Adaptive Background Estimation using Intensity Independent Features

Håkan Ardö, Rikard Berthilsson,
Centre for Mathematical Sciences,
Lund University, Lund, Sweden,
ardo@maths.lth.se, rikard@maths.lth.se

Abstract

The problem of subtracting the background in an image sequence is central to computer vision. The general idea is to model the values of each pixel as a random variable. This approach has proved to be efficient when treating slowly varying changes or changes that are fairly periodic.

In this paper we propose a novel method for background and foreground estimation that efficiently handles changes that also occur with non periodicity and fast. Furthermore, the method makes only very mild assumptions about the scene making it able to operate in a wide variety of conditions. This is done by introducing a novel set of invariants that are independent to the over all intensity level in the images. By using these features instead of the raw pixel data we automatically obtain a background estimator that is insensitive to rapid changes in lighting conditions. Furthermore, the features can be computed very efficiently using the so called integral image. Inspired by the work in [17] we update the probability model over time to make it able to handle new objects entering the background, but here we work directly with the histogram which reduces the execution time considerably. Finally, we present test data that shows that the model works well in some outdoor scenes. In particular it is shown that it can handle difficult outdoor scenes with rapidly bypassing clouds.

1 Introduction

To estimate the background in real time is an important first step for many video surveillance applications. Solving this problem, in a reliable way, allows remaining resources to be devoted to tracking and identifying moving objects and to interpret events that occur in the scene. This is relevant in for example automatic traffic analysis systems, where the goal may be to increase traffic safety or increase traffic flow by reducing risk for congestion. Concerning traffic safety, an intelligent traffic monitoring system can be used to gather statistics from traffic scenes that can form the basis, for example, for redesigning dangerous street crossings. Such statistics is today gathered through costly and manual occular inspection during several days, see [9]. Furthermore, by having access to instant and accurate traffic data throughout a road net, it is possible to reduce the risk for traffic congestion and optimizing traffic flow, see [11, 14].

BMVC 2006 doi:10.5244/C.20.109

In this paper we introduce a novel method for estimating the background in an image sequence taken by a stationary video camera. It is shown that we can extract the background from moving objects in real time and in a very reliable way, also in outdoor scenes where the lighting conditions is changing rapidly due to passing clouds. This is done by introducing a novel set of intensity independent features. We propose to use the histograms for the features as an approximation for their probability functions. Furthermore, it is possible to update the probability functions very efficiently. We also include a discussion on how the background estimation can be used for detecting when cars are parking in a parking lot, by analyzing a sequence as the one shown in Figure 3. It is shown that this approach is superior to for example the background estimation method introduced in [17]. This method is modeling the probability function of each background pixel as a mixture of Gaussian distributions using the EM algorithm. The dominant component is considered to be the background and is used for estimating foreground pixels. A more recent paper [7] extends this to include pan-tilt rotations of the camera. One major problem with these methods is that on partly cloudy, windy days, the changing cloud cover causes the lighting conditions to vary faster than the algorithm adopts and thus large cloud shadows turn up as foreground objects. In this paper we efficiently solve this problem by not building the background models from the raw pixel data, but from features that are lighting independent.

The background estimation problem has been studied extensively. The simplest and most common model estimates the mean color at each pixel together with the covariance and update these estimates with each frame, see [12, 8, 18]. A more complex model, closely related to this approach, can be found in [3, 4, 6]. The use of linear combinations of Gaussian functions for modeling the probability of background pixels has been further studied in for example [2, 13]. Other methods including the use of temporal differences, range measurements, and Kalman filtering, can be found in [1, 5, 16, 15]. The importance of choice of color spaces is discussed in [10].

2 Theory

The objective is to extract the foreground and consequently also the background from a sequence of images. Problems facing us includes

- keeping execution short,
- slowly varying lighting conditions,
- rapidly varying lighting conditions, and
- what should be considered background.

Below we will use integrals instead of sums when summing over images. The reason for this is that it admits us to use convenient notation and well known result from integration theory.

The presentation below is primarily done for sequences of gray scale images. This is mainly due to notational convenience. A short discussion is also included on how color images can be treated. Thus, let $I_t(x,y)$, where $(x,y) \in \mathbb{R}^2$ and $t = 0,1\ldots$ be a sequence of gray scale images. We have extended the images to be defined on the entire real plane. This is done by setting the image intensity equal to zero outside the real image.

2.1 Class of features

In order to compute a feature at each location we can use convolution

$$f_t(x,y) = I * h = \iint_{\mathbb{R}^2} I_t(x-a,y-b)h(a,b)dadb, \tag{1}$$

where h is a spatial filter mask. This gives a filter response at every point $(x,y) \in \mathbb{R}^2$ and the statistical properties of these can be used to classify background and foreground. The well known Stauffer–Grimson estimator is obtained by letting $h = \delta_{0,0}$ be the Dirac measure at the origin in which case $I * \delta_{0,0} = I$, i.e. we base the estimator on the raw pixel data.

