Comparison of target detection algorithms using adaptive background models

D.Hall[1], J. Nascimento[2], P. Ribeiro[2], E. Andrade[3], P. Moreno[2], S. Pesnel[1], T. List[3], R. Emonet[1], R.B. Fisher[3], J. Santos Victor[2] and J.L. Crowley[1]

INRIA Rhône-Alpes[1] ISR[2] University of Edinburgh[3]

Abstract

This article compares the performance of target detectors based on adaptive background differencing algorithms on public benchmark data. Several state of the art methods are described together with their parameterization. The performance is evaluated using computation time, recall and precision with respect to annotated ground truth. A surprising result is that the method with the most complex background model is outperformed by an efficient alternative implementation and a very simple background model combined with a Kalman filter.

1. Introduction

The video surveillance domain has a great demand for real time image processing systems that operate reliably 24h a day, 7 days a week. Efficient and reliable algorithms are required for this kind of system. Adaptive background differencing techniques become a widely used solution, since they can incorporate illumination changes as they occur in outdoor scenes during the day. In this article, we compare six state of the art adaptive background differencing techniques. The detectors are evaluated on the same public benchmark dataset which allows a fair and extensive comparison. This article gives insight into performance, computation time and usability of these methods.

The first method is a basic background subtraction algorithm *(BBS)*. This is the simplest algorithm and it provides a lower benchmark for the other algorithms which are more complex but based on the same principle.

The second algorithm is denoted as *W4* and operates on gray scale images. Three parameters are learned for each pixel to model the background: minimum intensity, maximum intensity and maximum absolute difference in consecutive frames. This algorithm incorporates the noise variations into the background model.

The third method is used in *Pfinder* [12] denoted here as *SGM* (Single Gaussian Model). This method assumes that each pixel is a realization of a random variable with a Gaussian distribution. The first and second order statistics of this distribution are independently estimated for each pixel.

The fourth method *(MGM)* is an adaptive mixture of multiple Gaussians as proposed by Stauffer and Grimson in [11]. Every pixel of the background is modeled using a mixture of Gaussians. The weights of the mixture and the parameters of the Gaussians are adapted with respect to the current frames. This method has the advantage that multimodal backgrounds (such as moving trees) can be modeled. Among the tested techniques, this is the one with the most complex background model.

The fifth approach *(LOTS)* proposed by Boult in [5] is an efficient method designed for military applications that presumes a two background model. In addition, the approach uses high and low per-pixel thresholds. The method adapts the background by incorporating the current image with a small weight. At the end of each cycle, pixels are classified as false detection, missed detection and correct detection. The original point of this algorithm is that the per-pixel thresholds are updated as a function of the classification.

The last approach *(Track)* combines the BBS method with a Kalman filter to predict search regions for each target from frame to frame and provide robust tracking. In order to meet real time constraints, the detection of new targets is restricted to entry regions.

This article compares the performance of target detection approaches. The experiments show that the method with the most complex background model is outperformed by simpler methods both in terms of detection rate and computation speed.

The article is organized as follows. Sections 2 to 7 describe the technical details of the approaches. Section 8 describes the database, the evaluation metrics and the experimental results. We finish with conclusion and an outlook.

2. Basic Background Subtraction

This method detects targets by computing the difference between the current frame and a background image for each color channel RGB. A thresholding operation is performed to classify each pixel as foreground region if

$$|I^t(\phi) - \boldsymbol{\mu}^t(\phi)| > T, \tag{1}$$

where $I^t(\phi)$ is a 3-dimensional vector representing the intensity values of the three color channels at image position ϕ. $\boldsymbol{\mu}^t(\phi)$ is the mean color (background) of the pixel, T is a constant. The operation in (1) performed for all image pixels ϕ.

Segmentation of objects from the background can be achieved by connected component analysis (e.g., using 8 - connectivity criterion). This step is performed after morphological filtering with a 3x3 mask (dilation and erosion) that eliminates isolated pixels. The same connected component analysis is performed in the methods *W4* and *SGM* (Section 3 and 4).

To take into account slow illumination changes which is necessary to ensure longterm tracking, the background image is subsequently updated by

$$B_i^{t+1}(\phi) = \alpha I^t(\phi) + (1-\alpha)B_i^t(\phi) \quad (2)$$

with the learning rate α. In the experiments we use $\alpha = 0.15$ and threshold $T = 0.2$ since this was found to be the a good value in a previous experiment [4]. These parameters stay constant over all sequences of the experiments.

