be transferred by means of open Internet, regardless of the presence of Network Address Translators or firewalls which typically interfere with establishment of multimedia communication sessions. At the foundation of the developed solution lays the Extensible Messaging and Presence Protocol (XMPP), which is designed for building Instant Messaging systems, but thanks to its virtually-unlimited extensibility, it is an increasingly popular tool for general-purpose application servers and distributed applications. Its use provides an added value of security and message integrity layer (based on TLS standard), authorization, addressing scheme and a container for information structuring within self-describing, XMLbased messages. Furthermore, XMPP grants access to a plethora of the protocol extensions developed by its mature community, which may be found surprisingly useful within the context of surveillance solution. Such particular extension, which is very important for the proposed system, is so-called Jingle protocol, which is a tool for establishing multimedia communication sessions. This forms the core of the system's audio and video streaming functionality. In fact Jingle is a session-control (signaling) protocol while the actual multimedia data transfer is performed out-of-band of XMPP connection due to performance reasons. For this purpose encrypted Realtime Transfer Protocol (RTP) sessions are utilized. As a consequence, the initiation of multimedia streaming may be problematic in the presence of NAT devices or firewall on the route between transmission endpoints. Therefore, an additional proxy service has been implemented within the system, which allows for the efficient multimedia transmission between any connected terminals regardless of their network conditions.

Two types of digital cameras are employed in the video monitoring system. Stationary (fixed), wide-angle cameras, especially megapixel ones, offer a wide field of view and are used for video content analysis and event detection. The other type – pan-tilt-zoom (PTZ) cameras – allow for adjusting their field of view as required and they are used for automatic tracking of objects, selected either manually by an operator or automatically by the event detection system. PTZ cameras provide an operator with a detailed view of the situation.

Video analysis performed in the node station is a multi-stage process (Fig. 2). It consists of low-level image processing modules and high-level event detection modules. First, all moving object present in a fixed camera field of view are detected in each video frame, independently. Then, all moving objects are tracked in the adjacent video frames, as long as they stay in the camera field of view, in order to obtain characteristics of their movements. Various static (e.g. shape, texture) and dynamic (e.g. location, speed, heading) object features are used to classify them into a few groups (e.g. humans, cars). Object features are used in the final, high-level analysis stage for automatic detection of important events. Event detection is supplemented by additional, specific modules, such as face detection and recognition, license plate recognition and others. The main functional modules of the framework will be presented in detail further in this chapter.

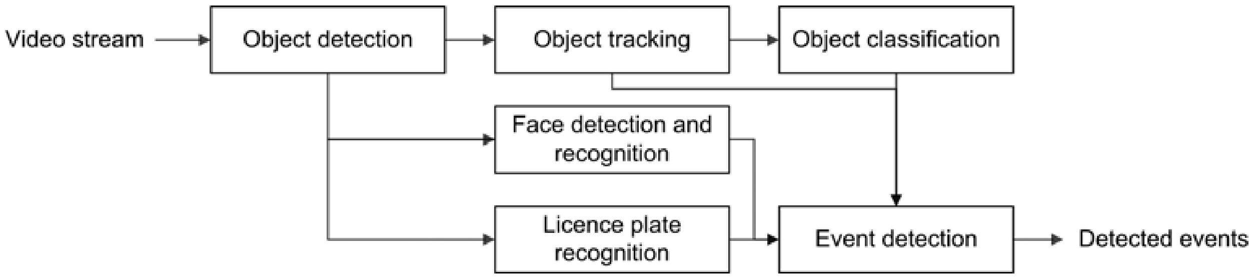


Fig. 2. Framework for video processing in the node station

3. Detection and tracking of moving objects

3.1 Object detection

Detection of moving objects is usually the first stage of video processing chain and its results are used by further processing modules. Most video segmentation algorithms usually employ spatial and/or temporal information in order to generate binary masks of objects (Li

& Ngan, 2007; Liu & Zheng, 2005; Konrad, 2007). However, simple time-averaging of video frames is insufficient for a surveillance system because of limited adapting capabilities. The solution implemented in the framework utilizes spatial segmentation for detection of moving objects in video sequences, using background subtraction algorithm (Yang et al., 2004). This approach is based on modeling pixels as mixtures of Gaussians and using an on-line approximation to update the model (Elgammal et al., 2000; Stauffer & Grimson, 2000). This method proved to be useful in many applications, as it is able to cope with illumination changes and to adapt to the background model accordingly to the changes in the scene, e.g. when motionless foreground objects eventually become a part of the background. Furthermore, the background model can be multi-modal, allowing regular changes in the pixel color. This makes it possible to model such events as trees swinging in the wind or traffic light sequences.

Background modeling is used to model current background of the scene and to differentiate foreground pixels of moving objects from the background (Dalka, 2006; Czyzewski & Dalka, 2007). Each pixel in the image is modeled with a mixture of *K* Gaussian distributions for this purpose. The probability that a pixel has the value x_t at the time *t* is given as:

$$p(x_t) = \sum_{i=1}^{K} w_t^{\,i} \eta\left(x_t, \mu_t^{\,i}, \Sigma_t^{\,i}\right) \tag{1}$$

where w_t^i denotes the weight and μ_t^i and Σ_t^i are the mean vector and the covariance matrix of *i*-th distribution at the time t, and η is the normal probability density function:

$$\eta(x_t, \mu, \Sigma) = \frac{1}{(2\pi)^{0.5 \cdot D} \sqrt{|\Sigma|}} e^{-0.5 \cdot (x_t - \mu)^T \Sigma^{-1} (x_t - \mu)} \tag{2}$$

where *D* is the number of elements describing pixel color; for the RGB color space D is equal to 3. It is assumed that each Gaussian distribution represents a different background color of a pixel. The longer a particular color is present in the video stream, the higher value of the weight parameter *w* of the corresponding distribution.

With every new video frame, the parameters *w*, μ and Σ of distributions for each pixel are updated according to the on-line K-means approximation algorithm. In the first step, distributions are ordered based on the value of the *r* coefficient given as:

$$r = \frac{w}{\sqrt{|\Sigma|}} \tag{3}$$

where $|\Sigma|$ is the determinant of the covariance matrix Σ. A particular color of the scene background is usually more often present in the observation data than any color of foreground objects and as such is characterized by the low variance. Thus a distribution with a higher *r* value represents the background color more accurately.

Every new pixel value x_t is checked against existing distributions, starting from the distribution with the highest value of the *r* coefficient, until the first match is found. The pixel matches the distribution if its color lies within 2.5 standard deviations of the distribution. If there is no match, a distribution with the lowest *r* value is replaced with the new one with the current pixel as its mean values, an initially low weight and high variances. The weight of the first matching distribution is increased, while the weights of other distributions are decreased based on the value of the learning rate parameter α (Dalka, 2006). The higher the α is the faster model adjusts to changes in the scene background (e.g. caused by gradual illumination changes), although moving objects, which remain still for a longer time (e.g. vehicles waiting at traffic lights), would quicker become a part of the background.

If there is a matching distribution, its mean and variance values are tuned according to the current value of the pixel; the speed of converging is determined by the learning rate α [7]. Only the first *D* distributions of pixel *x* in time *t* ordered by the decreasing *r* coefficient value are used as the background model where *D* is defined as:

$$D_x^t = \arg\min_d \left(\sum_{i=1}^{d} w_i^{\,t} > T \right) \tag{4}$$

If *T* is small, then the background model is usually unimodal. If *T* is higher, the background color distribution may be multimodal, which could result in more than one color being included in the background model. This make possible to model periodic changes in the background, properly. If the current pixel value does not match any of the first *D* distributions, it is considered as a part of a foreground object.

Object segmentation is supplemented with shadow detection and removal module. The shadow of a moving object moves together with the object and as such is detected as a part of the foreground object by a background removal algorithm. The shadow detection method is based on the idea that while the chromatic component of a shadowed background part is generally unchanged, its brightness is significantly lower (Horprasert et al., 1999; Dalka, 2006). Every new pixel recognized as a part of a foreground object during the background subtraction process is checked whether it belongs to a moving shadow. If the current pixel is darker than the distribution and its color lies within 2.5 standard deviations of the model for at least one of the first D distributions forming the background model, the current pixel is assumed to be a shadow and is considered as a part of the scene background.

In the result of background modeling, a binary mask denoting pixels recognized as belonging to foreground objects in the current frame is obtained. It needs to be refined by the means of morphological processing in order to allow object segmentation (Dougherty & Lotufo, 2003; Dalka, 2006). This process includes finding connected components, removing objects that are too small, morphological closing and filling holes in regions. Additionally, an algorithm for shadow removing from the mask using morphological reconstruction is implemented (Xiu et al., 2005). The morphological reconstruction procedure involves two binary images: a mask and a marker. In the mask image, all pixels belonging to either the moving object or the shadow have value of one, and all the background pixels have zero value. The marker is obtained by applying an aggressive shadow removal procedure to the object detection result, so that all the shadow pixels are removed, some pixels belonging to the moving object may also be removed in this process (the object mask is damaged). The marker is first 'cleaned' by removing isolated pixels, then it is dilated by a structural element which usually has a large size (typically, 9 x 9 structural element is used). The result of marker dilation is then combined with the mask using logical AND operation. As a result, shadows are removed and the moving object masks are properly reconstructed. Example results of moving object detection in a single video frame are presented in Fig. 3.