It is a well know problem that many background estimators are sensitive to rapid changes in lighting. Such rapid changes are often present and can occur for example when a cloud suddenly occludes the sun, when moving objects casting shadows, or for fast changes in indoor lighting.

In order to deal with this problem we would like to use features that are independent to changes that behave at least locally in a nice manner. For this reasons we assume that there exists some constant c such that

$$I_{t+1}(x+\Delta x, y+\Delta y) = cI_t(x+\Delta x, y+\Delta y), \quad (\Delta x, \Delta y) \in \Omega, \tag{2}$$

where Ω is some neighborhood of $(0,0)$. We call the condition (2) that the image sequence is **locally proportional**. Local proportionality is usually fulfilled, at least approximately, for most points (x,y) if Ω is sufficiently small. It is however not fulfilled for example *on* the boundary of a moving shadow, but it fulfilled on both sides of the boundary. To take advantage of the locally proportional assumption we introduce two filters $\phi(x,y)$ and $\psi(x,y)$ such that ϕ and ψ are non-zero only on Ω. Recall, for a function $f : \mathbb{R}^2 \to \mathbb{R}$, the notation

$$\operatorname{supp} f = \{(x,y) \mid f(x,y) \neq 0\}. \tag{3}$$

It follows that $\operatorname{supp}\phi \subseteq \Omega$ and $\operatorname{supp}\psi \subseteq \Omega$.

We propose to use

$$g_t(x,y) = \frac{I_t * \psi}{I_t * \phi} \tag{4}$$

as features for each $(x,y) \in \mathbb{R}^2$. Now it follows from the locally proportional assumption that

$$g_{t+1}(x,y) = \frac{I_{t+1} * \psi}{I_{t+1} * \phi} = \frac{cI_t * \psi}{cI_t * \phi} = \frac{I_t * \psi}{I_t * \phi} = g_t(x,y). \tag{5}$$

This means that for points (x,y) that fulfill the local proportionality the features $g_t(x,y)$ are independent of changes lighting.

2.2 Feature probabilities

To deal with points (x,y) in background that do not fulfill the locally proportional condition we follow the idea of updating probability functions. With this method it is well-known that we can deal with slowly varying changes and also in some cases (semi)-periodic fast changes such as for example the branches of a tree swaying in the wind. However, instead of using a parameterized probability functions whose parameters are

updated we choose to work with histograms (probability functions). Comparing with the parameterized probability functions such as multi modal Gaussian distributions the histogram makes no assumptions on the function. It is furthermore very straightforward to use being both easy to update and very fast in execution. Let (x_k, y_k), $k = 0, \ldots, n-1$, be a set of fixed points in the images. We would like to estimate the probability function for the values of

$$g_t(x_k, y_k), \qquad k = 0, \ldots, n-1, \tag{6}$$

and we want the probability function to be updated dynamically keeping it accurate for all times $t \geq 0$. Let $p_{k,t}$ be this probability function, which is dependent on what point (x_k, y_k) we are looking at and what time t we have. There are many ways to model the unknown probability function $p_{k,t}$. One of the simplest would probably be to model it as a Gaussian probability function and then estimating the mean and variance. Another more elaborate path would be to model $p_{k,t}$ as a linear combination of Gaussian probability functions as proposed by [17]. The crucial point here is that we use a new set of features that are intensity independent and we can use any suitable model for estimating $p_{k,t}$. However, in this paper we propose to work directly with the histogram that can be obtained over time. This leads to low execution time at the cost of a moderate increase in memory requirements. Furthermore, being nonparametric, the histogram makes no assumptions about the probability function.

Assuming that the probability function $p_{k,t}$ varies slowly with t we can estimate it from the histogram $q_{t,k}$ which is obtained by computing $g_t(x_k, y_k)$ for some values of t while keeping k constant. In order to compute the histogram we first have to choose some bins $a_0, \ldots a_{n-1}$ and round off the value of $g_t(x_k, y_k)$ to the nearest bin. We denote the nearest bin to $g_t(x_k, y_k)$ by $\overline{g}_t(x_k, y_k)$. The probability function is then obtained through normalization as

$$p_{t,k}(a_j) = \frac{q_{t,k}(a_j)}{\sum_j q_{t,k}(a_j)}, j = 0, \ldots, n-1 \tag{7}$$

where a_j are the bins. In order to update the probability function with a new measurement $\overline{g}_{t+1}(x_k, y_k)$ we introduce

$$\gamma_{t,k}(a_j) = \begin{cases} 1, & \overline{g}_t(x_k, y_k) = a_j \\ 0, & \text{otherwise} \end{cases}. \tag{8}$$