3. W4 method

This algorithm was proposed by Haritaoglu in [8]. The background scene is modelled by representing each pixel by three values; minimum intensity (Min), maximum intensity (Max), and the maximum intensity difference between consecutive frames (D) during the training period. These values are estimated over several frames and are periodically updated for those areas which do not contain target regions. In the experiments, 100 target free images are selected for parameter learning. No parameters need to set by hand.

Foreground objects are computed in four steps: $i)$ thresholding, $ii)$ region based noise cleaning, $iii)$ morphological filtering and $iv)$ object detection. Each pixel is classified as background or foreground using following equation. Giving the values of Min, Max and D, a pixel ϕ in an image I is considered as foreground pixel if

$$|\text{Min}(\phi) - I(\phi)| > D(\phi) \text{ or } |\text{Max}(\phi) - I(\phi)| > D(\phi) \quad (3)$$

The resulting thresholded image usually contains a significant amount of noise. Then a region based noise cleaning algorithm is applied that first applies an erosion operation and then a connected component analysis that allows to remove regions with less than 50 pixels. The result is a set of bounding boxes that contain the targets. The morphological operations dilation and erosion are now applied to the foreground pixels that are inside the bounding boxes. The final bounding boxes are computed and returned as targets.

4. Single Gaussian Model

In this section we describe the Single Gaussian Model algorithm *(SGM)* proposed by Wren in [12]. In this method, the intensity and color of each pixel is represented by a vector $[Y, U, V]^T$. We assume only slow scene changes. The mean $\boldsymbol{\mu}(\phi)$ and covariance $U(\phi)$ of each pixel ϕ can be recursively updated as follows

$$\boldsymbol{\mu}^t(\phi) = (1-\alpha)\boldsymbol{\mu}^{t-1}(\phi) + \alpha I^t(\phi), \quad (4)$$

$$U^t(\phi) = (1-\alpha)U^{t-1}(\phi) + \alpha\nu(\phi)\nu(\phi)^T \quad (5)$$

where $I(\phi)$ is the pixel of the current frame in YUV color space, α is the learning rate and $\nu(\phi) = (I^t(\phi) - \boldsymbol{\mu}^t(\phi))$.

After background adaptation, we compute for all image positions ϕ the log likelihood $l(\phi)$ of the difference $\nu(\phi)$ between current image and background. This value gives rise to a classification of individual pixels as background or foreground

$$l(\phi) = -\frac{1}{2}^T \nu(\phi)^T(U^{-1})^t\nu(\phi) - \frac{1}{2}\ln|U^t| - \frac{3}{2}\ln(2\pi) \quad (6)$$

A pixel ϕ is classified as foreground if $l(\phi) < T$ else it is background. Then SGM detects targets by clustering foreground pixels into blobs by a connected component analysis. In the experiments we use $\alpha = 0.005$ and $T = -300$.

5. Multiple Gaussian Model

In recent years time-adaptive per pixel mixture of Gaussian background models have been a popular choice for modelling complex and time varying backgrounds [9]. In this work we have implemented the original version of the Adaptive Mixture of multiple Gaussians background model *(MGM)* for motion tracking described in [11].

5.1. Algorithm

In this algorithm the pixel process is considered a time series of vectors for colour images. The history of a particular pixel ϕ is given by:

$$\{X_1, ...X_t\} = \{\mathbf{I}(\phi, i) : 1 \leq i \leq t\} \quad (7)$$

where I is the image sequence. The algorithm models the recent history of each pixel as a mixture of K Gaussian distributions. Thus the probability of observing the current pixel values is:

$$P(X_t) = \Sigma_{i=1}^K \omega_{i,t} * \eta(X_t, \mu_{i,t}, U_{i,t}) \quad (8)$$

where K is the number of distributions, $w_{i,t}$ is the weight estimate of the ith Gaussian in the mixture at time t, $\mu_{i,t}$ and $U_{i,t}$ are the mean value and covariance matrix of the ith Gaussian at time t, and η is the Gaussian probability density function.

$$\eta(X_t, \mu, U) = \frac{1}{(2\pi)^{\frac{n}{2}}|U|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t-\mu)^T U^{-1}(X_t-\mu)} \quad (9)$$

