Multimedia Tools and Applications manuscript No.
(will be inserted by the editor)

MetroSurv: Detecting Events in Subway Stations

Barbara Krausz · Rainer Herpers

Received: date / Accepted: date

Abstract Video surveillance has become a hot research topic due to the recently increased importance of safety and security issues. Usually, security personnel has to monitor a surveillance area and often they have to do this for 24 hours a day. Thus, it would be desirable to develop intelligent surveillance systems that support this task automatically. The system described in this contribution is thought of such an automatic surveillance system that has been developed to detect several dangerous situations in subway stations. The workflow and the most important steps from foreground segmentation, shadow detection, tracking and classification to event detection are described, discussed and evaluated in detail. The developed surveillance system yields satisfying results, as dangerous situations that need to be recognized are detected in most cases.

Keywords Video Surveillance · Foreground Segmentation · Shadow Detection · Tracking · Event Detection · Expert System

1 Introduction

Video surveillance is in the center of research at the moment. So far there are already many different application fields for video surveillance systems. For example in almost every shop there are cameras to observe customers and recognize shoplifting. Another field of application is traffic monitoring where the objective is to analyze and regulate traffic flow.
Nowadays, there are many projects dealing with the observation of humans or crowds

Barbara Krausz
Fraunhofer IAIS
Schloss Birlinghoven
53754 Sankt Augustin, Germany
E-mail: barbara.krausz@iais.fraunhofer.de

Rainer Herpers
Bonn-Rhein-Sieg University of Applied Sciences
Grantham-Allee 20
53757 Sankt Augustin, Germany
E-mail: rainer.herpers@fh-bonn-rhein-sieg.de

of humans in public spaces (see for example Gryn et al (2009)). In the busiest shopping area in London (Oxford Street), there are 35 cameras observing customers (see McCahill and Norris (2002)). On the Central Line of the underground in London, 500 cameras are installed. All these cameras have to be monitored by security personnel which is a time-consuming and very expensive task. Thus, it is desirable to develop systems that assist human operators monitoring the scenes or even operate independently. For instance, a surveillance system may compute alarms on suspicious events and advise a human operator to pay attention to a particular screen.
The described system called *MetroSurv* is thought of such an automatic surveillance system that is able to detect several dangerous situations in a subway station. The sample video sequences this system has been based on have been provided by the largest European public transportation company (RATP, France) within the *International Conference on Advanced Video and Signal based Surveillance 2005*, see CREDS (2005).
These video sequences have been recorded at the subway station "Porte de Lilas" in Paris and show several dangerous situations which need to be detected in real-time:

- *Proximity warning*: If a person steps onto the white line near the platforms or extends an arm, leg etc. over the rails, a proximity warning should be triggered.
- *Dropping objects on tracks*: This warning shall be raised if a person drops any kind of object onto the tracks.
- *Launching objects across the platforms*: If a person throws an object across the platforms, an alarm should be triggered.
- *Person trapped by the door of a moving train*: The surveillance system has to recognize if a person is trapped by the door of a train. False negative detections are not tolerated.
- *Walking on rails*: A critical alarm should be raised, if a person is walking along the tracks.
- *Fall on the track*: The surveillance system has to detect people that fall on the tracks.
- *Crossing the rails*: If a person crosses the tracks, the surveillance system has to raise a critical alarm after having recognized a "fall on the track" event.

This paper discusses all modules of *MetroSurv*. For an overview see Figure 1. Further details about the system can be found in Krausz (2007). *MetroSurv* firstly detects moving objects by classifying each pixel as foreground or background using a Gaussian Mixture Model approach. This step also includes the detection and removal of shadows, as motion detection algorithms may misclassify shadows as belonging to the foreground. Shadow detection is realized by introducing a reflection model that describes the physics of light and shadow formally. Objects detected during motion detection are tracked through the video sequence by establishing correspondences between objects of the current frame and objects found in the previous frame. Theren, the detected moving objects are classified into the classes "Human", "Train" or "Object" using a rule-based approach. In a high-level layer the system analyzes and interprets human motion. For that purpose, an expert system is utilized which combines information about detected objects with a priori knowledge to conclude what is happening in the scene. If one of the predefined situations occurs, an alarm is raised.
The main contributions of this paper include firstly a high-level feedback scheme to the foreground segmentation module to cope with situations like sudden illumination changes. Secondly, a novel shadow detection algorithm is proposed that is based on a

reflection model. Thirdly, an event recongition method is developed which makes use of an expert system to analyze the events taking place in the scene.
The paper is organized as follows. The next section presents related work. Then, section 3 gives an overview of the structure of *MetroSurv*. Section 4 briefly describes the foreground detection module and section 4.2 explains the usage of high-level feedback which is utilized to improve the results of the foreground segmentation. In section 4.3, the problem of detecting and removing shadow regions is addressed and discussed. The tracking procedure is described in section 5. After the classification step (see section 6) the event detection module is presented in section 7. Finally, section 8 presents obtained results before a conclusion is drawn in section 9.

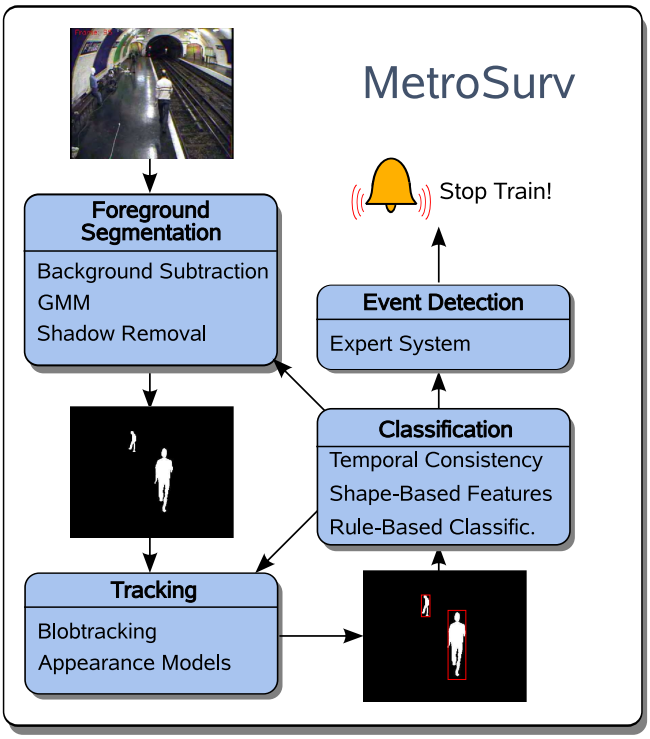

Fig. 1 Overview of *MetroSurv*. Firstly, *MetroSurv* performs foreground segmentation to detect moving objects. Moving objects are tracked continuously throughout the video sequence. Additionally, each detected object is classified as a human, train or unknown object. The results of the classification and the tracking module are used in the event detection module to analyze the activities. If one of the predefined events is detected, an alarm is triggered.