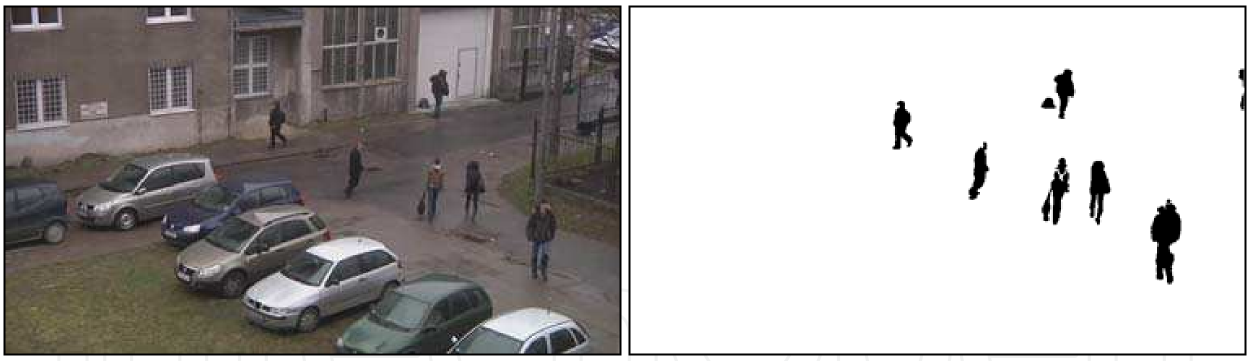


Fig. 3. Example results of moving object detection: original video frame (left) and binary mask denoting moving objects (right)

3.2 Object tracking

After the moving objects are found in each consecutive camera frame, movement of each object on the frame-by frame basis is needed. This is the task of an object tracking module. For each new detected moving object, a structure named a tracker is created. The position of the object in the current camera frame is found by comparing the results of object detection (the blobs extracted from the image) with the predicted position of each tracker. The prediction process estimates the state of each tracker from the analysis of the past tracker states. An approach based on Kalman filtering (Welch & Bishop, 2006) was used for prediction of trackers state in the presented framework.

The state of each tracked object is described by an 8-element vector, containing parameters related to object position in the camera image (*x*, *y*), the size of object's bounding box (width *w*, height *h*) and change of these parameters relative to the previous frame (Czyzewski & Dalka, 2008). Therefore, the state of the tracker in frame *n* is described by the vector X_n:

$$X_n = \left[x_n, y_n, w_n, h_n, dx_n, dy_n, dw_n, dh_n\right]. \tag{5}$$

The vector *X* is initialized in the first two frames in which the object was found. For each successive frame, the predicted state of the tracker is calculated by the Kalman filter. Next, the predicted state of the trackers is compared with the blobs found by object detection. A relation between a tracker and a blob is established if the bounding box of the tracker covers the bounding box of an object by at least one pixel. The tracker is then updated with the measurement - the position of the matching blob.

If there is an unambiguous one-to-one relation between one blob and one tracker, this tracker is updated by the state of the related blob. However, if there is more than one matching blob and/or tracker, a tracking conflict occurs. The authors proposed the following algorithm for conflict resolving. First, groups of matching trackers and blobs are formed. Each group contains all the blobs that match at least one tracker in the group and all the trackers that match at least one blob in the group. Next, all the groups are processed one by one. Within a single group, all the trackers are processed successively. If more than one blob is assigned to a single tracker, this tracker is updated with all blobs assigned to it, merged into a single blob. This is necessary in case of partially covered objects (e.g. a person behind a post) that causes the blob to be split into parts. In other cases. all the matching blobs are merged and the tracker is updated using its estimated position inside this blob group. This approach utilizes the ability of Kalman trackers to predict the state of the tracked object, provided that it does not rapidly change its direction and velocity of movement, so that the predicted state of the Kalman filter may be used for resolving short-term tracking conflicts. The estimated position is used for updating the tracker position, change of position is calculated using the predicted and the previous states. The predicted values of size and change in size are discarded and replaced by values from the previous filter state, in order to prevent disappearing or extensive growth of the tracker, if its size was unstable before entering the conflict situation. Therefore, it is assumed that the size of the object does not change during the conflict. The vector of parameters used for updating the Kalman tracker during the conflict may be written as:

$$X_n = \left[x_{pn}, y_{pn}, w_{n-1}, h_{n-1}, x_{pn} - x_{n-1}, y_{pn} - y_{n-1}, dw_{n-1}, dh_{n-1} \right] \tag{6}$$

where index p denotes the value predicted by the Kalman filter, n is the frame number.

A special case of tracking conflict is related to 'splitting objects', e.g. if a person leaves a luggage and walks away. In this situation, the tracker has to follow the person and a new tracker needs to be created for the luggage. This case is handled as follows (Szwoch et al, 2010). Within each group of matching trackers and blobs, subgroups of blobs separated by a distance larger than the threshold value are found. If there is more than one such subgroup it is necessary to 'split' the tracker: select one subgroup and assign the tracker to it, then create a new tracker for the remaining subgroup. In order to find the subgroup that matches the tracker, the image of the object stored in the tracker is compared with the image of each blob, using three measures: color similarity, texture similarity and coverage. The descriptors of the blob are calculated using the current image frame. The descriptors of the tracker are calculated during tracker creation and updated each time the tracker is assigned to only one blob (no conflict in tracking).

After the conflict resolving is done, the tracking procedure finishes with creating new trackers for unassigned blobs and removing trackers to which no blobs have been assigned for a defined number of frames. The process is repeated for each camera frame, allowing for tracking the movement of each object.

Fig. 4 presents an example of object tracking with conflict resolving, using the procedure described here. A person passes by a group of four persons walking together. During the conflict, positions of both objects are estimated using the Kalman filter prediction results. When these two objects become separated again, assignment of trackers to blobs is verified using the color, texture and coverage measures. As a result, both objects are tracked correctly before, during and after the conflict occurs.

The proposed algorithm for object tracking and conflict resolving provides correct results in case of short-term conflicts involving low to moderate number of objects. Performance of this procedure decreases in case of high number of objects conflicting with each other. It should be also noted that if the object detector passes incorrect data to the object tracker, e.g. masks of the objects are distorted because of improper lighting conditions, tracking errors are inevitable. Several directions of further work on the presented procedure are considered in order to improve the algorithm for long-term conflicts occurring with high frequency. For example, the parameters of Kalman filters related to estimation process may be selected adaptively, instead of being constant. This way, tracker may adapt to different tracking conditions. Moreover, a simplified estimation of tracker position, based solely on tracker estimation, may be enhanced by using an algorithm searching for the object in the image using characteristic features of the object. This way it will be possible to select the part of the blob that matches the tracked object, which will improve the tracking accuracy.

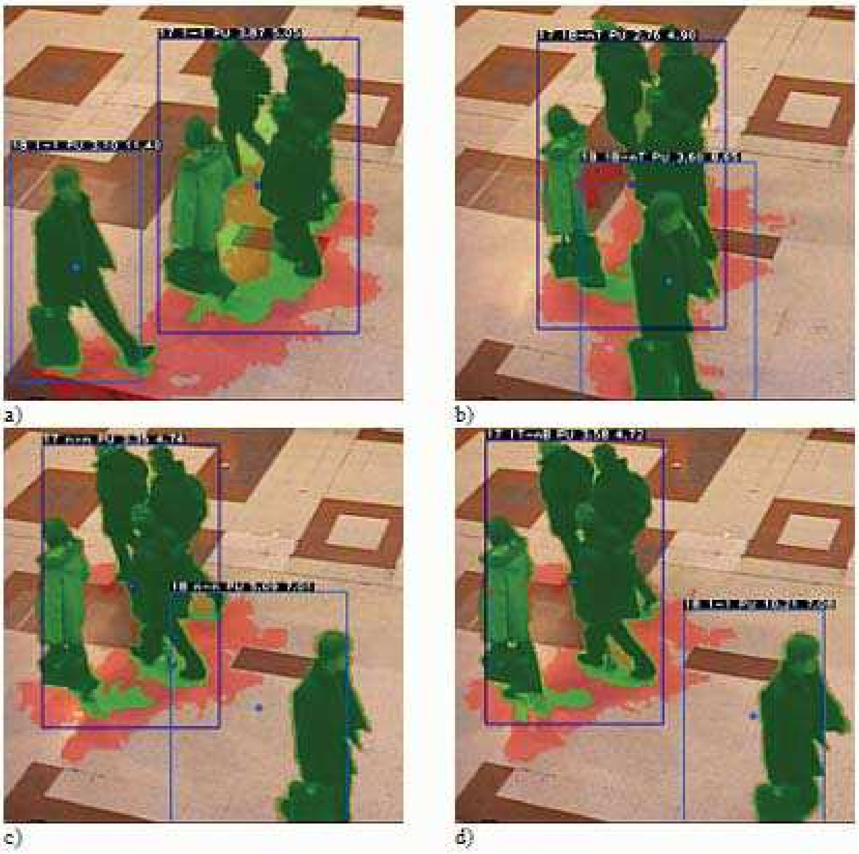


Fig. 4. Example of object tracking and conflict resolving results (images from the PETS 2006 database)

4. Object classification

4.1 Basic classification

Object classification is required for proper detection of various events involving specific types of moving objects (e.g. the presence of a person in the middle of a road is forbidden while vehicles are allowed). The video analysis framework provides a few modules devoted to object classification. The first module is responsible for dividing all objects into three groups: persons, vehicles and unanimated objects (e.g. luggage). Classification is based on object dimensions, therefore calibration of field of camera view is required. The calibration process involves selection of marked points in camera image and measuring their position in both image pixel space and in real world. For the purpose of presented system, a non-coplanar calibration mode should be used, meaning that calibration points with different height value should be used. Coordinates of calibration points are fed into the calibration procedure that calculates values for conversion between 2D image coordinates and 3D world coordinates (conversion from 2D to 3D requires that *z* coordinate, describing height, is provided). The authors used Tsai's calibration model to calculate 14 conversion parameters related to camera lens and camera orientation (Tsai, 1987). Using this conversion, the width and the height of the object is estimated (Szwoch et al., 2009). This estimation depends on the viewing angle, so time averaging has to be used to achieve good accuracy of size estimation. The objects are then assigned to classes using rules that test physical size and velocity of the objects (average and inter-frame change values are used).

If an object is 'removed' from the background (e.g. a luggage that was stationary for a prolonged time and was treated as a part of the background by the object detector, is taken by a person), it leaves a 'hole' in the background model, treated as a new object by the object tracker. The classification module has to decide whether a new tracker represents real object or part of the background. Otherwise, it would not be possible to detect e.g. abandoned luggage and taken (stolen) object, because these two situations would be treated in the same manner by the event detector. The proposed solution is based on observation that the contour of the detected object will contain edges only if it is not a part of background (Szwoch et al., 2010). Therefore, a grayscale image of the detected object (blob) and its mask (having non-zero values for pixels belonging to the blob and zero values otherwise) are processed by the Canny edge detector, resulting in E_B and E_M images. Next, E_M is morphologically dilated by a 7 x 7 structuring element *SE*, the result is combined with E_B and again dilated with the same element:

$$R_B = \left[\left(E_M \oplus SE \right) \cap E_B \right] \oplus SE \tag{7}$$

A measure used for detection is calculated by dividing a number of non-zero pixels in R_B by a number of non-zero pixels in E_M dilated with *SE*. It was found during the experiments that if this measure is above 0.6, the blob represents the actual object, otherwise it is a part of the background.