For computational simplicity the covariance matrix is assumed to be of the form $U_{k,t} = \sigma_k^2 \mathbf{I}$ avoiding a costly matrix inversion at the expense of some accuracy. The algorithm assumes that red, green and blue channels are independent and that the pixel process is non-stationary. These assumptions result in an on-line K-means approximation algorithm for the mixture model. In the on-line approximation every new pixel X_t is checked against the K existing Gaussian distribution. A match is found if the pixel value is within $L = 2.5$ standard deviations of a distribution. This is effectively a per pixel per distribution threshold and can be used to model shadowed and lighted regions on the scene. If the current pixel value matches none of the distributions the least probable distribution is updated with the current pixel values, a high variance and low prior weight. The prior weights of the K distributions are updated at time t according to:

$$\omega_{k,t} = (1 - \alpha)\omega_{k,t-1} + \alpha(M_{k,t}) \tag{10}$$

where α is the learning rate and $M_{k,t}$ is 1 for the model which matched the pixel and 0 for the remaining models. After this approximation the weights are renormalised. The changing rate in the model is defined by $1/\alpha$. The parameters μ and σ for the unmatched distributions remain the same. The parameters for the matching distribution are updated as follows:

$$\mu_t = (1 - \rho)\mu_{t-1} + \rho X_t \tag{11}$$

$$\sigma_t^2 = (1 - \rho)\sigma_{t-1}^2 + \rho(X_t - \mu)^T(X_t - \mu) \tag{12}$$

$$\rho = \alpha\eta(X_t|\mu_k, \sigma_k) \tag{13}$$

For change detection an heuristic searches for the learnt distributions which have more supporting evidence. The Gaussians are ordered based on the ratio of ω/σ. This increases as the Gaussian's weight increases and its variance decreases. The first B distributions accounting for a proportion T of the observed data are defined as background.

$$B = argmin_b \left(\Sigma_{k=1}^b \omega_k > T\right) \tag{14}$$

In the current implementation the original algorithm [11] is modified to updated the background model only for pixels detected as background in the previous frame. This decreases the absorption rate of stationary objects (dropped bags, immobile people) into the background model. After foreground detection the pixels are morphological filtered (noise reduction) and labeled. Connected components smaller than T_c pixels are discarded. Other background update strategies such as the ones described in [9] use feedback of higher level modules, whereas here this small modification provides an open loop low level feedback for the updates.

5.2. Parameterization for the experiments

The model is initially trained with 13 training sequences. No modification in the update rules are used for training. The algorithm parameters are set to $K = 5$ Gaussians, learning rate $\alpha = 0.002$ and $L = 2.5$ standard deviations to look for matching Gaussians. The detection performance is tested using the 14 test sequences.

For testing the initial 20 frames of each sequence suffer a full adaptation, after that, only pixels classified as background have their distributions updated. For detection the additional algorithm parameters are set to $T = 0.97$ corresponding to the percentage of pixels observed accounted for by the most stable distributions (background). This results in multi-modal distribution for the background. $L = 4.5$ to allow for larger deviations from the original background model. T_c = 25 pixels to remove small (noisy) connected components. This non-optimized version can process 2.8 frames per second of size 384×288 pixels on a P4 3.2 GHz.

6. LOTS

The target detector proposed in [5] operates on gray scale images. It uses two background images and two per-pixel thresholds. The two backgrounds allow to model periodic changes of the background such as moving trees. The per-pixel threshold image can treat each pixel differently, allowing the detector to be robust to localized noise in low-size image regions. The per-pixel threshold evolves based on a pixel label provided by a Quasi Connected Components analysis (QCC). This is a light version of the traditional connected component analysis that is also used to provide the target's bounding boxes.

6.1. Algorithm

The steps of the algorithm can be summarized as:

1. **Background and threshold initialization**. Set the background models B_1, B_2, and the threshold values T_L (low threshold), T_H (low threshold). The values in B_1 and B_2 are the lower and higher "non-target" pixel values in the scene, considering some temporal window. The per-pixel threshold T_L, is then initialized to be above the difference between the two backgrounds:

$$T_L(\phi) = |B_1(\phi) - B_2(\phi)| + \mathcal{U}(\phi) \tag{15}$$

where $\mathcal{U}$ represents noise with an uniform distribution in $[1, 10]$, and ϕ an image pixel. A higher threshold T_H is computed by:

$$T_H(\phi) = T_L(\phi) + \mathcal{V} \tag{16}$$

where $\mathcal{V}$ is the sensitivity of the algorithm.