2 Related Work

A short overview of several foreground segmentation approaches is given in Hu et al (2004). *Temporal differencing* uses the fact that two subsequent frames are slightly different in regions containing moving objects. Lipton et al (1998) for example compute

the difference between successive frames and detect motion, if this difference exceeds a certain threshold. Motion can also be detected by considering the *optical flow*, see for example Giachetti et al (1998). However, the most commonly used technique is *background subtraction*. Pixels are classified as foreground, if the difference between the background model and the current frame at a certain location exceeds a threshold. The accuracy of this method highly depends on the modeling of the background. Haritaoglu et al (1998) estimate the background model in a training period over several video frames when no people are visible. However, in this case the background model does not adapt to scene changes, e.g. if the illumination conditions change. Therefore, this approach does not qualify for an application. This problem is addressed in the tracking system *Pfinder* described in Wren et al (1997) where the color distribution is assumed to be Gaussian. In each frame mean intensity value and standard deviation of the Gaussian are recursively updated. A method superior to these techniques has been presented in Stauffer and Grimson (1999) where a mixture of Gaussian functions is used to model each pixel. *MetroSurv* is motivated by the Gaussian mixture model approach and extends it by making use of high-level feedback similar to the approach proposed in Barth and Herpers (2005) which enables us to adapt quickly to scene changes. Furthermore, a shadow detection module is integrated into the foreground segmentation module. For shadow detection, two main approaches exist, see Salvador (2004): *Model-based* techniques rely on a priori knowledge of the scene. With knowledge of the illumination conditions, the geometry of the environment and the objects (peopl, cars etc.) present in the scene they can estimate the position of cast shadows. *Property-based* techniques on the other hand just consider general properties of shadows and make use of *local*, *spatial*, *geometric* or *temporal* features. As a local feature, color and brightness distortion are considered in Horprasert et al (1999) whereas different color spaces are used in Cucchiara et al (2001) and in Salvador (2004). Other approaches examine edges or texture as spatial features, see for example Frahm et al (2003). They assume that the texture of a surface covered by a shadow is not changed. Geometric features are mainly used to reduce false positives by taking the environment into account (see Hsieh et al (2004)). Salvador (2004) utilizes a temporal feature when considering two shadow regions in consecutive frames as instances of the same shadow, if both regions overlap. Our approach integrates a local feature based on the reflectance and a geometric feature in a post processing step similar to the approach in Salvador (2004).

Detected objects are tracked in subsequent frames. Tracking can be performed in an iterative procedure or using Bayesian theory. The Mean Shift algorithm (see Comaniciu and Meer (1999)) for instance is a gradient-based approach that starts with the previous position of an object, evaluates the correctness by computing the distance between the target model and the candidate model and iteratively refines the position. On the contrary, probabilistic approaches like Particle Filters (see Isard and Blake (1998)) are non-iterative and rely on Bayesian theory. They estimate a probability distribution representing the state of a dynamically changing system by evaluating observations. *MetroSurv* integrates an approach proposed by Senior (2002) which establishes correspondences between detected objects of the previous and the current frame by computing the spatial distance and the distances between the appearance by comparing appearance models which are based on a color model and a probability mask.

Objects that have been detected and tracked in subsequent frames are classified into different object classes. Hu et al (2004) distinguishes two main categories of classi-

fication methods: *Shape-based* classification approaches just consider the shape of an object. For that purpose, several features of an object are extracted. Lipton et al (1998) for example use dispersedness of the blob to distinguish between humans and vehicles whereas Zang and Klette (2003) utilize the aspect ratio of the bounding box. Both systems extract those features and use simple rules to classify each object. Collins et al (1999) also compute features of objects (dispersedness, area and aspect ratio) and feed them into a neural network to classify objects into the classes "single human", "multiple humans" and "vehicles". *Motion-based* classification methods use the fact that human motion is periodic in contrast to the motion of rigid objects like vehicles. Time-frequency analysis is used in Cutler and Davis (2000) to analyze the periodicity of motion and classify the object. In *MetroSurv* shape-based features are extracted and used in a rule-based approach for classifying objects.

In order to recognize events taking place in the scene approaches for event detection mostly take scene context into account. Often, predefined events have to be detected. For example, Bremond and Medioni (1998) use *Finite State Machines* that consist of states representing certain sub-scenarios like "the car enters zone A". The whole scenario is recognized if the automaton reaches a final state. Cupillard et al (2004) also employ finite state automata to recognize simple scenarios. For example, the automaton for the scenario "a person jumps over a barrier without validating his ticket" is composed of five states: a person is tracked, the person is at the beginning of the validation zone, the person moves fast, the person is beyond the barrier and the person is at the end of the validation zone. Starner and Pentland (1995) use *Hidden Markov Models* to analyze motion of a hand to recognize American sign language. With Parametrized-HMMs, a more complex approach is used in Wilson and Bobick (1998) to interpret human gestures. One main disadvantage of HMM approaches is the need for training samples. Furthermore, the topology and the number of states of the model have to be determined. In addition to finite state machines, Cupillard et al (2004) use *Bayesian Networks* for the recognition of more complex scenarios. This approach was proposed in Hongeng et al (2000) where for example a Bayesian Network is created and trained for the scenario "object A slows down toward object B". More recently, Muncaster and Ma (2007) apply Dynamic Bayesian Networks. Similar to Hidden Markov Models, training samples are needed to determine the parameters of the network. Ivanov and Bobick (2000) analyze tracking results and generate discrete events like "car-enter" or "person-exit", together with likelihood values. These events form the vocabulary of a *Stochastic Context-Free Grammar* and are processed by a parser to recognize events. Ghanem et al (2004) make use of *Petri Nets* which have been shown to be similar to rule-based expert systems. A transition represents a rule whereas markings correspond to facts. Ghanem et al (2004) describe an ontology for event recognition and shows the use of Petri Nets in the domain of a parking lot. There are several works using *Logical Predicates* in combination with an inference mechanism. Shet et al (2005) and Shet et al (2006), for instance, define rules for activities like "stealing". In each frame the system analyzes the results of the low-level layers and generates facts which are then put into a knowledge base. A Prolog inference engine is then used for reasoning. Similar to Shet et al (2006), *MetroSurv* defines rules for events that need to be detected. Contrarily, we do not use a backward chaining mechanism as Prolog offers, but we make use of an expert system that integrates a priori knowledge of the scene context and combines it with information about detected objects.

3 System Overview

The described system developed for detection of dangerous situations in subway stations is composed of four modules which interact with each other as shown in Figure 1:

- The *Foreground Segmentation* module decides for each pixel of each frame if it is considered as foreground and thus could be part of a moving object. For that purpose, background subtraction is applied. The background image is modeled using a variation of the Gaussian Mixture Models (GMM) approach proposed by Stauffer and Grimson (1999). It extends the original GMM approach by taking a variable number of Gaussian functions to avoid underfitting and overfitting. *MetroSurv* uses an implementation of this GMM-variation by Zoran Zivkovic[1].
  In contrast to many other surveillance applications, *MetroSurv* detects and processes shadows as shadow regions may cause several problems. For instance, a shadow region may connect the foreground regions of two moving objects resulting in a single foreground region or it may even be recognized as a moving object. For shadow detection a novel algorithm is proposed which is based on a reflection model that describes the physics of light and shadow formally.
- The *Tracking* module employs blobtracking to follow moving objects. By modeling the appearance of each object it is even able to track objects that are (partially) occluded.
- The *Classification* module classifies each detected object into the classes "Human", "Train" or "Unknown". For that purpose, it uses features of the object's shape to classify it by means of rules. Additionally, it takes temporal consistency into account. Moreover, the classification results are used as high-level feedback to improve the performance of the foreground segmentation module and the tracking module.
- The *Event Detection* module analyzes the results of the classification and the tracking module and tries to detect the predefined events of CREDS. *MetroSurv* makes use of the rule-based expert system CLIPS[2] that holds the current state of the scene in its working memory and contains rules for detecting events.