4.2 Recognition of the object type

The second module is used to divide a general class of objects into specific subclasses (types). This module will be described using a vehicle type classifier (using types such as sedans, vans, trucks, etc.) as an example. Video-based object type classification utilizes results of moving object detection and tracking. Only images of objects classified as vehicles, without any occlusions with other objects, are analyzed in this example. Numerous vehicle image descriptors are used for vehicle type classification (Dalka & Czyzewski, 2010), they may be divided into two groups. The first group includes features based on vehicle mask only, such as: mask aspect ratio, eccentricity of the ellipse fitted to the mask, extent, defined as the proportion of the mask bounding box area to the mask area, solidity, defined as the proportion of the mask convex hull area to the mask area, proportion of the square of the mask perimeter to the mask area, 24 raw, central and normalized moments of the mask up to the third order (without trivial ones) and a set of seven Hu invariant moments of the mask (Flusser & Suk, 2006); the moments are invariant under translation, changes in scale and rotation. The second group of vehicle descriptors is based on image content. Because there is no correlation between vehicle type and its color, only luminance images are used. All image pixels outside of a vehicle mask are ignored. Two sets of vehicle image descriptors are computed; the first one is based on SURF (Speeded Up Robust Features) and the second one is derived from gradient images using Gabor filters.

Speed Up Robust Features (SURF) (Bay et al., 2006) is a scale- and rotation-invariant local image descriptor around a selected interest point. Its main advantages are repeatability, distinctiveness, and robustness as well as short computation time. Interest points (their location, orientation and size) may be chosen manually or automatically (e.g. using Fast-Hessian detector that is based on the determinant of the Hessian matrix. SURF descriptors ale calculated in the square regions centered around each interest point. The region is divided into 4×4 equal subregions. In each subregion, the Haar wavelet responses in horizontal d_x and vertical d_y directions (in relation to the interest point orientation) are calculated. Sums and absolute sums of wavelet responses form a four-element feature vector v for each subregion:

$$v = \left(\sum_{d_x<0} d_x, \quad \sum_{d_x\geq 0} d_x, \quad \sum_{d_x<0} \left|d_x\right|, \quad \sum_{d_x\geq 0} \left|d_x\right|, \quad \sum_{d_y<0} d_y \quad \sum_{d_y\geq 0} d_y \quad \sum_{d_y<0} \left|d_y\right| \quad \sum_{d_y\geq 0} \left|d_y\right| \right) \tag{8}$$

This result in a SURF descriptor vector containing 128 elements for each interest point. The wavelet responses are invariant to illumination offset. Invariance to contrast is achieved by turning the descriptor into a unit vector.

SURF descriptor vectors are obtained for four interest points that are set manually in the centers of four rectangular, non-overlapping areas the vehicle image is divided into; the areas are located symmetrically around a center of gravity of the vehicle mask. The size of each interest point is equal to the height or width of the area, depending on which value is greater. Final vehicle feature vector based on SURF descriptors contains 128 × 4 = 512 elements.

The second set of vehicle image descriptors is based on filtering a gradient image with a bank of Gabor filters. Image gradients are calculated in vertical and horizontal directions independently using Sobel operator with an aperture size equal to 3. The final gradient image is obtained by adding squared vertical and horizontal gradients. Images are scaled to the fixed resolution 100×80 pixels. Gabor filter kernels are similar to the 2D receptive field profiles of the mammalian cortical simple cells. Therefore they reveal desirable characteristics of spatial locality and orientation selectivity (Daugman, 1988). In the spatial domain, a 2D Gabor filter is a Gaussian kernel function modulated by a sinusoidal plane wave (Grigorescu et al., 2002). A bank of eight Gabor filters based on two different wavelengths (2.5 and 4) and four different orientations (0°, 45°, 90° and 135°) is used. A scaled gradient image I is convolved with each Gabor filter g with two variants of phase offset φ, according to the equation:

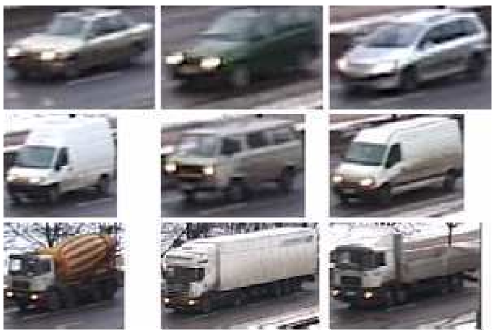
$$I_G = \sqrt{\left(g_{\phi=0} * I\right)^2 + \left(g_{\phi=\pi/2} * I\right)^2} \tag{9}$$

This results in eight filtered vehicle images. For each image, seven Hu invariant moments are derived (Flusser & Suk, 2006). Final vehicle feature vector based on Gabor filters contains 7 × 8 = 56 elements.

A feed-forward Artificial Neural Network (ANN) with one hidden layer is used as a classifier for vehicle descriptors. The number of ANN inputs i_{ANN} corresponds with the number of vehicle features. The number of outputs o_{ANN} is equal to the number of vehicle types recognized. An expected output consists of a maximum value on one output and minimal values on other outputs. Therefore a vehicle type corresponding with the maximum output value is returned as the classification result. ANN is trained with a resilient backpropagation algorithm (RPROP). Sigmoid activation functions are used in all neurons. Vehicle descriptors are divided into training and validations sets randomly (Dalka & Czyzewski, 2010).

For the purpose of experiments, a 30-minute video recording from a traffic camera has been tested. All moving vehicle images have been automatically extracted using object detection and tracking algorithms, validated and hand-labeled with an appropriate vehicle class. Three vehicle subclasses were used: sedans, vans and trucks. Sample vehicle images are presented in Fig 5. The database contained images of 525 different vehicles (367 sedans, 80 vans and 78 trucks). Each vehicle was represented by 40 images on average (Fig. 5). Therefore independent results of classifications of images of the same vehicle can be aggregated in order to increase total effectiveness. The final class assigned to a vehicle is equal to the most frequently labeled class for all images of the vehicle.

Vehicle type classification is a highly complex task because of large variety of vehicles belonging to each class and the fact, that vehicle physical dimensions and poses vary during their movement in a camera field of view. Nevertheless, up to 95% of vehicles can be classified correctly (Fig. 5). Experiments prove that feature vector consisting of vehicle mask statistical parameters and image features based on SURF and Gabor filters is sufficiently universal to characterize vehicles with different pose, size and resolution.



Vehicle type	No. of vehicles	% of correct classifications
sedan	315	97.4%
van	41	94.4%
truck	39	84.6%
all types	395	95.8%

Fig. 5. Sample vehicle images for each vehicle type: first row - sedans, second row - vans, third row - trucks; vehicle images are rescaled individually to the same vertical size (left) and results of vehicle type classification (right)

4.3 Shape analysis

Shape is one of the properties which could be analyzed to acquire more information describing an object. There are several task for which such data could be useful. After building a proper classifier, shape information could be applied for shape-based event detection or for object classification, providing additional information about the state of the object (e.g. person walking, sitting, running, etc.). Recognition of real objects in 2D images is a difficult task, but in comparison to methods which classify objects on the basis of their real dimensions and velocities, this method does not require any camera calibration techniques. Considering shape recognition for the purpose of object classification, a few things need to be denoted. Surveillance systems consist of multiple cameras placed and oriented variously. This causes the shape of an observed object to differ between cameras. Furthermore, objects may rotate around their own axis creating more various silhouette representations (Fig. 6).

The proposed shape-based object classification method assumes that the camera observation angle is known to the algorithm. With this condition fulfilled a set of binary images is prepared. These images correspond to objects representing various classes and types at a specified horizontal angle. In the simplified example presented here, three classes are considered: car, human and unknown.

Before training any classifier, all gathered shape images need to be parameterized (Fig. 7). This step is required to lower the amount of data being processed as well as to unify the shape representation. There are two general groups of methods used for shape parameterization (José, 2004). First group treats the shape as the whole region – i.e. Zernicke's Moments (Hae-Kwang, 2000). The other group operates on the shape contour – i.e. Chain Code, Pairwise

Geometric Histogram (Ashbrook, 1995). Independently from the group, each method can be characterized by its properties making it usefull for a certain purpose. In this paper a custom method is used. In the first step of this parameterization, the processed image is resized to a set dimensions (100x100 pix) preserving proportions. Next, pixels belonging to the shape are added in rows and columns. After this operation Hu moments are calculated and as the result of parameterization a final vector of 207 values is created for each image. Among all machine learning algorithms, Support Vector Machines classification method is chosen for the purpose of data clustering. SVM parameters are set during experiments for optimal performance and generalization (Chih-Wei 2003). Such approach analyzes objects independently in every video frame. In order to improve the algorithm performance, the final class assignment is done by averaging decisions from the last *k* frames equal to 0.5 s.