2. **Detection and Labeling** First we create D, which contains the minimum of the differences between the new image I, and the backgrounds B_1 and B_2:

$$D(\phi) = \min_j |I(\phi) - B_j(\phi)|, j = 1, 2. \quad (17)$$

D is then compared with the threshold images. Two binary images D_L and D_H are created. The active pixels of D_L and D_H are the pixels of D that are higher than the thresholds T_L and T_H respectively.

QCC computes for each thresholded image D_L and D_H images Ds_L and Ds_H with 16 times smaller resolution. Each element in these reduced images, Ds_L and Ds_H, has the number of active pixels in a 4x4 block of D_L and D_H respectively. Both images are then merged into a single image that labels pixels as detected, not-detected and false detected. This process labels every pixel, and also deals with targets that are not completely connected, considering them as only one region.

A 4-neighbor connected components is then applied to this image and, the regions with less than $\mathcal{A}$ pixels are eliminated. The remaining regions are considered as detected. The *detected* pixels are the ones from D_L that correspond to detected regions. The *false detection* are the active pixels in D_L, but do not correspond to detected regions, and the *not detected* pixels are the inactive pixels in D_L.

3. **Backgrounds and threshold adaptation**. The backgrounds are updated as follows:

$$B_i^{t+1}(\phi) = \begin{cases} (1-\alpha')B_i^t(\phi) + \alpha' I^t(\phi) & \phi \in T^t \\ (1-\alpha)B_i^t(\phi) + \alpha I^t(\phi) & \phi \in N^t \end{cases} \quad (18)$$

where $\phi \in T^t$ represents a pixel that is labelled as *detected* and $\phi \in N^t$ a false or not detected pixel. Generally α' is smaller than α and only the background corresponding to the smaller difference D is updated. Using the pixel labels, thresholds are updated as follows:

$$T_L^{t+1}(\phi) = \begin{cases} T_L^t(\phi) + 10 & \phi \in \textit{false detection} \\ T_L^t(\phi) - 1 & \phi \in \textit{not detected} \\ T_L^t(\phi) & \phi \in \textit{detected} \end{cases} \quad (19)$$

This adaptation procedure decreases slowly the threshold for *not detected* pixels. This is done until their label changes. If they become *detected*, that is considered real targets, the threshold is maintained constant. If they become *false detections*, due to noisy pixels or if the threshold is too low, it is increased. This way the system have a constant rate of false detected pixels that can be chosen changing the increasing and decreasing steps.

6.2. Parameterization for the experiments

In the first step of the algorithm presented in Section 6, done only once, it is assumed that during a period of time there are no targets in the image. In this ideal scenario the two backgrounds are computed easily. The image sequences used in this work do not have target-free images, so another approach was used. B_1 is initialised as the mean of K consecutive frames. $B_2 = B_1 + u$ with u additive noise of $\mathcal{N}(\mu = 10, \sigma = 20)$. The thresholds are initialized as follows: (i) initialize T_L randomly, $T_L = \mathcal{N}(\mu = 10, \sigma = 20)$, and (ii) run the sequence from the end to the beginning and adapt the threshold. Then the resulting threshold was used to start testing the sequence.

After the initialization, the detection and labeling step is performed every frame. The adaptation step is performed every N frames, where N is related to the background adaptation constant (see [5] for details).

In the experiments we use the parameters as proposed by Boult $\alpha = 0.000306$ and $\alpha' = \frac{\alpha}{4}$ [5]. The sensitivity, $\mathcal{V} = 40$, is chosen from Receiver Operating Characteristics [2] and the minimum area, $\mathcal{A} = 100$ pixels, from the working scenario. The system runs at more than 130Hz on a 1.6GHz processor using images of 384x288 pixels.

7. Real-time tracking system

The tracking system is composed of a central supervisor that calls subsequently the video demon, the robust tracking module and the target detection module (Figure 1). The supervisor manages the data flow between the modules. The detection module detects new targets that are added to the target list. The tracking module provides robust tracking of the current targets using a Kalman filter. The system can monitor 5 entry regions and track robustly up to 8 targets in images of 384×288 pixels at 30Hz on a 2 GHz processor.