4 Foreground Segmentation

The first step of our video surveillance system is the detection of moving objects. For that purpose, each pixel of an incoming video frame I_t has to be classified as part of a moving object (foreground) or as part of the background. The result of the foreground segmentation is a binary image FG_t that marks pixels as belonging to the foreground ($FG_t(x, y) = 1$) or background ($FG_t(x, y) = 0$). All subsequent processing steps depend on the foreground segmentation results, because they just consider foreground regions of the frame. Foreground segmentation is therefore crucial for the quality of a video surveillance system. There are several difficult situations a surveillance system should be able to cope with: For example, background clutter should not be classified as foreground, the appearance of a moving object may be similar to the background or the illumination condition change. In order to improve the results of the foreground segmentation algorithm shadow regions need to be considered during

[1] CvBSLibGMM: http://staff.science.uva.nl/∼zivkovic/Publications/CvBSLibGMM.zip
[2] CLIPS: http://www.ghg.net/clips/CLIPS.html

a special substep, as shadow regions change the appearance of the background and might thus be classified as foreground regions. This causes some problems: A shadow region might be considered as a moving object or two distinct objects (e.g. two people) might merge due to a shadow region. *MetroSurv* addresses this problem by detecting shadow regions and distinguishing shadow regions from real foreground object regions (see section 4.3).

4.1 Adaptive Gaussian Mixture Models

MetroSurv models the background image with a modified version of the Gaussian Mixture Model approach. In contrast to Stauffer and Grimson (1999), it uses a variable number of Gaussian functions as proposed in Zivkovic and van der Heijden (2004, 2006). This approach avoids under- and overfitting of the background model. The foreground segmentation module in *MetroSurv* works as follows: For each new data sample the Euclidean distances to all K Gaussian distributions is computed. For the best matching distribution it is decided, if this distribution belongs to the background by considering its weight as well as its variance. If this distribution belongs to the background, this pixel is classified as a background pixel. The variance and the mean values of all distributions are updated according to Stauffer and Grimson (1999), whereas the weight of a distribution is updated by applying a slightly different equation given in Zivkovic and van der Heijden (2006). If a weight is less than zero, this Gaussian distribution is dropped and K is decremented. If none of the distributions matches, a new Gaussian is introduced with the pixel value as its mean, an initial variance and an initial weight. After that, the weights of the Gaussian distributions are normalized. The output of this algorithm is a classification of the considered pixel as background or foreground pixel. Furthermore, the parameters of the distributions are adapted and can be used for the next incoming data sample.

4.2 High-Level Feedback

MetroSurv improves its foreground segmentation results by utilizing feedback of high-level modules such as the classification module:

- *Changing illumination*: The neon light in subway stations often cause flicker in the image sequence which results in problems in the foreground segmentation module. Similar to Barth and Herpers (2005) this problem is addressed by increasing a parameter that controls the update velocity (α) for all pixels except for pixels belonging to an object classified as a human or a train. Changing illumination condition is detected, if the number of extracted blobs exceeds a threshold which is determined empirically.
- *Changing illumination* due to presence of a train: When a train enters the scene, the illumination in the scene changes drastically, because the train cabin is illuminated and the headlamps of the train illuminate the scene as well. If an approaching train is detected in the scene, the update velocity α for all pixels (except for pixels of a human or a train) is increased as described above, so that the background model adapts quickly to the changed illumination.

- *Ghosting*: *MetroSurv* handles the problem known as waking person or ghosting. If the classification module in *MetroSurv* detects a ghost, α is increased at that location in order to incorporate those pixels quickly into the background model.
- *Incorporation*: When a train does not move for a while, it incorporates into the background model. To avoid this problem the parameter α is set to zero for all pixels belonging to the train. Now, the background model in the local area of the train is not updated and the train is entirely classified as foreground in subsequent frames.

4.3 Shadow Detection

For shadow detection it is advantageous to combine several features. Thus, *MetroSurv* exploits local, spatial and geometric features. Most local features dealing with color spaces are not appropriate for shadow detection in *MetroSurv* because of the high noise level of the video data. One useful local feature is based on the reflectance which is used in the first step of *MetroSurv*'s shadow detection module. Herein, a hypothesis is proposed which states, if the current pixel belongs to a shadow region or not. The local feature is extended to the neighborhood of a pixel (spatial feature) in the second step in which the initial hypothesis is verified. Finally, a geometric feature is used during a postprocessing step, in which false positive detection results are reduced.

4.3.1 Hypothesis

The initial hypothesis is proposed for all pixels classified as foreground in the background subtraction module: If the (R, G, B) values of the current pixel are lower than the values of the corresponding pixel in the background image, this pixel is considered as a shadow pixel as an initial hypothesis.

4.3.2 Verification

Then, this initial hypothesis must be verified. The response of the camera's cth sensor is:

$$I_c = \sigma_c(\lambda)S(\lambda)E(\lambda) \tag{1}$$

where σ_c is the sensor's sensitivity, $E(\lambda)$ is the illuminant signal and $S(\lambda)$ is a function that relates the reflectance of a surface to that of a perfect Lambertian surface (Barnard (1999)). Now consider two adjacent points p and p'. At this point two different assumptions can be made:

1. Since p and p' are adjacent, they receive the same irradiance: $E(\lambda, p) = E(\lambda, p')$.
2. The two points p and p' lie on the same surface and have therefore the same reflectance: $S(\lambda, p) = S(\lambda, p')$.

Several shadow detection algorithms are based on these two assumptions. Assumption 1 forms the basis of the algorithms presented in Lo and Yang (2006) and Subramanya et al (2005) whereas Bhattacharyya (2004) relies on the other assumption. Our algorithm uses both assumptions.

Reflectance Ratio First, if assumption 1 holds, the ratio of the intensities can be simplified to:

$$\frac{I_c(p)}{I_c(p')} = \frac{\sigma_c(\lambda)S(\lambda,p)E(\lambda,p)}{\sigma_c(\lambda)S(\lambda,p')E(\lambda,p')} = \frac{S(\lambda,p)}{S(\lambda,p')} \tag{2}$$

The same applies for the corresponding pixels in the background model:

$$\frac{I_{c,BG}(p)}{I_{c,BG}(p')} = \frac{S_{BG}(\lambda,p)}{S_{BG}(\lambda,p')} \tag{3}$$

If p and p' are located in shadow regions, the surface in the current frame and in the background model is the same which means that $S(\lambda,p) = S_{BG}(\lambda,p)$ and $S(\lambda,p') = S_{BG}(\lambda,p')$. Now both ratios should be equal:

$$R_{FG}(p,p') = \frac{I_c(p)}{I_c(p')} = \frac{I_{c,BG}(p)}{I_{c,BG}(p')} = R_{BG}(p,p') \tag{4}$$

This property is called "color constancy between pixels" in Lo and Yang (2006) and "reflectance ratio" in Subramanya et al (2005).
The difference $D(p,p') = |R_{FG}(p,p') - R_{BG}(p,p')|$ between these ratios gives a hint, if this pixel could be a shadow pixel. Similar to Lo and Yang (2006), a window W around the pixel p is taken into account by summing up the differences. In addition to that, all three color channels are considered. The final score $Sc_{RR}(p)$ for pixel p is calculated as:

$$Sc_{RR}(p) = \sum_{c\in\{R,G,B\}} \sum_{p'\in W, p'\neq p} D_c(p,p') \tag{5}$$

If this score is small, the point is assumed to be a shadow pixel.
There are two problems: First, the assumption does not hold for sharp shadow edges, where the irradiance of the two points differs. Second, condition 4 can be fulfilled, when p and p' are not in shadow, but lie on a surface with a similar reflectance $S(\lambda,p) \approx S_{BG}(\lambda,p)$ and $S(\lambda,p') \approx S_{BG}(\lambda,p')$.

Irradiance Ratio Now assume that assumption 2 holds and consider again two adjacent points p and p'. The ratio of the intensities can be written as:

$$\frac{I_c(p)}{I_c(p')} = \frac{\sigma_c(\lambda)S(\lambda,p)E(\lambda,p)}{\sigma_c(\lambda)S(\lambda,p')E(\lambda,p')} = \frac{E(\lambda,p)}{E(\lambda,p')} \tag{6}$$

Let p and p' be shadowed pixels. In this case the irradiance is equal and the ratio given above is equal to 1:

$$\frac{I_c(p)}{I_c(p')} = 1 \tag{7}$$

Unfortunately, this is also true if p and p' are located in an illuminated area. Therefore, the condition given in equation 7 can just be used to retract a pixel previously classified as shadow. Similar to the reflectance ratio, the irradiance ratio is calculated for a window W around the pixel p and for all color channels. If the mean value is not close to 1, p and p' are not both located in a shadowed region and the shadow region is reclassified as foreground.