We then update the probability function according to

$$p_{t+1,k} = (1-\alpha) p_{t,k} + \alpha \gamma_{t+1,k}. \tag{9}$$

Note that $\sum_j p_{t,k}(a_j) = 1$ for all time t and points (x_k, y_k). The constant $0 \leq \alpha \leq 1$ is referred to as the learning constant. The larger it is the faster the old probability function is forgotten. In order to make the updating faster we use the fact that

$$\frac{\sum_j p_{t,k}(a_j)}{(1-\alpha)} q_{t+1,k} = q_{t,k} + \frac{\alpha \sum_j q_{t,k}(a_j)}{(1-\alpha)} \gamma_{t,k}. \tag{10}$$

The factor,

$$\frac{\alpha \sum_j q_{t,k}(a_j)}{(1-\alpha)}, \tag{11}$$

becomes independent of k if α is kept constant and if the starting value is

$$q_{0,k}(a_j) = \begin{cases} 1, & g_0(x_k, y_k) = a_j \\ 0, & \text{otherwise} \end{cases}, \tag{12}$$

for all k. We then update the histogram according to

$$q_{t+1,k} := q_{t,k} + \frac{\alpha \sum_j q_{t,k}(a_j)}{(1-\alpha)} \gamma_{t,k}, \tag{13}$$

which is obtained by replacing the left hand side of (10) with $q_{t+1,k}$. This gives the histogram up to scale. However, the scale is not important here. It follows that the update is done very fast, requiring only one addition per histogram. Note also that $\sum_j q_{t,k}(a_j)$ only depends on t and not k. Thus, computing the probability $p_{t,k}(a_j)$ requires only division by a number that is the same for all positions (x_k, y_k).

2.3 Optimized features

There is a great degree of freedom in choosing the filters ψ and ϕ which can be exploited to optimize the accuracy. A natural choice is to choose ψ such that $\operatorname{supp} \psi$ is approximately of the same size as the objects (foreground) we would like to be able to detect. The filter ϕ can then be chosen such that $\operatorname{supp} \phi$ is large enough to contain a significant portion that is not covered by an object of interest. A more elaborate way would be to find ψ and ϕ such that the feature $g_t(x_k, y_k)$ has significantly different values when (x_k, y_k) is a foreground pixel compared to when it belongs to the background. This will however require that we annotate the image sequence with what regions are foreground and background by hand before training. However, it is worth noting that foreground objects will eventually become background if they stop moving in the scene regardless of what features we choose. This is due to the dynamic updating of the histograms. The optimization of filters should be thought of as optimization on average between specific moving objects and general background.

A problem is that the features $g_t(x_k, y_k)$ may take arbitrarily large values in a worst case scenario and this will cause problem when estimating a histogram. Assume that we require our feature values to be integers in the interval $[0, n-1]$. As a first step we would then like to find an affine transformation $h : \mathbb{R} \to \mathbb{R}$ such that $h(g_t(x_k, y_k)) \in [0, n-1]$ as often as possible. Let $m_{t,k}$ and $\sigma_{t,k}$ be the mean and standard deviation of the values of $g_t(x_k, y_k)$, respectively. we update these values according to

$$m_{t+1,k} = (1-\alpha) m_{t,k} + \alpha g_t(x_k, y_k) \tag{14}$$

and

$$\sigma_{t+1} = (1-\alpha)\sigma_{t,k} + (1-\alpha)|g_t(x_k, y_k) - m_{t,k}|, \tag{15}$$

where α is the learning constant as described above. We now transform the feature $g_t(x_k, y_k)$ by the affine transformation

$$\frac{g_t(x_k, y_k) - m_{t,k}}{\sigma_{t,k}} \frac{n}{4} + \frac{n}{2} \tag{16}$$

which has mean equals to $n/2$ and standard deviation $n/4$. To be sure that we never get values out side the interval $[0, n-1]$, we introduce the transformed features

$$h_t(x_k, y_k) = \left[\min\left(\max\left(\frac{g_t(x_k, y_k) - m_{t,k}}{\sigma_{t,k}} \frac{n}{4} + \frac{n}{2}, 0 \right), n-1 \right) \right], \tag{17}$$

where $[x]$ denotes rounding to nearest integer $\leq x$.

2.4 General intensity independent features

In this section we show a general result for intensity independent features. Let $\mathbb{R}_+$ be the set of positive real numbers. Then an intensity independent feature $f : \mathbb{R}_+^n \to \mathbb{R}^m$ is characterized by the property $f(x) = f(cx)$ for any real $c > 0$ and $x \in \mathbb{R}_+^n$. It is easily seen that

$$f(x) = \sum_{j=1}^{n} m_j \log(x_j) \tag{18}$$

is intensity independent if $\sum_{j=1}^{n} m_j = 0$. The following theorem shows that the reverse is also true, i.e. all intensity independent features can be written as sums of the form (18). For convenience we introduce the notation $\log(x)$ to denote the vector $(\log(x_1), \ldots, \log(x_n))$ and similarly for $\exp(x)$.