This architecture allows the integration and fusion of several detection modules. In order to meet the real time constraints, we use a single detection module based on adaptive background differencing using manually defined detection regions [3]. Robust tracking is achieved by a first order Kalman filter that propagates the target positions and extents in time and updates them by measurements from the detection module. A target is deleted only when the detected pixels in the region do not exceed the detection threshold in 10 subsequent frames. This reduces the number of target losses within a track.

The robust tracking system returns events in form of vectors composed of centroid and width and height of the target

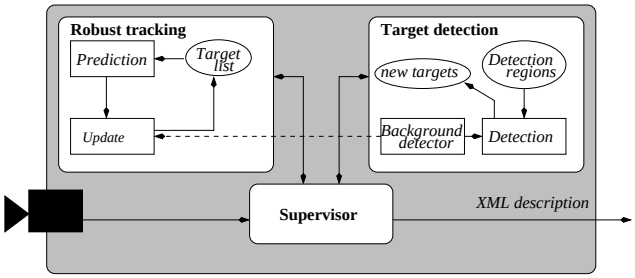

Figure 1: Architecture of the tracking and detection system controlled by a supervisor.

regions $\vec{y}(t_i) = (x_c, y_c, w, h)^T$. The tracking system depends on a number of parameters such as detection threshold d_c (minimum size of targets), noise threshold n_c (pixel energy below this threshold is considered as noise) and parameters that control split and merge of targets (these parameters determine how close targets need to be for merging or splitting).

7.1. Adaptive background differencing

We use a simple and efficient algorithm of adaptive background differencing as proposed by the authors [3]. The background is modeled by a single RGB background image.

A difference energy image I_d is computed from the current image $I = (I_{red}, I_{green}, I_{blue})$ and a single background image $B = (B_{red}, B_{green}, B_{blue})$ for all image coordinates ϕ as follows:

$$I_d(\phi) = \frac{1}{3}(|I_{red}(\phi) - B_{red}(\phi)| + |I_{green}(\phi) - B_{green}(\phi)| + |I_{blue}(\phi) - B_{blue}(\phi)|) \quad (20)$$

A binary image I_b is created by thresholding I_d with the noise threshold n_c.

$$I_b(\phi) = \begin{cases} 1, if I_d(\phi) > n_c \\ 0, else \end{cases} \quad (21)$$

For all regions of interest (entry regions and regions that are likely to contain targets), we compute

$$card(ROI) = \sum_{\phi \in ROI} I_b(\phi) \quad (22)$$

A target is detected, when $card(ROI) > d_c$ with d_c detection threshold. The first moment of image coordinates of the pixels with $I_b > 0$ give the center of gravity of the target and the second moment (covariance) gives the extent. From the covariance we compute the width and height of the bounding box and the global orientation of the target.

The background image B is updated every kth frame using a weighted averaging technique that updates the background image by incorporating the current image with a small weight α (we use $k = 20$ and $\alpha = 0.1$ in the experiments). The background image is only updated for those pixels that do not belong to targets. This procedure constitutes a simple first order recursive filter along the time axis for each pixel with the effect that slow changes such as moving shadows are integrated into the background. This method allows to create a background that is valid over a long time.

$$B_t(\phi) = \begin{cases} \alpha I_t(\phi) + (1-\alpha) B_{t-1}(\phi), (\phi) \in \text{bg} \\ B_{t-1}(\phi), \text{else} \end{cases} \quad (23)$$

In order to assure real time constraints, the detection of new targets is restricted to a small number of manually defined entry regions. This technique reduces significantly the computation time of the detection, is more robust to false positive detections and does not miss any true targets (under the condition that the entry regions cover all regions where new targets may appear).

7.2. Robust Tracking

The robust tracking module operates on the list of current targets. For each target a search region and a Gaussian mask centered on the most likely position is determined using a first order Kalman filter. The targets are then updated by collecting data from the detection module that processes the search region and computes first and second moments of the energy image weighted by the Gaussian mask. The Gaussian mask makes the tracking robust to outliers.

After the update step of each target, the module manages split and merge of targets. The distance between close targets is measured. If this distance is smaller than the merge threshold, the targets are merged.