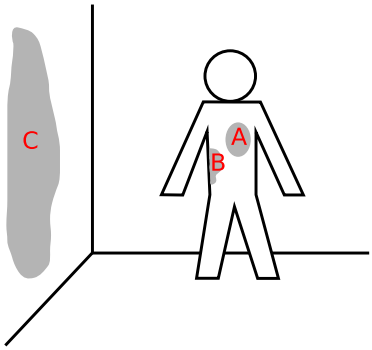

Fig. 2 Analysis of the shadow boundaries (Salvador (2004)). As region C has no adjacent foreground pixels, it is correctly classified as shadow whereas region A is completely surrounded by foreground pixels and is therefore reclassified as foreground. Region B is also reclassified as foreground as it is connected to to enough pixels classified as foreground pixels.

4.3.3 False Positive Reduction

After these two steps, there may still be some regions inside the moving objects which are misclassified as shadow regions. In a postprocessing step similar to that in Salvador (2004) and Bevilacqua (2003), these false positives are detected. First, all detected shadow pixels must be associated with a shadow region. For that purpose, a sequential labeling algorithm described in Nayar and Bolle (1996) is applied. Second, the pixels adjacent to a shadow region must be analyzed.
Let M_b^l be the number of background pixels adjacent to shadow region l and M_f^l the number of foreground pixels adjacent to shadow region l. If a certain fraction of all adjacent pixels belongs to the foreground, $\frac{M_f^l}{M_f^l + M_b^l} \geq mf_{thresh}$, then this shadow region is assumed to be misclassified in the previous step. Empirically, this threshold was set to $mf_{thresh} = 0.7$.
Figure 2 shows an example for this analysis: Shadow region C, which is not connected to the shadow casting object, has no adjacent foreground pixels and is therefore classified as shadow whereas region A lies inside a moving object and is reclassified as a foreground region. The condition for region B is fulfilled, so that it is also reclassified as foreground.

5 Tracking

The tracking module in *MetroSurv* is inspired by the works of Senior et al (2001), Senior (2002) and Cucchiara et al (2004). In each frame the motion detection module extracts all blobs which are called *visual objects* in *MetroSurv*. With the knowledge of the *tracks* already detected in the previous frame, the tracking system tries to find correspondences between these new visual objects and the tracks. The tracking procedure is as follows: In the first step of the tracking module the visual objects extracted from frame t are mapped to the tracks of frame $t-1$. Each object is represented by an appearance model. If a visual object could not unambiguously be mapped to a track (for example in case of occlusions), the appearance model is used to assign pixels to

tracks.
For mapping visual objects to existing tracks which are both characterized by a bounding box the *Bounding Box Distance* measure *bbDist* is used: Consider two bounding boxes A and B . If one of the two centroids lies inside the other bounding box, *bbDist* is zero. Otherwise *bbDist* is equal to the minimum Euclidean distance from the centroid C_A of bounding box A to the closest point P_B on B and the Euclidean distance from C_B to the closest point P_A on A. Besides a small bounding box distance, the areas of the blobs must be similar.
If a visual object is mapped to more than one existing track, the appearance models which are associated to the tracks are used to assign pixels of the visual object to one of the tracks. The appearance model consists of a color model and a probability mask similar to Senior et al (2001) and Senior (2002). The color model is initialized by copying all pixels belonging to the track's blob into the color model *cm*. The probability mask *pm* is initialized by setting these pixels to a predefined probability value. As the appearance of objects change during tracking, the appearance model has to be updated as follows:

$$\begin{aligned} cm_t(x,y) &= cm_{t-1}(x,y) \cdot (1-\alpha) + I_t(x,y) \cdot \alpha, \text{ if } I_t(x,y) \in Track \\ pm_t(x,y) &= pm_{t-1}(x,y) \cdot (1-\alpha) + \alpha, \text{ if } I_t(x,y) \in Track \end{aligned} \tag{8}$$

where α is a learning rate between zero and one. Pixels of the appearance model, which do not belong to this track, are updated according to:

$$\begin{aligned} cm_t(x,y) &= cm_{t-1}(x,y) \cdot (1-\alpha), \text{ if } I_t(x,y) \notin Track \\ pm_t(x,y) &= pm_{t-1}(x,y) \cdot (1-\alpha), \text{ if } I_t(x,y) \notin Track \end{aligned} \tag{9}$$

For assigning pixels to existing tracks template matching between the appearance model and the current frame inside a search region around the previous position of the track is performed. For template matching only those pixels of the color model are used which have a probability value in the probability mask greater than 0.5. Template matching is performed using normalized cross correlation.

6 Classification

In each frame the classification module of *MetroSurv* classifies each track into the classes Cl = {Human, Train, Unknown, Ghost}. Firstly, the trajectory of the object is analyzed for ghost detection. If the object has not moved for a while, it is assumed to be a ghost. After that, it is classified as a human, train or unknown object. Taking temporal consistency into account, a track Tr keeps track of its classification results, so that the probability $P_{Human}(Tr)$ can be calculated as follows:

$$P_{Human}(Tr) = \frac{k_{human}(Tr)}{k_{human}(Tr) + k_{train}(Tr) + k_{unknown}(Tr)} \tag{10}$$

where k_{Cl} stands for the number of classification results saying that the objects is an instance of class Cl. The probabilities $P_{Train}(Tr)$ and $P_{unknown}(Tr)$ are calculated accordingly.
Each track is characterized by the shape of its blob. For a blob several features are determined: area, perimeter, breadth, height and orientation. This features are used in

a simple rule-based classification step: Firstly, the classification system assumes that humans as well as trains stand on the platform or on the tracks, respectively. Thus, the user has to define the position of the platform and tracks by providing an image with this information to *MetroSurv*. If the maximum y-value of a blob is outside these regions, the blob is classified as an unknown object.
A train can be distinguished from humans by considering its height and its area. If the area or the height of the blob exceeds a certain threshold, the blob can definitely be classified as a train and $k_{train}(Tr)$ is incremented. If the blob's area is less than this threshold but greater than a second threshold, the track is classified as a human. Both thresholds have been determined empirically. Otherwise the extracted features are used to discriminate between humans and other objects. For that purpose, typical values for the different features have been measured beforehand. If most of the criteria are fulfilled, the object is assumed to be a human and the counter $k_{human}(Tr)$ is incremented. Otherwise, it is classified as an unknown object.

7 Event Detection

The event detection module in our system takes the results of the low-level vision modules as input. For each object detected in the scene the following information is available: centroid, bounding box, trajectory, velocity, moving direction, classification (human, train or unknown object). The module raises an alarm as output if one of seven predefined events takes place.
Ideally, the system should be flexible, so that it can be extended to detect further events. The knowledge representation should be very easy to understand, as domain experts should be able to formulate the events the system has to detect. It is very important for a surveillance system to analyze the results of the vision modules in real-time. Furthermore, knowledge about the scene context should be taken into account.

The event detection module is inspired by the works of Shet et al (2005) and Fernandez et al (2007). As a detailed description of the events that need to be detected is provided, it seems natural to choose a rule-based approach. Figure 3 shows an overview of the event detection module. The core consists of a rule-based expert system. In the following section the basics of expert systems are introduced.