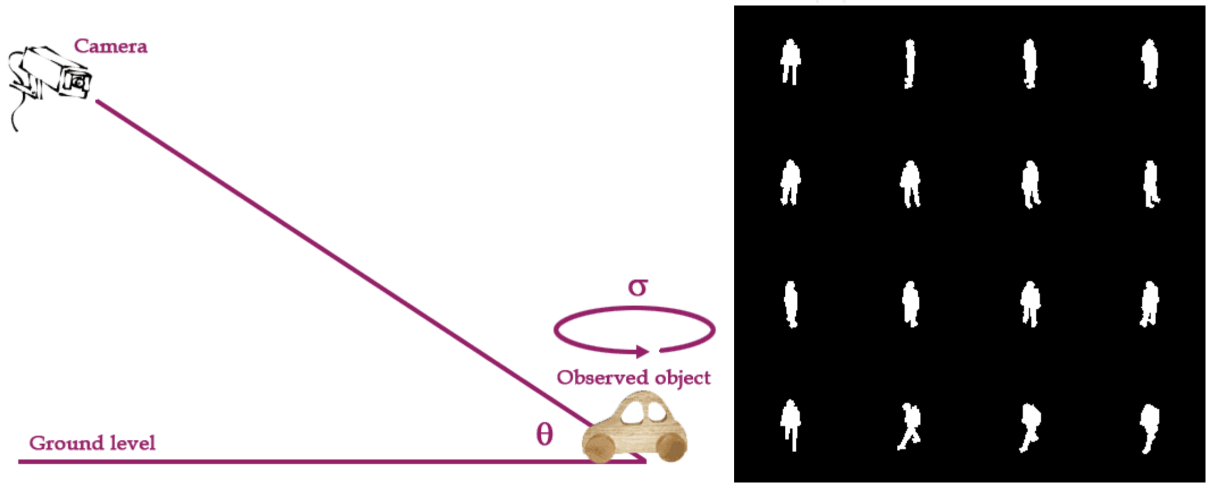


Fig. 6. 3D object recognition in 2D image problem (left) and sample 3D model projections at a set angle (right)

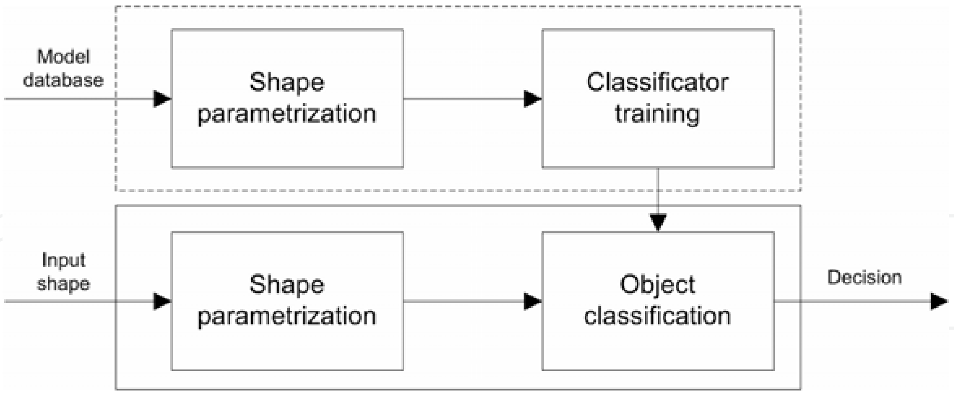


Fig. 7. Block diagram of data flow in the described method. Dotted border denotes the operation performed once per algorithm execution

The algorithm was tested using a set containing approximately 100 objects of each class. Classification results are presented in Fig. 8 and in Table 1, in the form of confusion matrix. In the tests, the proposed method worked as expected, achieving c.a. 90 percent accuracy. Experiments show that models for observation angle equal to 20, 40 and 60 are sufficient to cover the camera orientations from 10 to 70 degrees while sustaining similar performance. The described classifier distinguishes only three general classes but it is possible to recognize sub-classes within these by building a cascade classifier. As shown in this section, shape recognition can be successfully applied for object classification without the necessity of calibrating cameras but still providing the same or even better degree of accuracy.

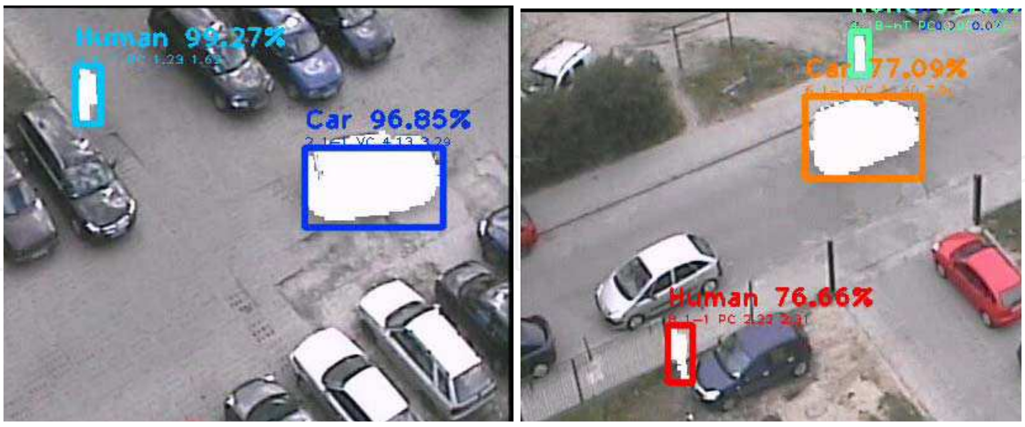


Fig. 8. Examples of classification results obtained using the shape recogniser, with temporal probability denoted

	Car [%]	Human [%]	Unknown [%]
Car [%]	95.51	2.49	2.00
Human [%]	4.22	84.21	11.57
Unknown [%]	6.74	2.55	90.71

Table 1. Results of shape based object classification for camera angle approx. 40o

5. Face detection and recognition

The modules of the proposed framework described so far provide essential data for automatic event detection. However, in practical applications, more sophisticated modules may be needed, such as face detection and recognition algorithm described in this section. This optional module may be used e.g. for detection of presence of a particular person in the camera view or to check whether the detected person is in database of wanted people.

Two algorithms have to be implemented in this module. First, face detection is performed in order to select the image parts that contain faces. For this task we use an approach based on cascade of Haar classifiers (Viola & Jones, 2001). The classifier is trained with a set of face images scaled to the same size. During analysis, the classifier is applied to the image sections containing detected moving objects and classified as persons. If a face is detected, the classifier outputs a region of the image containing the face.

The second part of the module performs face recognition and is much more complex. The algorithm is expected to identify the detected person based on a series of face images. Face recognition techniques are under constant development since late 60s of 20th century, beginning with primitive local attempts trying to employ face geometry analysis. As this prototypic techniques failed due to intra-personal variability being greater than extra-personal one, some new approaches have arisen in late 80s, targeting at synthesis of effective face image representation, including so called holistic (global) approaches such as EigenFaces, based on principal component analysis (PCA)(Turk & Petland, 1991), FisherFaces (Fisher's Linear Discriminant, FLD)(Fisher, 1936) and neural network classifiers.

Moreover, hybrid methods emerged like *3D Morphable Models* (3DMM)(Vetter & Blanz, 1999). Next step in this area was possible thanks to advances in video surveillance and digital image processing systems which has occurred in the late 90s. Using a sequence of video frames instead of static face image allowed for employing probabilistic frameworks, able to exploit additional context contained in frame sequence. Contemporary attempts focus mostly on exploring probabilistic frameworks and excelling hybrid approaches like 3D modeling and active appearance modeling.

The main problem of implementing face recognition algorithms in real systems, such as the presented framework, is the insufficient number of image samples per person in the database. In most cases, the database contains a single photo of a wanted person, e.g. a passport photo. Using this sample image only, it is not possible to recognize a person if the pose is different from the sample. Our efforts focused on exploring possibilities of 3DMM method for creation of additional sample images. For this task, 3D face scanner based on phase shifting interferometry (PSI) method was constructed and programmed to acquire 3D face image data and build 3D image database. The collected 3D scans (29 women and 25 men faces) were used to create three models of face geometry (male, female and generic one, Fig. 9). In each scanned image, a set of 13 facial features was marked manually. In each group, faces were aligned using facial points and normalized. The algorithm for fitting a 2D image to the 3D face model works by finding a minimum of the function *S*:

$$S = \sum_{n=1}^{N} \left(\alpha_n - \beta_n \right)^2 \tag{10}$$

where N is a number of facial features (N = 13), α_n and β_n are coordinates of n-th facial point in 2D image and in 3D model, respectively. After the best match is found, texture of the image is transferred to the model and a new, textured 3D face model is constructed. This model may be then rotated in 3D space and compared with face images, e.g. the ones detected in the camera image.

The image matching algorithm using the created 3D models was tested using 200 sample images from FERET database. In most cases, reconstruction results were correct. However, the algorithm failed to match smaller regions such as eye-corners or the iris. The future work in this area will focus on automatic detection of facial points and implementing a geometry morphing algorithm in order to increase the reconstruction accuracy.

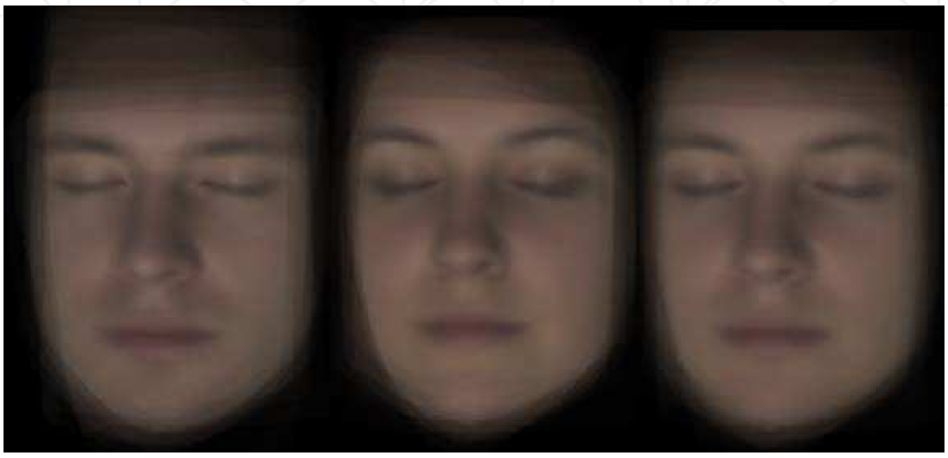


Fig. 9. The averaged 3D face models, from left: male, female and generic

For creation of facial features vector, needed for face matching, both Eigenface and Fisherface methods were implemented and compared using 2 training sets of 100 images each, from FERET database. The second set contained the same identities as the first one, but with some occlusions added and mimic expressions present. These images were rendered on the 3D averaged face model in order to simulate variations in exposition (pose and illumination) and then converted back to 2D in order to obtain virtual samples. This allowed using databases with varying number of samples per identity (from 1 sample to 40 samples). Simulations revealed that for substantial pose variation (around 40 degrees), Fisherfaces outperforms Eigenfaces approach, but its accuracy spans between 90% (for 10÷30 identities) and 10% (for sets of 50÷70 identities) which is locally less than Eigenfaces. With moderate pose variation (around 22 degrees), Eigenfaces approach substantially outperforms Fisherfaces approach as the latter yields very diverse accuracy in function of number of samples per identity and in function of trained identities number (Fig. 10). It also appeared that PCA-based approaches are highly sensitive to pose variation but they have nearly constant accuracy ratio along increasing sizes of training databases. Repeating simulations for the second set (the one with mimic variations and occlusions) confirmed the earlier observations, although overall accuracy decreased to 50% in best case.