For splitting of targets, the module performs a connectivity analysis of the pixels within the bounding box. If there are several components, their distance is evaluated and if this distance is superior to the split threshold, the target is split into its components.

7.3. Parameterization

The performance of the tracking system depends on the correct choice of the parameters. In many systems, these parameters are set manually. In this article, we use an automatic parameter regulation technique as in [4] that selects the best parameter setting with respect to an output quality metric. This metric is based on a Gaussian mixture model (GMM) computed from positive examples using a generative technique. Under the constraint that the GMM is computed from representative training data, the GMM represents the distribution of correct (positive) observations.

For the definition of this metric, we transform the hand labelled bounding boxes of the training sequences into 4-dimensional vectors $\vec{x} = (x_c, y_c, w, h)$. These 18411 examples are clustered using k-means with varying number of clusters k. The best clustering solution is retained and transformed into a GMM. For each cluster C_j we estimate the mean μ_j and covariance U_j from the data points associated to the cluster that form the parameters of the Gaussian. The resulting GMM allows to compute for any observed bounding box $\vec{x}$ the probability that this bounding box belongs to the Gaussian mixture model by following equation:

$$p(\vec{x}) = \sum_{j=1}^{K} p(\vec{x}|C_j)P(C_j) \tag{24}$$

$$p(\vec{x}|\mu_j, U_j) = \frac{1}{(2\pi)^{d/2}|U|^{1/2}} e^{-0.5(\vec{x}-\mu_j)^T U^{-1}(\vec{x}-\mu_j)} \tag{25}$$

with μ_j and U_j mean and covariance of Gaussian C_j. The priors $P(C_j)$ are estimated from the training data:

$$P(C_j) \approx \frac{|C_j|}{M} \tag{26}$$

with $|C_j|$ number of data points associated to C_j during training and M total number of data points used for training.

This metric allows to judge the quality of the output of a particular tracking system and a particular scene. Automatic parameter selection proceeds as follows. First, the parameter of the tracking system are set, then the test sequence is played and the system produces output in form of bounding boxes. This output gives rise to a quality score using the above metric. The system tests a predefined number of parameter settings and keeps the result with the best quality score. In this article, we test 400 parameter settings (20 generations of 20 individual settings) of a 4 dimensional parameter space (detection threshold, noise threshold, split parameter, merge parameter) using a genetic algorithm [7].

8. Experiments

In this section we evaluate the performance of the different methods on the same data set. The CAVIAR entry hall sequences [6] (27 sequences) are partitioned into 14 sequences for testing (13692 frames, 21217 boxes) and 13 sequences for training (12023 frames, 18411 boxes). This database contains people interacting in an entry hall of a public building at different times of the day (see Figure 2). The light regions are close to the saturation point of the camera and moves within the sequence. These are indoor sequences, but we need to deal with typical problems of outdoor scenes. We compare the performance of the following methods: *BBS* (Section 2), *W4* (Section 3), *SGM* (Section 4), *MGM* (Section 5), *LOTS* (Section 6) and *Track* (Section 7).

Figure 2: Example frame of the evaluation database.

The system performance is measured for each method by recall and precision of the targets compared to the hand-labelled ground truth (TP: correct (true positive), FP: insertion (false positive), FN: missed (false negative)) as in [1].

$$\text{recall} = \frac{\text{TP}}{\text{TP + FN}} \qquad \text{precision} = \frac{\text{TP}}{\text{TP + FP}} \tag{27}$$

Precision and recall are given for a particular overlap threshold T (see eq 28). In addition we compute the area under the curve (AUC) for precision and recall as in [10]. The AUC in this article is computed by the mean of the values sampled between [0.0,1.0] with steps of 0.001. AUC is a comparison measure with the advantage that it is independent of a particular overlap threshold. A perfect system would have AUC values of 1.0.