7.1 Expert Systems

We have utilized the rule-based expert system CLIPS[3]. Such an expert system offers a simple syntax for writing rules and facts so that domain experts are able to understand and formulate rules on their own. Nevertheless, the language is very powerful and the expert system offers a ready-to-use inference mechanism which works in real-time. In contrast to the system VidMAP in Shet et al (2005) which employs Prolog, an expert system offers forward chaining and therefore works data-driven whereas Prolog works goal-driven. For that reason, VidMAP contains a reasoning thread that is invoked every 5 seconds to insert facts into the knowledge base and query the Prolog engine.

Figure 4 shows the general layout of an expert system with forward chaining. Such a system consists of a *working memory*, a *rule set* and an *inference engine*. The working

[3] CLIPS: http://www.ghg.net/clips/CLIPS.html

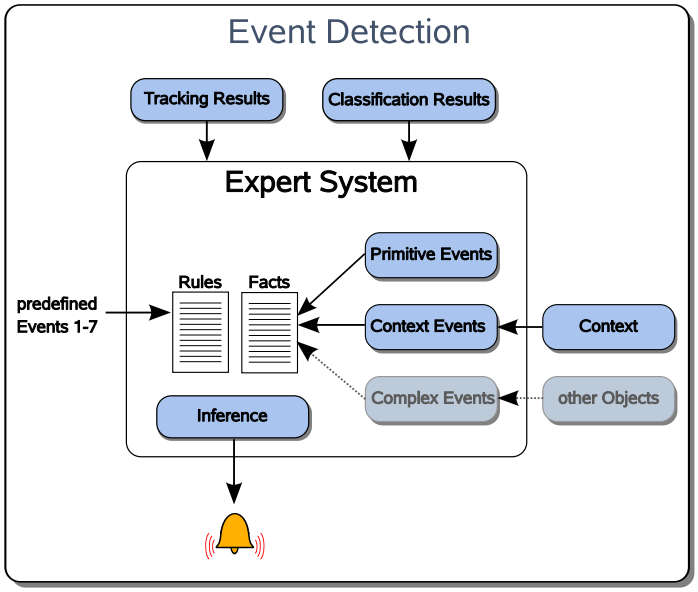

Fig. 3 Event detection. This module uses an expert system to analyze the activities taking place in the scene. For that purpose, tracking and classification results are analyzed and facts are inserted into the knowledge base. Firstly, tracks are considered independently to generate primitive events. Secondly, the context of the environment is taken into account and context events are generated. The last step in the activity analysis module (complex events) is skipped in this system, because no interactions between objects in the scene have to be analyzed.
An inference engine analyzes the facts in the knowledge base and the rules which model the seven predefined events that have to be detected. If such an event is detected, an alarm is raised.

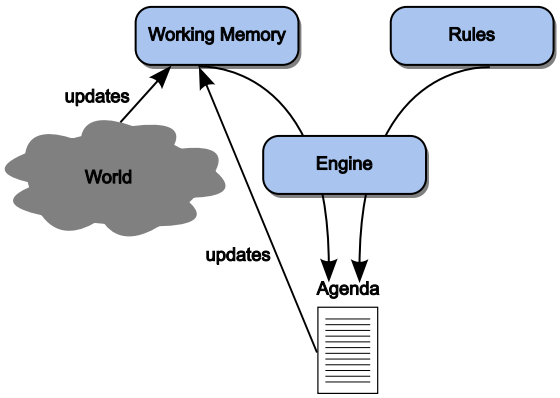

Fig. 4 Main components of an expert system with forward chaining, see Kazarov and Ryabov (1998).

memory contains all facts. A fact represents a piece of information and is the basic unit of data in an expert system, see CLIPS (2006). In CLIPS the structure of (non-ordered) facts can be defined using the `deftemplate` construct, for example

```
(deftemplate trackobject
        (slot label)
        (slot classification)
        (multislot centroid)
        ...
)
```

For example, a fact like `(trackobject (label 1)(classification human)(centroid 55 103))` can be added to the list of facts. These facts in the working memory represent the knowledge about the current state. New facts are added to the working memory if the current state changes. The working memory is also updated by rules. In CLIPS rules have the following form, see Jackson (1998):

```
(defrule <rule-name>
      <premise 1>
      ...
      <premise m>
      =>
      <action 1>
      ...
      <action n>
)
```

The left-hand side of a rule consists of a set of conditions. The right-hand side defines actions that are executed if the rule fires. Often, new facts are added to the working memory, existing facts are modified or retracted. The inference engine determines which rules are applicable and puts them onto the *agenda*. If multiple rules are applicable, the order of the rules is determined by considering a conflict resolution strategy. All applicable rules are fired according to the conflict resolution strategy until no applicable rule remains.

7.2 Knowledge Representation

Usually, surveillance systems, which are designed to detect certain events in video sequences, rely on knowledge. Three main types of knowledge can be distinguished:

1. *Scene context*: Surveillance systems operate in certain environments. Mostly, knowledge about this environment is used for inference. Knowledge about the geometry of the environment can be provided to the system in form of 3D models (see for example Vu et al (2002)).
2. *Current state*: The surveillance system has to represent the current state of the scene in a formal way. For example, it has to be aware of the humans' positions, their velocity and moving direction.
3. *Detected events*: As a surveillance system has the job of monitoring an area and raising an alarm, it has to know which events need to be detected.

Knowledge must be provided to the system in a formal way. Its representation should be easy to express and to understand for humans, as often domain experts define activities the system has to recognize. Of course the representation of knowledge often depends on the chosen activity analysis approach in the surveillance system. For example in Cupillard et al (2004) knowledge about the activities that should be detected in the video sequences are provided to the system through the states and the topology of the finite state automaton.
An example for a knowledge representation language in the context of video surveillance is ERL (Event Recognition Language), see Nevatia et al (2003): ERL aims at formalizing the third kind of knowledge. It provides a language for representing events that should be detected by the surveillance system. Such a representation of an event has the following form:

```
EventType EventName (ObjType ObjName,...)
  Event description;
```

The `EventType` is `PRIMITIVE` for simple events, which are performed by a single actor, `SINGLE_THREAD` for a combination of primitive events or `MULTI_THREAD` for a combination of multiple single-thread events which have some temporal, spatial or logical relations and even can involve multiple scene objects. `Event description` contains the representation of the event. The description of multi-thread events can include boolean expressions or temporal constraints. The parameters `ObjType ObjName,...` include all objects that are involved in this event. This representation language is independent of the event recognition approach. In Nevatia et al (2003), an event is translated to a tree structure which is passed to an inference mechanism. Nevatia et al (2003) proposes to recognize primitive events by using Bayesian Networks. Single-thread events are recognized by employing a variation of HMMs and multi-thread events can be detected by evaluating all sub-events and their relationships.
A similar knowledge representation language can be found in Vu et al (2003).
Our system represents knowledge as follows:

1. *Scene context*: Knowledge about the scene context is provided to the system by the definition of zones. Figure 5 shows the zones of interest. The green zones correspond to the platforms, the yellow zones represent the white lines and the red zone stands for the tracks. If three dimensional information about the scene context (tracks, ticket machines, seats) was available, this information could be represented by VRML models.
2. *Current state*: The current state of the scene is represented by the facts that are asserted to the knowledge base of the expert system.
3. *Detected events*: Predefined events our system should be able to detect are represented by the rules of the expert system.

7.3 Facts: Event Generation

Similar to Fernandez et al (2007), the objects detected by the vision modules are analyzed in multiple stages. Knowledge about the current state of the scene is then represented by asserting facts to the knowledge base. Firstly, a single object is considered independently: the results of the vision modules are analyzed and events are generated for this object. As Fernandez et al (2007) points out, there are three levels

(a) (b)

Fig. 5 (a) Zones of interest for the subway station in (b). The green zones correspond to the platforms, yellow zones represent the white lines and the red zone corresponds to the tracks.