Additionally, an optimal number of meaningful eigenvectors for both methods was investigated. It was observed that PCA-based algorithms perform with steadily increasing accuracy along with increasing number of meaningful eigenvectors. In case of FLD-based method, there is a threshold below which the accuracy is significantly below 50%. It was also observed that FLD accuracy drops rapidly if all eigenvectors are kept. For satisfactory scores in FLD it is necessary to remove c.a. 2% of the least meaningful eigenvectors.

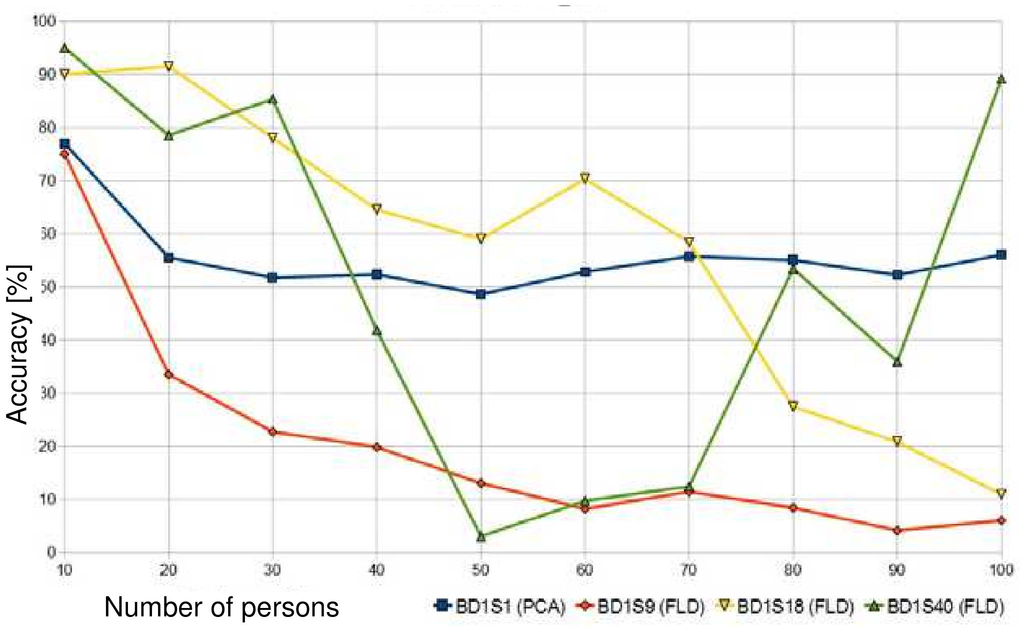


Fig. 10. Comparison of Fisherface (FLD) approach accuracy vs. Eigenface (PCA) method in function of number of identities in training set, for different numbers of samples per identity (PCA - 1, FLD - 10, 20, 40); only pose variation of 22 degrees applied

Although the results of the tests performed so far are promising, the algorithm is not ready yet for implementation in the proposed framework in real-world scenarios. Further research is needed in order to improve accuracy of the face matching algorithm and efficiency of the

3D face model construction. A larger database of faces will be constructed and used for testing face recognition algorithm against results of face detection performed in monitoring system camera images. It has be also noted that due to complexity of the algorithm and very high memory requirements, it seems reasonable to use a computer cluster for concurrent face recognition in a large number of cameras.

6. License plate detection and number recognition

A module for detection of license plate and recognition of license number is another example of a specialized algorithm that may be useful for automatic event detection in the proposed framework. For example, if a parked car is detected in the camera image, the module may read the license plate number and verify it in the database, in order to check e.g. if the vehicle is authorized to park in the particular place. In secured areas, the module may be used to register all vehicles entering and leaving the area. Moreover, if the system has access to an appropriate database, it is possible to detect stolen cars or traffic law violations. Performance of the algorithm depends on resolution of the image, placement of the camera and its view angle. Other factors such as speed of the passing cars, light and weather conditions may also require special approach.

Detection of a license number plate location is a problematic issue. Approaches to the detection problem are mainly based on combination of morphology and edge detection (Dong et al., 2006) which are fast, but detection errors occur when complex background is present. Methods using color features (Shi et al., 2005) are usually sensitive to light changes and night conditions. Algorithms that use Hough transform (Liu & Luo, 2010) usually need some restriction to image size or background.

The proposed algorithm is a solution which uses mainly color information about car back red light, edge detector and morphology. In this case, processed color does not change visibly according to light conditions. The algorithm was tested in real-world conditions in the Gdansk University of Technology campus, with a fixed camera pointed at an entrance gate, where the back of a car stopping at the gate is visible in the camera view. The size of the image was 704x576 pixels and the frame rate of the camera was 25 images per second. The expected size of the number plate in this image was 100x25 pixels.

The module operates on results of object detection and tracking, described earlier. Two processing stages are employed: license plate detection in the image of a moving object classified as a vehicle and license number recognition using optical character recognition (OCR). The plate detection part of the algorithm is responsible for finding coordinates of the number plate in a single video frame and for transmitting the results to the OCR module. For proper localization, the information about plate size and camera view angle is needed. Both parameters are constant in a given setup. It is also assumed that a car is visible for at least 1 second, which results in at least 25 images of the vehicle (the camera's fps rate is 25). The condition is met in the mentioned situation, when a car stops for a few seconds before passing through a gate.

The plate detection algorithm implements several image processing operations such as Sobel filtering, mathematical morphology or contours finding. The input image is extracted from the video frame. The output data is an integer vector of plate coordinates. A block diagram of the detection algorithm is shown in Fig. 11. The proposed approach is based on detecting red back lights and finding a location of the license plate in a space between them. First, the input image is rotated according to camera position. In order to reduce noise influence, the image is blurred using Gaussian smooth operator. Next, red areas are detected by finding pixels with dominant red channel value. For 8-bit per channel image, the threshold is set to 40. The selected red areas are saved as separate objects represented as contours. Car lights must be of appropriate size. Therefore, the area of every contour is checked in relation to the zoom of the camera and the expected size of the light. If a contour size is too small, it is not included in further processing. In the described setup the lights should be larger than 200 pixels. In case of finding a very large contour (in our case bigger than 5000 pixels), it is assumed that a red car is present. If at least two contours with correct sizes are found, the contour data is processed by the plate detection block.

In all valid contours, the alignment of every possible contour pair is checked. If the contours are in proper distance from each other and a horizontal line can be drawn between them, they are considered the contours of car back lights. For given conditions, the tolerance of horizontal placement is about 30 pixels. The distance between contours should be in the range of 2–4 plate lengths. If a large red contour is detected, it is assumed that it represents the whole car or a part of the car. In the latter case, the contour needs to be expanded to cover the whole vehicle. In both situations, the result of contour processing is a region that contains the license plate. In the next step, the original image is processed with Sobel operator and then dilated. As a result of this operation, an image containing vertical edges is obtained. The number plate is searched for in this image by moving a window vertically through the area containing the plate. The height of this window equals the expected height of a number plate. The result of this search is a rectangle with highest vertical edge concentration. Next, this area is swept horizontally with a window having width equal to the expected plate width. If hard edges, which represent the border of the number plate, are detected, the detection algorithm finishes its work and the rectangle covering the number plate is finally obtained. In example shown in Fig. 12, the result of number plate detection is presented. Red contours, including lights, are visible. The area where the number plated is searched for is marked by a blue rectangle and the final detected region is marked by a green rectangle.

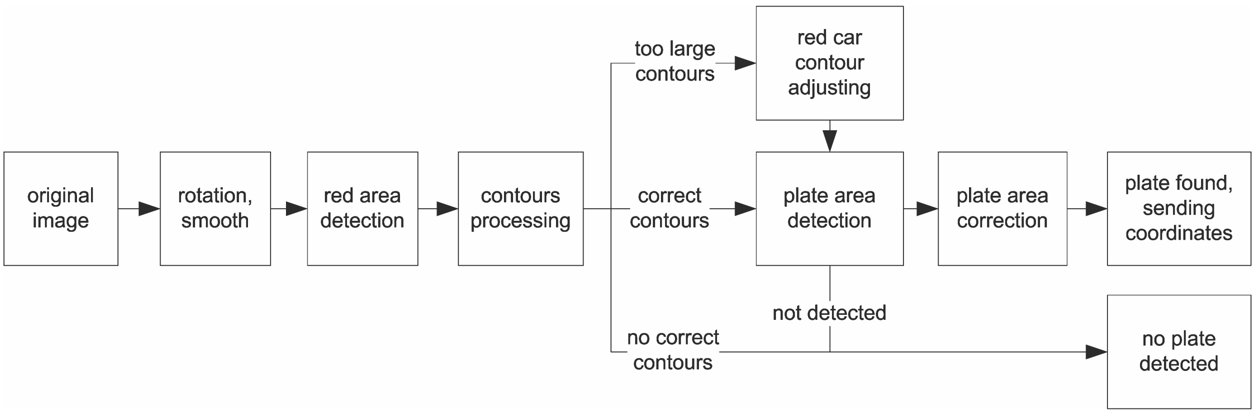


Fig. 11. Block diagram of the license plate detection algorithm

The area of the image containing the detected number plate is used by the OCR module to recognize the license number of a vehicle. The open source library GOCR was implemented in the presented module for recognizing characters. The GOCR algorithm can be divided into four parts. First, the image is converted to greyscale. Then, thresholding of the image is