Figure 3 and Figure 4 display respectively recall and precision of the different methods evaluated on the test sequences. Table 1 shows the precision and recall with an overlap requirement of 50%. A correct match is registered, when the bounding boxes of the targets A_{obs} and A_{truth} overlap at least $T = 50\%$.

$$\frac{A_{obs} \cap A_{truth}}{A_{obs} \cup A_{truth}} \geq T \tag{28}$$

with

$$A(x, y, w, h) = [x - \frac{w}{2}, x + \frac{w}{2}] \times [y - \frac{h}{2}, y + \frac{h}{2}] \tag{29}$$

All methods (except *W4*) have similar recall (for an overlap of 60% and AUC). The methods should therefore be evaluated with respect to the precision. The method *Track* achieves the best precision, because it is the only method that takes into account temporal filtering using a Kalman filter. This result closely followed by *LOTS*. This is interesting, since *LOTS* does no temporal filtering. The next best performing algorithms are *SGM* followed by *MGM*. *W4* has a good precision, but since the recall is bad, this method seems not to be appropriate for the task. The simple method

Method	Recall		
	% at $T = 50\%$	abs values	AUC
BBS	42.5	9024/21217	0.379
W4	30.3	6426/21217	0.276
SGM	42.8	9075/21217	0.380
MGM	38.2	8097/21217	0.373
LOTS	47.9	10161/21217	0.375
Track	44.4	9425/21217	0.348
	Precision		
	% at $T = 50\%$	abs values	AUC
BBS	27.6	9024/32734	0.246
W4	43.0	6426/14951	0.392
SGM	46.0	9075/19711	0.409
MGM	36.7	8097/22081	0.358
LOTS	59.7	10161/17012	0.467
Track	64.8	9425/14536	0.507

Table 1: Comparison of recall and precision of the methods evaluated on the test sequences.

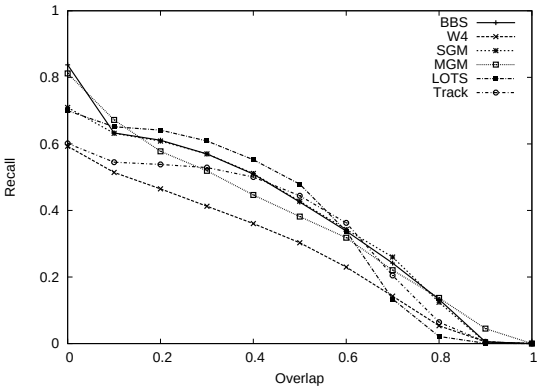

Figure 3: Comparison of recall with respect to overlap.

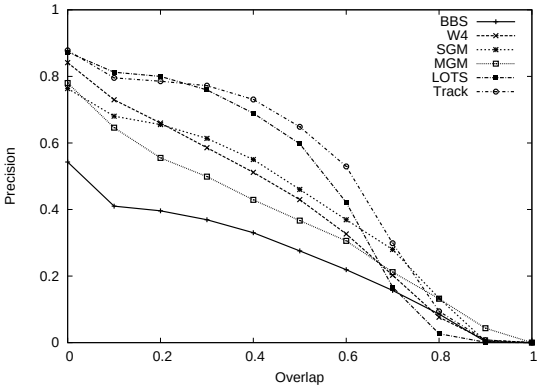

Figure 4: Comparison of precision with respect to overlap.

BBS gives the lower benchmark on precision, since it produces a very high number of false detections. The method *MGM* has the best recall for high overlap thresholds, which means that the size estimates are quite good. This is due to the connected component analysis of the methods *MGM*, *SGM*, *W4* and *BBS* for bounding box estimation.

In the experiments, several problems occurred. The detection and tracking results are evaluated with respect to hand labelled ground truth. In the case of two persons walking side by side, the detectors report one single target. The ground truth labels each person of a group as individual target. This causes problems for the detection system. Groups of persons are perceived as single targets by the detectors. For small overlap requirements, these cases[1] are counted as one correct match and one missed target. For large overlap requirements (since the bounding box covering both persons is larger), these cases are counted as two missed targets and one insertion.

The *Track* method relies on entry regions and therefore misses all targets that are within the scene at the beginning of the sequence. All detectors build a background model from several frames. Anything that is not part of this learned background is detected. The ground truth labelles targets only once they start moving. This causes a lot of insertions by the detectors, since persons waiting but not moving and the person at the reception desk are systematically detected, but not labelled in the ground truth. To remove the false detections of the receptionist, we declared a region of no detection around the reception desk.

The hand labelling chooses the smallest bounding box that covers the target. The method *LOTS* uses the result of the QCC algorithm for bounding box computation. The QCC is a lighter version of a connected components algorithm. The method *Track* estimates the bounding boxes from the position and covariance of target blobs for speed up. For compact objects, both bounding box computation methods give similar results. The sequences contain many targets where a person is raising a hand. In this case, the labelled bounding box and the bounding box estimated from the covariance are significantly different. This is the reason for the rapid decrease of correct targets with increasing overlap threshold.