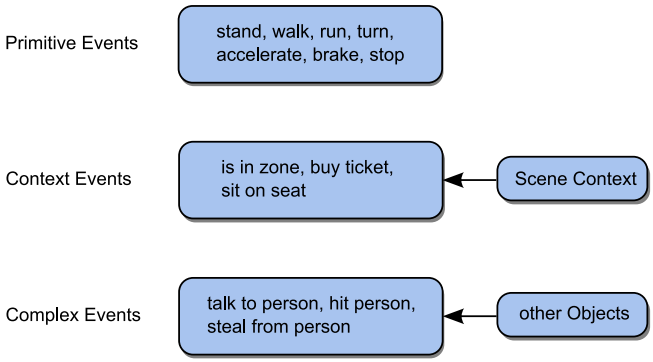

Fig. 6 Event Generation (see Fernandez et al (2007)). Events are grouped into three different types. Primitive events refer to events of a single independent object. When taking scene context into account, context events can be generated. For generating complex events other scene objects must be taken into account.

that can be used to analyze an object: When examining the trajectory, events like "run" or "turn around" can be generated. Considering the body posture of a person, it can be concluded, if a person sits down, stands or moves the head up and down. If the face of a person is analyzed, events like "concentrated" or "worried" could be generated.
Secondly, an object is analyzed with respect to its environment. Now events are generated for this object by taking knowledge about the scene context into account. In the last step interactions between objects in the scene are examined by considering all detected objects. Figure 6 shows some examples for these three types of events.

In the following the event generation in our system is described.

Primitive Events As stated before, the inputs of the analysis module are results of the tracking and the classification module. For each object present in the scene, there are six pieces of information which have to be analyzed in the first step. The results of this first step are *primitive events* (see Fernandez et al (2007)). The word "primitive" refers to the fact that each object is processed independently. The following *primitive events* are generated:

– *stopped*: An object has stopped, if its velocity in x-direction and y-direction are both below one pixel per frame.

- *moving_normal*: An object moves, if one of its velocity components is below three pixels per frame.
- *moving_fast*: An object moves fast, if one of its velocity components is greater than three pixels per frame.
- *appears*: An object has just appeared in the scene, if a corresponding fact is not yet in the knowledge base.
- *disappears*: An object has just disappeared, if it is no longer present in the scene.
- *direction_angle*: The moving direction is measured in degrees.

Context Events In the second stage information about the scene context is taken into account and *context events* are generated. Knowledge about the scene context is provided to the system by defining *zones of interest* (see Figure 5(a)). By considering this context information together with the current position of an object, our system can conclude in which zone the object is located in. Since in our case no calibration data for the cameras is available, it is not possible to estimate an object's position in three dimensional coordinates. Thus, the zone a human is located in is estimated by considering the highest y-value of a blob corresponding to a person. The context events that are generated for each object are:

- *inside_zone*: All zones an object is located in.
- *completely_inside_zone*: If an object is located in just one zone, it is completely inside this zone.
- *moving_to_zone*: By considering the current position and the moving direction, the zone an object is moving to can be computed. This movement could be used as a hint. For example, if a person stands on the platform, but moves to the tracks, a proximity warning is more probable. Note that in the current implementation this hint is not used.

Complex Events If a surveillance system has to detect interactions between objects, like two people talking, a third step in the analysis stage could be applied. After having analyzed an object independently (primitive events) and with respect to the context of the scene (context events), interactions between objects could be examined by considering all objects detected by the vision modules. In the case of this system no interactions between objects have to be detected, so this step is skipped and no complex events are generated.

7.4 Rules: Event Detection

In the following subsections, the detection of some events is explained. For each event there exists a rule in the rule set of the expert system that invokes a function that just logs the alarm. This function could be extended to ask a human operator to evaluate the situation or to cause a train to stop.

Proximity warning A proximity warning for object o is raised if it is classified as human and if it is completely inside the yellow zone (white line, see Figure 5) or if it is inside the green and the red zone.

```
(trackobject (label ?id)
   (classification unknown)
   (motion ~stopped)
   (completely-inside-zone tracks)
   (angle ?angle))
```

Listing 1 Properties of an object dropped on the tracks. The operator $\sim$ is a connective constraint. Such a constraint is satisfied, if the following constraint is not satisfied (see CLIPS (2006)). In this case the object must not have stopped. The angle of the moving direction must lie between $-35°$ and $-145°$.

Dropping objects on tracks This alarm is raised for an object with label `?id` (in CLIPS `?id` denotes a variable), if it has the properties defined in listing 1. The angle of the moving direction (`?angle`) must lie between $-35°$ and $-145°$.

Launching objects across the platforms This event is similar to the previous event. In contrast to that, the angle of the moving direction (`?angle`) must be greater than $-35°$ or less than $-145°$.

Person trapped by the door of a moving train This event is recognized if the bounding box of an object classified as train overlaps the white line (see listing 2), as the blobs of the train and the person merge into a single foreground region. Additionally, a proximity warning is raised, too.

```
(defrule alarm4
    ?fact <- (trackobject (label ?label)
                (bl ?minx ?maxy)
                (tr ?maxx ?miny)
                (inside-zone $?zones)
                (classification train))
    (test(member$ platform (create$ $?zones)))
     =>
     (alarm 4 ?label ?minx ?maxx ?miny ?maxy)
)
```

Listing 2 Complete rule for the event "Person trapped by the door of a moving train". bl stands for the bottom left corner of the bounding box and tl for its top right corner. If this rule is applicable, a function is called which logs this alarm together with the object's id and its bounding box.

Walking on rails A person is walking on the rails, if an object with label `?id` is detected, that has the properties defined in listing 3. The angle of the moving direction (`?angle`) must have a value greater than $35°$ and less than $145°$ or less than $-35°$ and greater than $-145°$.

Fall on the tracks This alarm is raised, if a person touches the red zone. See listing 4 for the complete rule.

```
(trackobject (label ?label)
    (completely-inside-zone tracks)
    (classification human)
    (angle ?angle))
```

Listing 3 Walking on rails.

```
(defrule alarm6
    ?fact <- (trackobject (label ?label)
                  (bl ?minx ?maxy)
                  (tr ?maxx ?miny)
                  (inside-zone $?zones)
                  (classification human)
                  (motion ~stopped))
    (test(member$ tracks (create$ $?zones)))
     (or (test(member$ line (create$ $?zones)))
     (test(member$ platform (create$ $?zones)))
     )
    =>
    (alarm 6 ?label ?minx ?maxx ?miny ?maxy)
)
```

Listing 4 Complete rule for the event "Fall on the track".

Crossing the rails This event is similar to the event "Walking on rails". The only difference is the constraint on the angle of the moving direction: Its value has to be between $-35°$ and $35°$ or it has to be greater than $145°$ or less than $-145°$.

8 Results

The foreground segmentation algorithm has been proven to give good results for some difficult situations in comparison to several other algorithms as shown in Krausz (2007). In particular, the results for the problem "Camouflage" are very satisfying. This problem type is very important for *MetroSurv*, since people visible in the video sequences often wear dark clothes that look similar to the background.
Figures 7 and 8 show the effectiveness of the high-level feedback: Figure 7 compares the result of the foreground segmentation when the detection of ghosting is enabled and when it is not activated. In Figure 8(b), a train is detected and the speed of update is thus adapted.

Figure 9 presents some results of the shadow detection algorithm proposed in section 4.3. Green pixels are classified as shadow regions. Red Pixels indicate that these pixels were classified as shadow pixels in the first two steps of the algorithm (hypothesis and verification), but were reclassified as foreground pixels in the false positive reduction step.