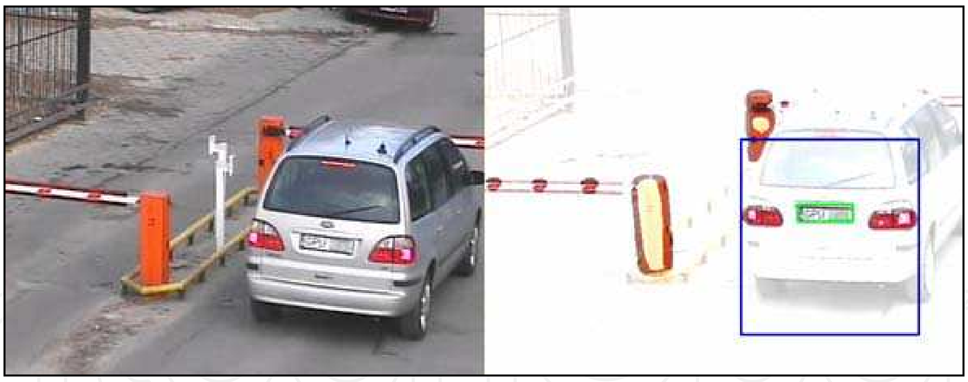


Fig. 12. Example of the license plate detection: original frame from the camera (left) and the processed image (right)

performed. Next, the cluster detection algorithm detects areas in the image which may contain letters. In the final step, the detected clusters are compared with the patterns in the database. The performance depends on several configuration parameters of the GOCR engine, such as:

- *grey threshold* – the maximum luminance of a pixel (0 to 255), which is recognized as black (automatic setting of this parameter may be used);
- *dust size* – maximum size (in pixels) of a detected cluster which is not treated as a character; since the size of the number plate's area is at least 100x25 pixels and there are no lowercase characters, the characters are expected to occupy at least 100 pixels;
- *certainty* – each decision of recognizing a detected cluster as a letter depends on the certainty threshold (0-100) required to accept the decision; in the presented example, high certainty of recognized numbers is required, therefore the parameter is set to 95;
- *char filter* – list of characters that are expected to be found in the image; in this case the possible characters are capital Latin letters (A-Z) and numbers (0-9).

The use of the database has a large influence of the efficiency of the OCR engine. Prior to recognizing number plates, the engine needs to be trained with examples of number plates. Some images in the database are distorted with noise and blur and simulate different light conditions.

The recognition of characters usually comes with some errors, due to light conditions or insufficient resolution of the image. Therefore, some processing of the results is needed to increase the certainty of the final decision. In the first stage, the output of the GOCR engine is verified using certain rules concerning number plates. It is common that zeros, ones or blank spaces are inserted at the beginning of the registration number. Such characters need to be deleted, since number plates in Poland have to begin with a capital letter. The other aspect is the length of the detected character string. Polish registration numbers are always 7 characters long with a space after the second or third character. Therefore, if the string recognized by the OCR engine is longer, it needs to be cropped. For further improvement, a majority voting of the results is performed. OCR is performed independently for each frame in which a car is present in the view. The resulting char arrays are stored in a result buffer. After the buffer of 25 OCR results (one second of video stream) is filled, majority voting check is evaluated. The final number plate string is the most frequently appearing in the result buffer.

The results of the tests indicate that the proposed module provides satisfactory level of recognition accuracy. Most of the errors result from inaccuracy of the plate detection part of the algorithm. Therefore, future work will concentrate on improving the procedure for license plate detection, especially in difficult conditions (adverse lighting, dimmed car lights, etc.).

7. Event detection

An event detector is the final module in the proposed framework (Fig. 2). It utilizes results obtained during the earlier analysis stages in order to check whether defined rules that describe important events are fulfilled. In other words, this module interprets the data concerning all the objects visible by the camera and tests if a situation that may be a potential security threat has occurred. If this is the case, further actions are performed, e.g. notification is sent to the system operator (visual alert, logging or using a camera to view the area in which the event was observed).

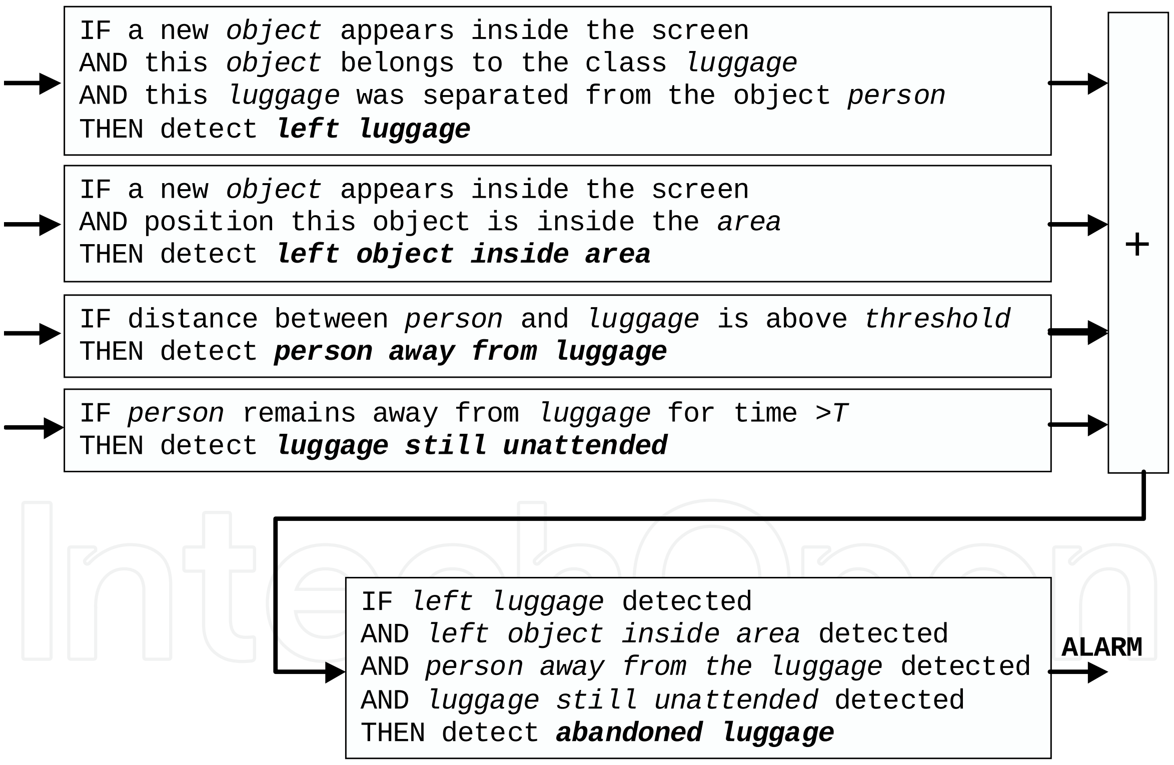
The system for event detection is rule-driven. The rules may be selected by the operator from a predefined list of typical events or they may be constructed using a graphical user interface by combining elementary conditions and setting their parameters, so that the rules allow for detection of any desired situation. In the system developed by the authors, the event detection module uses only deterministic IF-THEN conditions. In future work, application of more advanced rule interpretation systems, e.g. based on fuzzy logic is planned. This will provide more flexible approach for determining whether a complex condition is fulfilled.

The rule interpretation system operates on data provided by the analysis modules situated lower in the framework hierarchy. This data may include positions of moving objects, their current state (size, velocity, etc.), type (class and subclass) and specific object information (recognized person name or license plate number). The event detector also analyses interactions between objects and the scene (e.g. whether the object is inside a defined area), as well as interactions between objects themselves (e.g. whether two objects are close to each other). Additionally, the event detector uses information about previously detected events (e.g. whether a particular event has occurred recently), so there is a feedback loop in this module. Therefore, it may be stated what while other modules analyze the camera image to provide information about what happened in the observed scene, the event detector interprets this data and informs the system operator what does this situation mean and whether it is a potential security threat.

In the presented framework, the authors divided the event detector into two parts. The low-level event detector operates on data provided by other modules and detects elementary events. The high-level detector analyses the detected low-level events, maintains history of the previous events and notifies the operator on detected security threats. For example, the low-level module will analyze the data provided by the object detector, tracker and classifier modules and it will detect the event described as 'a vehicle inside the defined area'. The high-level detector will check how long the vehicle remains inside the area, what type of area it is and whether the vehicle is permitted to park in the area. If the conditions are fulfilled, the operator will be notified with an alarm such as 'the car XX1234 is parking inside the Restricted Zone 1 for longer than 3 minutes without an authorization'.

There is a large number of potential security threats for which high-level detection rules may be created. Typical events that are commonly regarded as security threats include trespassing, loitering, burglary, theft, assault, vandalism (e.g. graffiti painting or destroying bus stops), etc. Detection of luggage that was left unattended in public spaces such as airports or railway stations is of particular interest, as it often results in closing the facility, evacuation of people and costly action of special forces. Therefore, there is a strong need to implement a detection of abandoned luggage in monitoring systems. A test version of detector for this type of events was developed and implemented by the authors (Szwoch et al, 2010). It will be presented here as an example of the working automatic event detector.

Detection of abandoned luggage using a high-level rule is possible if several low-level events have been detected before. The first condition is that a person has to leave a luggage. This is detected if an object of class 'person' (or 'person with luggage', if objects subclasses are used) is split into two objects: 'person' (now without luggage, but with the same tracker still assigned to it) and 'luggage' (which remains stationary and receives a new tracker). The second rule checks if the luggage remains within an area in which detection is performed. The third rule is fulfilled if distance between the person and the luggage exceeds the threshold. The fourth rule tests whether the person does not return to the left luggage for a defined period. The final, high-level rule examines all the low-level events detected in a number of last analyzed camera frames. If all four rules described above were fulfilled, the high-level event described as 'a person left unattended luggage in area A' is detected and notification is sent to the system operator. All the rules described here, written in natural language using IF…THEN clauses, are presented in Fig. 13.