Table 2 shows additional statistics for the targets with an overlap of $\geq 50\%$. All methods have equal position errors. The observation concerning the size is confirmed with these statistics. *Track* and *LOTS* perform worse than the other approaches for the sake of speed. The use of detection regions in *Track* increases the time lag for initial target detection which is compensated by very good results in continuously tracking targets once they are detected (small number of dropped frames). This is due to the temporal filtering of the Kalman filter.

[1]There are 1656 groups in the test sequences.

Error Metric [unit]	BBS mean (std.)	W4 mean (std.)	SGM mean (std.)	MGM mean (std.)	LOTS mean (std.)	Track mean (std.)
position [pix]	4.9 (4.5)	5.0 (4.8)	5.0 (4.4)	5.1 (5.8)	5.0 (5.3)	5.2 (5.8)
size [%]	3.1 (24.2)	5.6 (23.6)	-1.6 (18.5)	-0.2 (24.5)	33.5 (33.3)	11.7 (98.2)
entry [frames]	34.4 (47.5)	38.6 (126.7)	34.4 (48.2)	40.1 (45.2)	39.1 (131.1)	107.9 (193)
dropped [frames/100]	20.4 (26.5)	41.5 (27.5)	21.7 (26.8)	28.0 (29.0)	14.4 (22.4)	8.8 (14.5)
processing time [Hz]	8.3	16.7	4.5	2.8	130	70

Table 2: Additional error metrics at 50 % overlap.

9. Conclusions

We presented six types of adaptive background differencing techniques with background models of different complexity. These approaches were evaluated on indoor sequences with difficult lighting conditions. Manually labelled ground truth is available for these sequences. It is interesting to note that the most complex background model using an adaptive mixture of Gaussians per pixel *(MGM)* is outperformed by the simpler methods *SGM*, *LOTS* and *Track* both in terms of detection performance (precision and recall) and computation time.

Several problems occurred in the experiments. All detectors learn a background model from several frames. The ground truth labels only mobile targets. Persons that don't move during the sequence are considered as background. These targets are typically detect by these methods and counted as false positives. Furthermore, the ground truth contains bounding boxes for each individual person of a group. The detectors can only detect one target for a group, because we have no higher level analysis associated to the detectors. This fact significantly reduces the recognition rates of the detectors. The comparison between the detectors is still valid, because the results are obtained using the same ground truth.

The next step would be to integrate the *LOTS* method and the tracking system. This would enhance the robustness of target detection, reduce the number of false positives and allows to decrease the computation time further. Both methods currently run in real time on 1/4 PAL images. A further speed up would allow to process higher resolution images and conserve processing time for higher level image analysis.

References

[1] S. Agarwal and D. Roth. Learning a sparse representation for object detection. In *European Conference on Computer Vision*, pages 113–130, 2002.

[2] Anonymous.

[3] Anonymous.

[4] Anonymous.

[5] T.E. Boult, R.J. Micheals, X. Gao, and M. Eckmann. Into the woods: Visual surveillance of noncooperative and camouflaged targets in complex outdoor settings. *Proceedings of the IEEE*, 89(10):1382–1402, October 2001.

[6] R.B. Fisher. The PETS04 surveillance ground-truth data sets. In *International Workshop on Performance Evaluation of Tracking and Surveillance*, Prague, Czech Republic, May 2004.

[7] D.E. Goldberg. *Genetic Algorithms in Search and Optimization*. Addison-Wesley, 1989.

[8] I. Haritaoglu, D. Harwood, and L. S. Davis. W^4: real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):809–830, August 2000.

[9] M. Harville. A framework for high-level feedback to adaptive, per-pixel, mixture-of-gaussian background models. In *European Conference on Computer Vision*, pages 543–560, May 2002.

[10] J. Min, M. Powell, and K.W. Bowyer. Automated performance evaluation of range image segmentation algorithms. *IEEE Transactions on Systems Man and Cybernetics - Part B - Cybernetics*, 34(1):263–271, 2004.

[11] C. Stauffer and W.E.L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, August 2000.

[12] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, July 1997.