The tracking module of *MetroSurv* is evaluated by considering a video sequence showing four people. The results of the evaluation are satisfying. Two people could be tracked correctly in 93.4% or 99.6% of the time they are visible, respectively, although they are involved in a number of occlusion situations (60% resp. 50.8%).
The classification module is also evaluated using the aforementioned video sequence.

(a) (b)

Fig. 7 Foreground segmentation results if the detection of ghosting is enabled (Figure (b)). If a ghost is detected, the update velocity α is increase in this region.

(a) (b)

Fig. 8 Foreground segmentation results if high-level feedback is used (Figure (b)): When a train is detected, the speed of update is adapted.

All but one person are classified correctly as humans in more than 98% of the time they are visible. Although a simple rule-based approach is used, the classification results are very satisfiying and efficient.

The detected events of our system have been compared to the ground truth annotation of the video sequences. Most of the predefined events could be detected with a sufficient detection rate. For example, the event "Walking on Rails" is detected with a detection rate of 83%, whereas "Person trapped by the door of a moving train" is always detected. As the event detection module analyzes the tracking and classification results, it highly relies on the accuracy of these results. For example, the detection rate of proximity warnings is 67% as the low-level vision modules does not work properly in some cases. Consequently, it is important to note that the most important criterion for the quality of an automatic surveillance system is the accuracy of the low-level modules. If the event detection module has to analyze inexact or erroneous data, its detection rate will be low. In order to improve the system's robustness, fuzzy logic could be integrated into the event detection module by using a fuzzy expert system (see Liao (2005) for an overview).

Our system has a sufficient performance with about 12 fps when executed in parallel on four processors. Of course, the performance depends on the complexity of the scene. In videos sequences with a single person, the performance goes up to 30 fps when being executed on four processors in parallel. Figure 10 depicts the relative processing time of the analysis module for two different video sequences where in one video sequence only a single person is visible. In the second video sequence, a group of people is present in

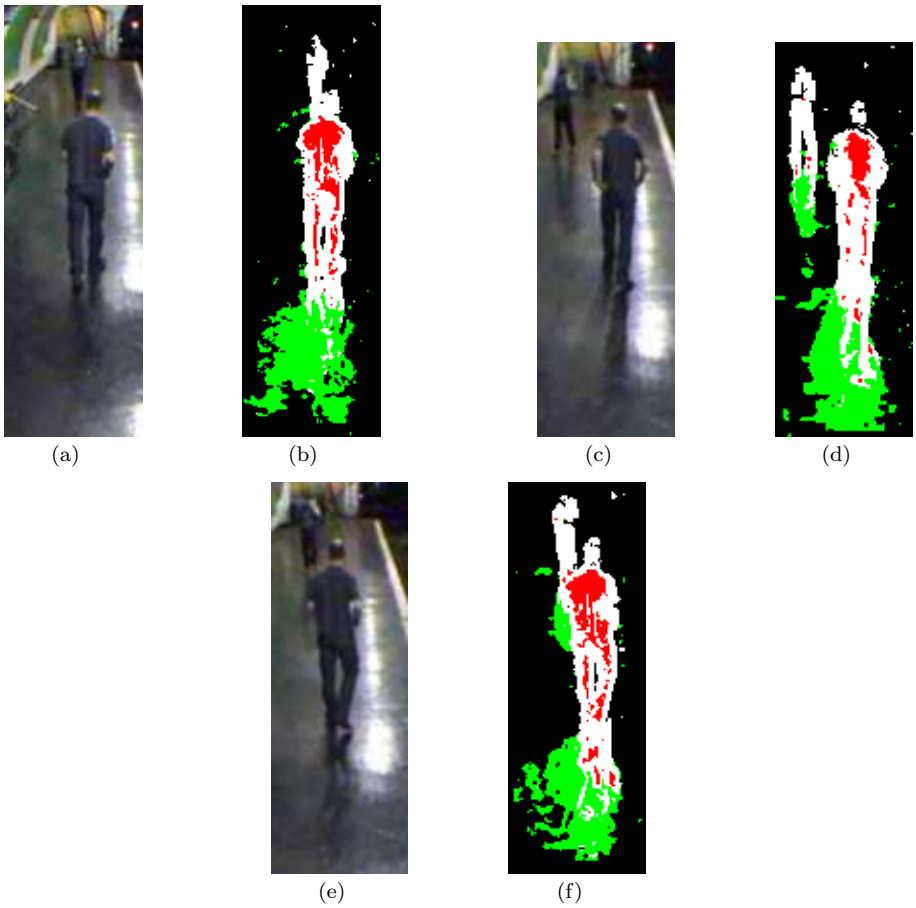

Fig. 9 Results of the shadow detection algorithm. Green regions are classified as shadows. Red pixels were classified as shadow pixels, but are then reclassified in the false positive reduction step.

the scene. Due to several occlusion situations most of the time is spent on tracking. In both video sequences less than 5% of the processing time is spent on the event detection, since the event detection module is very fast due to the inference engine of CLIPS.

The event detection module is flexible and can easily be extended to recognize further events, since it makes use of the inference engine CLIPS which offers a simple syntax. Consequently, it is even possible for domain experts to formulate events. Furthermore, it is even possible to apply the event detection module in other application domains. For that purpose, rules have to be adapted to incorporate knowledge about the specific application domain and to specify events that have to be detected.

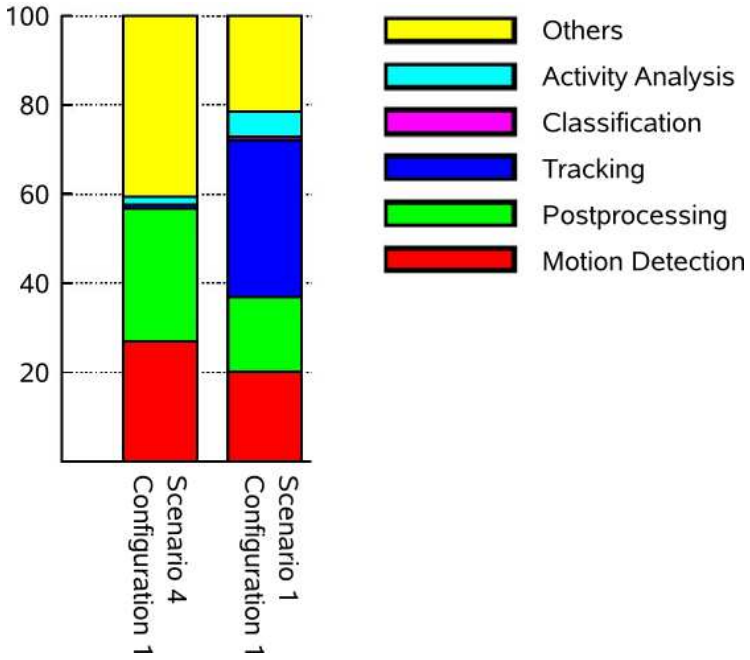

Fig. 10 Execution time of the modules in percent for two different video sequences. In the first video sequence (Scene 4 Configuration 1) only a sinlge person is visible whereas in the second video sequence a group of people is present in the scene. For that reason, most of the time is spent on tracking in the latter video sequence. The event detection module takes less than 5% of the execution time.

9 Conclusion

An automatic video surveillance system for the detection of dangerous events in a subway station has been presented and its modules have been described and evaluated: The foreground segmentation module uses a variable number of Gaussians to model the background. In order to improve its results feedback of high-level modules is utilized. Furthermore, a novel algorithm for the detection of shadow regions has been proposed and shown to be very effective for reclassifying shadow regions as background. The tracking module which uses ideas proposed by Senior (2002) is able to track objects even if they are partially occluded. Each detected object is successfully and efficiently classified by a simple rule-based approach. Finally, the described event detection module uses an approach which distinguishes it from most other works in the field of surveillance systems. It provides a simple syntax for rules, so that even domain experts are capable of describing events that should be detected. Furthermore, the expert system CLIPS offers a fast inference engine.