IF a new *object* appears inside the screen
AND this *object* belongs to the class *luggage*
AND this *luggage* was separated from the object *person*
THEN detect ***left luggage***

IF a new *object* appears inside the screen
AND position this object is inside the *area*
THEN detect ***left object inside area***

IF distance between *person* and *luggage* is above *threshold*
THEN detect ***person away from luggage***

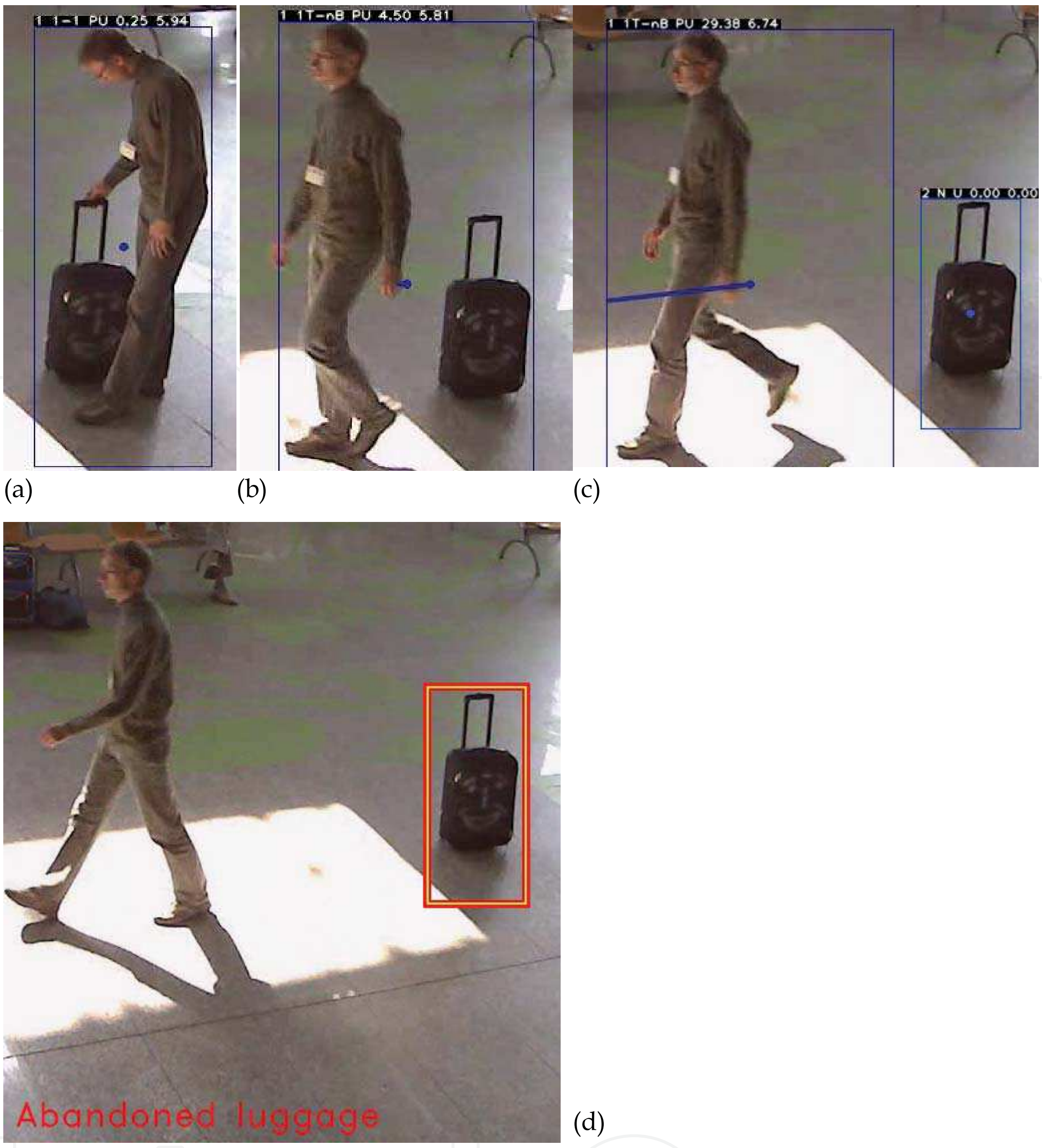
IF *person* remains away from *luggage* for time >*T*
THEN detect ***luggage still unattended***

IF *left luggage* detected
AND *left object inside area* detected
AND *person away from the luggage* detected
AND *luggage still unattended* detected
THEN detect ***abandoned luggage***

Fig. 13. Rules for detection of abandoned luggage

An example of abandoned luggage detection in real-life application, in the Poznan-Lawica airport in Poland, is presented in Fig. 14. It can be seen that all the necessary low-level events are detected, which results in detection of abandoned luggage using the high-level rule and visual alert on the screen. Results of tests performed at the airport proved that the proposed event detection module works correctly in simple situations, when the view on the person leaving luggage is not obstructed. However, performance of the event detector

IntechOpen



(a) (b) (c)

(d)

Fig. 14. An example of abandoned luggage detection: (a) initial state, (b) detected left luggage, (c) detected person away form luggage, (d) detected luggage still unattended, high-level event 'abandoned luggage' detected, visual alert

depends greatly on accuracy of the data provided by low-level modules, especially the object detector and the object tracker. In case of high number of conflicting objects (e.g. during rush hours at the airport) and adverse lighting conditions (e.g. reflections on the floor), a high number of errors in object detection and tracking significantly decreased the accuracy of event detection. Therefore, further work will be focused on improving object detection and tracking in difficult conditions.

Whenever a potentially dangerous event is detected by the video analysis framework, it is useful to provide a detailed view of the situation to an operator by aiming a PTZ camera at an object of interest and tracking it automatically as it moves. For example, if abandoned luggage is detected, the camera may track the person that left the luggage. This task requires finding pan p, tilt t and zoom z settings for a dome camera that will guarantee that an object of a known location will be present in a video stream from the camera (Szwoch & Dalka, 2010). Object position is obtained from object detection and tracking modules that analyze video streams from fixed cameras. The position must be expressed in real-world coordinates (e.g. meters), therefore the fixed camera has to be calibrated. Furthermore, the PTZ camera position and object location must be defined in a common, real-world, Cartesian 3D coordinate system. Alternatively, a conversion between both coordinate systems has to be available.

It is assumed that all objects move on the ground plane, therefore an object height coordinate is equal to zero. Object position is defined as (x, y, 0) and current velocity of the object along both axes is denoted as v_x and v_y. A dome camera settings include its position (x_c, y_c), height above the ground h_c, and pan offset α_c, that is defined as an angle (in degrees) between the Y-axis of the world coordinate system and camera's zero pan position (clockwise).

There is a significant delay in a system caused by video processing, data transmission and executing PTZ command by the camera. This delay must be compensated in order to assure that a fast-moving object is always present in a video frame from the PTZ camera. System delay compensation is performed by setting the PTZ camera to the predicted position of the object. Prediction time should be equal to the total delay in the system. A linear predictor is used that estimates object position based on its current estimated speed and heading (direction of movement).

Predicted object position ($\hat{x}$, $\hat{y}$, 0) is given with equations:

$$\hat{x} = x + d \cdot v \cdot \sin(\Theta) \qquad \hat{y} = y + d \cdot v \cdot \cos(\Theta) \tag{11}$$

where d is the system delay in seconds and v and Θ are object's current speed and heading calculated as follows:

$$v = \sqrt{v_x^2 + v_y^2} \qquad \Theta = \operatorname{atan2}(v_y, v_x) \tag{12}$$

The pan p and tilt t parameters for the camera are calculated with equations:

$$p = 90 - \operatorname{atan2}(\hat{y} - y_c, \hat{x} - x_c) - \alpha_c \qquad t = \begin{cases} -\arctan\left(\dfrac{h_c}{\sqrt{(\hat{x} - x_c)^2 + (\hat{y} - y_c)^2}} \right) & \hat{x} \neq x_c \vee \hat{y} \neq y_c \\ -90 & \text{otherwise} \end{cases} \tag{13}$$

The last camera parameter setting, the zoom, is set based on the object's distance from the PTZ camera. The closer the object is to the camera, the smaller is the zoom value. This approach assures that object dimensions in a video stream remain more or less constant.

8. Conclusions and future research

The multi-stage video analysis framework proposed by us and described in this chapter is a flexible and efficient solution for automatic analysis of camera images in the monitoring systems. The framework is intended mainly for automatic event detection in video. It was shown that successful event detection requires performing several stages of image processing. Some of the modules, such as object detection, tracking and classification, are necessary for event detection, because the rules describing potential security threats require data provided by these modules. Other modules, such as face recognition or license number recognition, are supplementary and they enhance the framework, providing additional data for event detection and allowing use of more complex detection rules, e.g. searching for a particular person or a vehicle. A flexible structure of the framework makes it possible to adapt the system to different needs, adding new modules in the future.

The functional modules described in this chapter were implemented in the current version of the framework that was used for preliminary testing. Most of these modules utilize algorithms found in the literature, enhanced and modified to suit the needs of the framework. The main contribution of the authors, apart from the design of the framework and method of data exchange between modules, is implementation of the object tracking algorithm with resolving of tracking conflicts, the algorithm for vehicle classification, the method for creating database of 3D face models and using them in face recognition, and a structure for interpretation of event detection rules.

It has to be noted that errors that occur at the lowest level of image processing significantly decrease the accuracy of event detection at the highest analysis level. This was observed in the test system for automatic detection of abandoned luggage at the Poznan airport in Poland (Szwoch et al, 2010). As long as the number of moving objects is small and the lighting is good, the event detection is successful. However, with a large number of moving persons and unfavorable light, errors in object detection and tracking are propagated throughout the framework, resulting in undetected events at the system output.

The future research will be aimed mainly at improving and enhancing the functional modules. The possible areas of improvement were indicated earlier for each algorithm presented in this chapter. Current work is focused mainly on reduction of number of errors in the object detection and tracking phases, by improving the background subtraction procedure (elimination of artifacts resulting from adverse lighting conditions) and by enhancing the procedure for resolving tracking conflicts (e.g. using feature matching algorithm for finding the object in the image region). We believe that with these enhancement, accuracy of the event detection will be improved significantly.

Apart from enhancing the existing modules, new modules will be developed and added to the framework. There is already an ongoing research on module for analysis and prediction of crowd behavior. It will allow for detection of advanced situations, such as a potential fight between groups of football fans, the riot on the street or a panic situation.

One possible enhancement of the framework that was not addressed so far is concurrent analysis of images from multiple cameras. Real video monitoring systems consist of a large number of cameras. Therefore, implementation of an algorithm for tracking movement of objects between cameras (capturing a known object as it enters a field of view of the camera) is planned for the next stage of research. Moreover, after enhancing the framework, tests will be performed that will provide quantitative assessment of its performance.

Video content analysis in large monitoring systems, e.g. in the modern football arena with several hundreds of cameras, imposes very high demand for the processing power, memory and data storage. Therefore, the framework was designed in such a way that it is possible to perform the video analysis using either a centralized 'supercomputer' cluster or a distributed network of typical personal computers ('node stations'), so that available processing resources may be utilized optimally.

The proposed framework, after enhancing its functional modules and the framework it self, will provide an useful solution for automatic detection of a wide range of events that may be potential security threats. It should noted here that the framework is not intended to be a fully automatic surveillance system that is able to detect every event occurring in the observed area. The system operator will still be the one that makes decisions and the framework will be only an automated 'assistant' that notifies the operator on the detected events. This system may also help in shifting the focus in modern monitoring systems from reviewing the recordings after an event occurred to a real-time identification of security threats, resulting in improved level of public security in the monitored areas.

9. Acknowledgements

Research funded within the project No. POIG.02.03.03-00-008/08, entitled "MAYDAY EURO 2012 – the supercomputer platform of context-depended analysis of multimedia data streams for identifying specified objects or safety threads". The project is subsidized by the European regional development fund and by the Polish State budget.

10. References

Ashbrook, A.P.; Thacker, N.A. & Rockett, P.I. Multiple shape recognition using pairwise geometric histogram based algorithms, *IEEE 5th International Conference on Image Processing and its Applications*, pp. 90-94, Edinburgh, July 1995

Bay, H.; Tuytelaars, T. & Van Gool, L. (2006). SURF: Speeded up robust features, *9th European Conference on Computer Vision*, Graz, May 2006

Czyzewski, A. & Dalka, P. (2007). Visual traffic noise monitoring in urban areas. *Int. Journal of Multimedia and Ubiquitous Engineering*, Vol. 2, No. 2, pp. 91–101

Czyzewski, A. & Dalka, P. (2008). Examining Kalman filters applied to tracking objects in motion, *9th International Workshop on Image Analysis for Multimedia Interactive Services*, pp. 175-178, Klagenfurt, May 2008

Dalka, P. (2006). Detection and segmentation of moving vehicles and trains using Gaussian mixtures, shadow detection and morphological processing. *Machine Graphics and Vision*, Vol. 15, No. 3/4, pp. 339–348

Dalka, P. & Czyzewski, A. (2010). Vehicle classification based on soft computing algorithms, *7th Conf. on Rough Sets and Current Trends in Computing*, Warsaw, June 2010

Daugman, J. G. (1988). Complete Discrete 2-D Gabor transforms by neural networks for image analysis and compression. *IEEE Trans. Acoustic, speech and signal processing*, Vol. 36, No. 7, pp. 1169-1179

Dong P., Yang J-h & Dong J-j. (2006). The application and development perspective of number plate automatic recognition technique, *Proc. 2nd Conference on Information and Communication Technologies ICTTA '06,* pp. 744-747, October 2006, Damascus

Dougherty, E. & Lotufo, R. (2003). *Hands-on morphological image processing*, SPIE Press

Elgammal, A.; Harwood, D. & Davis, L. (2000). Non parametric model for background subtraction. *Lecture Notes in Computer Science*, Vol. 1843, pp. 751-767

Fisher, R.A. (1936). The use of multiple measures in taxonomic problems. *Ann. Eugenics*, Vol. 7, pp. 179-188

Flusser, J. & Suk, T. (2006). Rotation moment invariants for recognition of symmetric objects. *IEEE Transactions on Image Processing*, Vol. 15, No. 2, pp. 3784–3790

Grigorescu, S.E.; Petkov, N. & Kruizinga, P. (2002). Comparison of texture features based on Gabor filters. *IEEE Trans. on Image Processing*, Vol. 11, No. 10, pp. 1160-1167

Horprasert, T.; Harwood, D. & Davis, L. (1999). A statistical approach for real-time robust back-ground subtraction and shadow detection, *Proc. of IEEE Frame Rate Workshop*, pp. 1–19, Kerkyra, Greece, September 1999

Hsu, C-W.; Chang, C-C. & Lin, C-J. (2003). A practical guide to support vector classification, Technical report. Dept. of Computer Science, National Taiwan University

Kim, H-K.; Kim, J-D.; Sim, D-G. & Oh, D-I (2000). A modified Zernike moment shape descriptor invariant to translation, rotation and scale for similarity-based image retrieval, *IEEE Int. Conference on Multimedia and Expo*, Vol. 1, pp. 307-310, New York, July 2000

Konrad, J. (2007). Videopsy: Dissecting visual data in space time. *IEEE Communication Magazine*, Vol. 45, No. 1, pp. 34-42

Li, H. & Ngan, K. (2007). Automatic video segmentation and tracking for content-based applications, *IEEE Communication Magazine*, Vol. 45, No. 1, pp. 27–33

Liu, Y. & Zheng, Y. (2005). Video object segmentation and tracking using y-learning classification, *IEEE Trans. Circuits and Syst. For Video Tech.*, Vol. 15, No. 7, pp. 885-899

Liu Ch-Ch. & Luo Z-Ch. (2010). An extraction algorithm of vehicle license number using pixel value projection and license plate calibration, *2010 International Symposium on Computer Communication Control and Automation (3CA)*, pp. 256-259, Tainan, May 2010

Martinez, J.M (2004). *MPEG-7 overview (version 10)*. MPEG Consortium, Palma de Mallorca

Shi X., Zhao W. & Shen Y (2005). Automatic license plate recognition system based on color image processing, In: *Computational Science and Its Applications*, Vol. 3483, Ed. O. Gervasi et al., Springer-Verlag, New York

Stauffer, C. & Grimson, W. (2000). Learning patterns of activity using real-time tracking. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 8, pp. 747–757

Szwoch, G.; Dalka, P. & Czyzewski, A. (2009). Estimation of object size in the calibrated camera image. *Elektronika*, Vol. 50, No. 3, pp. 10-13.

Szwoch, G. & Dalka, P. (2010). Automatic detection of abandoned luggage employing a dual camera system, *3rd IEEE Int. Conf. on Multimedia Communications, Services & Security*, pp. 56-61, Krakow, May 2010

Szwoch, G.; Dalka, P. & Czyzewski, A. (2010). A framework for automatic detection of abandoned luggage in airport terminal, *3rd International Symposium on Intelligent and Interactive Multimedia: Systems and Services*, Baltimore, July 2010

Tsai, R. (1987). A versatile camera calibration technique for high accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal of Robotics and Automation*, Vol. 3, No. 4. pp. 323-344

Turk, M. & Petland, A (1991). Eigenfaces for recognition. *Journal of Cognitive Neurosicence*, Vol. 3, No. 1, pp.71-86

Xiu, L.; Landabasso, J. & Pardas, M. (2005). Shadow removal with blob-based morphological reconstruction for error correction, *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. II–729–732, Philadelphia, March 2005

Vetter, T. & Blanz, V. (1999). A morphable model for the synthesis of 3D faces, *Computer Graphics Proceedings. Siggraph 1999*, pp. 187–194, Los Angeles, August 1999

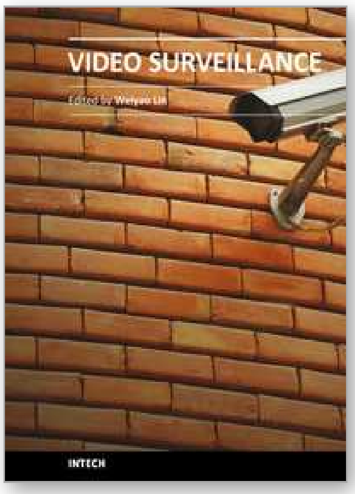
Viola, P. & Jones, M (2001). Rapid object detection using boosted cascade of simple features, *IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 1, pp. 511-518, Kauai, December 2001

Welch, G. & Bishop G. (2006). *An introduction to the Kalman filter*. Technical report, TR-95041, Department of Computer Science, Univ. of North Carolina, Chapel Hill

Yang, T.; Li, S.; Pan, Q. & Li, J. (2004). Real-time and accurate segmentation of moving objects in dynamic scene, *ACM 2nd International Workshop on Video Surveillance and Sensor Networks*, pp. 136-143, New York, October 2004

IntechOpen

IntechOpen



Video Surveillance
Edited by Prof. Weiyao Lin

ISBN 978-953-307-436-8
Hard cover, 486 pages
Publisher InTech
Published online 03, February, 2011
Published in print edition February, 2011

This book presents the latest achievements and developments in the field of video surveillance. The chapters selected for this book comprise a cross-section of topics that reflect a variety of perspectives and disciplinary backgrounds. Besides the introduction of new achievements in video surveillance, this book also presents some good overviews of the state-of-the-art technologies as well as some interesting advanced topics related to video surveillance. Summing up the wide range of issues presented in the book, it can be addressed to a quite broad audience, including both academic researchers and practitioners in halls of industries interested in scheduling theory and its applications. I believe this book can provide a clear picture of the current research status in the area of video surveillance and can also encourage the development of new achievements in this field.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Andrzej Czyżewski, Grzegorz Szwoch, Piotr Dalka, Piotr Szczuko, Andrzej Ciarkowski, Damian Ellwart, Tomasz Merta, Kuba Łopatka, Łukasz Kulasek and Jędrzej Wolski (2011). Multi-Stage Video Analysis Framework, Video Surveillance, Prof. Weiyao Lin (Ed.), ISBN: 978-953-307-436-8, InTech, Available from: http://www.intechopen.com/books/video-surveillance/multi-stage-video-analysis-framework

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2011 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.