References

Barnard K (1999) Practical colour constancy. PhD thesis, Simon Fraser University
Barth A, Herpers R (2005) Robust head detection and tracking in cluttered workshop environments using GMM. In: Annual meeting of the German Association for Pattern Recognition

Bevilacqua A (2003) Effective shadow detection in traffic monitoring applications. In: Journal of WSCG, vol 11, pp 57–64
Bhattacharyya J (2004) Detecting and removing specularities and shadows in images. Master's thesis, McGill University, Montreal
Bremond F, Medioni G (1998) Scenario recognition in airborne video imagery. In: Proc. DARPA Image Understanding Workshop
CLIPS (2006) CLIPS: Reference Manual, Volume 1 Basic Programming Guide
Collins R, Lipton A, Kanade T (1999) A system for video surveillance and monitoring. In: American Nuclear Society 8th Internal Topical Meeting on Robotics and Remote Systems
Comaniciu D, Meer P (1999) Mean shift analysis and applications. In: Proc. Int. Conf. on Computer Vision, vol 2
CREDS (2005) Call for real-time event detection solutions (creds) for enhanced security and safety in public transportation. http://www-dsp.elet.polimi.it/avss2005/CREDS.pdf
Cucchiara R, Grana C, Piccardi M, Prati A (2001) Detecting objects, shadows and ghosts in video streams by exploiting color and motion information. In: Proc. Int. Conf. on Image Analysis and Processing
Cucchiara R, Grana C, Tardini G, Vezzani R (2004) Probabilistic people tracking for occlusion handling. In: Proc. Int. Conf. on Pattern Recognition
Cupillard F, Avanzi A, Bremond F, Thonnat M (2004) Video understanding for metro surveillance. In: Proc. Int. Conf. on Networking, Sensing and Control
Cutler R, Davis LS (2000) Robust real-time periodic motion detection, analysis, and applications. IEEE Trans on Pattern Analysis and Machine Intelligence 22(8):781–796
Fernandez C, Baiget P, Roca FX, Gonzalez J (2007) Semantic annotation of complex human scenes for multimedia surveillance. In: Int. Conf. on Advances in Artificial Intelligence
Frahm JM, Evers-Senne JF, Koch: R (2003) Distributed interaction processing and visualization of 3d scenes in realtime. In: Int. Symp. on Image and Signal Processing and Analysis
Ghanem N, DeMenthon D, Doermann D, Davis L (2004) Representation and recognition of events in surveillance video using petri nets. In: Conf. on Computer Vision and Pattern Recognition Workshop
Giachetti A, Campani M, Torre V (1998) The use of optical flow for road navigation. IEEE Trans on Robotics and Automation 14(1):34–48
Gryn JM, Wildes RP, Tsotsos JK (2009) Detecting motion patterns via direction maps with application to surveillance. Computer Vision and Image Understanding 113(2):291 – 307
Haritaoglu I, Harwood D, Davis LS (1998) W4: Who? when? where? what? a real time system for detecting and tracking people. In: Proc. Int. Conf. on Face & Gesture Recognition
Hongeng S, Bremond F, Nevatia R (2000) Bayesian framework for video surveillance application. In: Int. Conf. on Pattern Recognition
Horprasert T, Harwood D, Davis L (1999) A statistical approach for real-time robust background subtraction and shadow detection. In: Int. Conference on Computer Vision '99 FRAME-RATE Workshop
Hsieh JW, Yu SH, Chen YS, Hu WF (2004) A shadow elimination method for vehicle analysis. In: Int. Conf. on Pattern Recognition
Hu W, Tan T, Wang L, Maybank S (2004) A survey on visual surveillance of object motion and behaviors. IEEE Trans on Systems, Man and Cybernetics 34(3):334–352
Isard M, Blake A (1998) Condensation – conditional density propagation for visual tracking. Int J Computer Vision 29:5–28
Ivanov YA, Bobick AF (2000) Recognition of visual activities and interactions by stochastic parsing. IEEE Trans on Pattern Analysis and Machine Intelligence 22(8):852–872
Jackson P (1998) Introduction to Expert Systems (3rd Edition). Addison Wesley
Kazarov A, Ryabov Y (1998) Clips expert system tool: a candidate for the diagnostic system engine
Krausz B (2007) Detection of Events in Video Surveillance. Master's thesis, FH Bonn-Rhein-Sieg, Sankt Augustin
Liao SH (2005) Expert system methodologies and applications–a decade review from 1995 to 2004. Expert Systems with Applications 28(1):93 – 103
Lipton A, Fujiyoshi H, Patil R (1998) Moving target classification and tracking from real-time video. In: Int. Workshop on Applications of Computer Vision
Lo KH, Yang MT (2006) Shadow detection by integrating multiple features. In: Int. Conf. on Pattern Recognition
McCahill M, Norris C (2002) CCTV in london

Muncaster J, Ma Y (2007) Hierarchical model-based activity recognition with automatic low-level state discovery. Journal of Multimedia 2:66–76
Nayar S, Bolle R (1996) Reflectance based object recognition. Int J Computer Vision 17(3):219–240
Nevatia R, Zhao T, Hongeng S (2003) Hierarchical language-based representation of events in video streams. In: Workshop Event Mining
Salvador E (2004) Shadow segmentation and tracking in real-world conditions. PhD thesis, Lausanne
Senior A (2002) Tracking people with probabilistic appearance models. In: Int. Workshop Performance Evaluation of Tracking and Surveillance Systems
Senior A, Hampapur A, li Tian Y, Brown L, Pankanti S, Bolle RM (2001) Appearance models for occlusion handling. In: Int. Workshop on Performance Evaluation of Tracking and Surveillance Systems
Shet VD, Harwood D, Davis LS (2005) Vidmap: Video monitoring of activity with prolog. In: IEEE Conf. on Advanced Video and Signal Based Surveillance
Shet VD, Harwood D, Davis LS (2006) Multivalued default logic for identity maintenance in visual surveillance. In: Europ. Conf. on Computer Vision
Starner T, Pentland A (1995) Real-time american sign language recognition from video using hidden markov models. Int Symp on Computer Vision
Stauffer C, Grimson W (1999) Adaptive background mixture models for real-time tracking. In: IEEE Conf. on Computer Vision and Pattern Recognition
Subramanya SR, Bunyak F, Ersoy I (2005) Shadow detection by combined photometric invariants for improved foreground segmentation. In: IEEE Workshop on Application of Computer Vision
Vu VT, Bremond F, Thonnat M (2002) Human behaviour visualisation and simulation for automatic video understanding. In: Int. Conf. on Computer Graphics, Visualization and Computer Vision
Vu VT, Bremond F, Thonnat M (2003) Automatic video interpretation: A novel algorithm for temporal scenario recognition. In: Int. Joint Conf. on Artificial Intelligence
Wilson AD, Bobick AF (1998) Recognition and interpretation of parametric gesture. In: Int. Conf. on Computer Vision
Wren CR, Azarbayejani A, Darrell T, Pentland A (1997) Pfinder: Real-time tracking of the human body. IEEE Trans on Pattern Analysis and Machine Intelligence 19(7):780–785
Zang Q, Klette R (2003) Object classification and tracking in video surveillance. In: Proc. Computer Analysis of Images and Patterns
Zivkovic Z, van der Heijden F (2004) Recursive unsupervised learning of finite mixture models. IEEE Trans on Pattern Analysis and Machine Intelligence 26(5)
Zivkovic Z, van der Heijden F (2006) Efficient adaptive density estimation per image pixel for the task of background subtraction. Pattern Recognition Letters 27(7):773–780