Theorem 2.1. *A feature $f : \mathbb{R}_+^n \to \mathbb{R}^m$, where $n > 1$, is intensity independent if and only if it can be written as*

$$f(x) = g(M \log(x)), \qquad x \in \mathbb{R}_+^n, \tag{19}$$

for some $g : \mathbb{R}^{n-1} \to \mathbb{R}^m$ and M is an $(n-1) \times n$ matrix with row sums equal to 0.

Proof. Let $x \in \mathbb{R}_+^n$. Then, $f(x) = f \circ \exp \circ \log(x)$. Let P be a $n \times n$ non singular matrix such that the first row of P contains only ones and the other rows are orthogonal to this. The the row sums are $= 0$ except for the first row. It follows that $f(x) = f(\exp(P^{-1} P \log(x)))$. Set $h(y) = f(\exp(P^{-1} y))$. Set

$$P = \begin{pmatrix} N \\ M \end{pmatrix} \tag{20}$$

where N is a $1 \times n$ matrix with only ones and M an $(n-1) \times n$ matrix. It then follows that $N \log(cx) = n \log(c) N \log(x)$ and $M \log(cx) = M \log(x)$. The intensity independence gives that $h(P \log(cx)) = h(P \log(x))$ for all $x \in \mathbb{R}_+^n$ and $c > 0$, implying that $h(a_0, b) = h(a, b)$, where $a_0, a \in \mathbb{R}$ and $b = M \log(x) \in \mathbb{R}^{n-1}$. The theorem follows by setting $g(b) = h(a_0, b)$. ∎

3 Implementation

The convolution for computing (4) can of course be done using FFT with a computational cost of $\mathcal{O}(MN \log(MN))$ for $M \times N$ images. However, if we let the filters ϕ and ψ, be simple functions like for example Haar wavelets then we can use the well known integral image to speed up the computation. Let

$$J(x, y) = \int_{-\infty}^{x} \int_{-\infty}^{y} I(a, b) da db \tag{21}$$

be the integral image of a gray scale image I. Then J can be computed with a computational cost of about $4MN$ additions for an $M \times N$ image I. For notational convenience we just treat the filter ϕ below and furthermore we assume that

$$\phi(x,y) = \begin{cases} 1, & |x| \leq c, |y| \leq c \\ 0, & \text{otherwise} \end{cases}. \tag{22}$$

In order to fulfill $\operatorname{supp}\phi \subseteq \Omega$ we should choose $0 < c$ sufficiently small. It follows that

$$I * \phi(x,y) = J(x-c,y-c) + J(x+c,y+c) - J(x-c,y+c) - J(x+c,y-c) \tag{23}$$

requiring only four additions for each (x,y). A general Haar wavelet is a linear combination of functions like (22) resulting in maximum of 16 additions for each (x,y). A system was implemented in C running on image sequences having 352×288 in resolution. For each pixel (x_k,y_k) we

1. compute features $g_t(x_k,y_k)$
2. update $m_{t,k}$ and $\sigma_{t,k}$
3. compute the affinely transformed features $h_t(x_k,y_k)$
4. update the corresponding histograms.

This runs at about 20 frames per second on a 2.4 GHz P4 processor. A binary image, estimating the foreground, was obtained by setting a threshold for the probabilities. This threshold was the same for each pixel (x_k,y_k).

4 Extensions

For color images we can apply the same machinery as above by for example treating each color channel separately. This requires that the color channels are independent which is normally not the case for RGB. However, it does hold approximately for other types of color codings. We can also use several filters ϕ_j and ψ_j, $j = 0,1,\ldots$. This improves the accuracy at the cost of more memory requirements and computational load. Here again we have to assume independence if the model should be manageable.

5 Experiments

By using a 16 minutes (20 fps) video sequence monitoring a parking lot, the proposed algorithm was compared to the one suggested in [17]. The sequence was recorded on a cloudy and windy day. Four frames, just as a cloud shadow passes over the ground, are shown in Figure 1.

The corresponding binary output images from the two algorithms are shown in Figure 2. Figure 2 clearly shows that the cloud shadows, as displayed by the method [17], is almost entirely gone with the proposed method. Furthermore, the bicycles are still shown as foreground.

The input sequence is 352x288 pixels and the proposed algorithm processes 21 frames per second on a 2.4 GHz P4. This is significantly faster than our implementation of the

Figure 1: Four frames from the cloudy input sequence used to test the proposed algorithm.

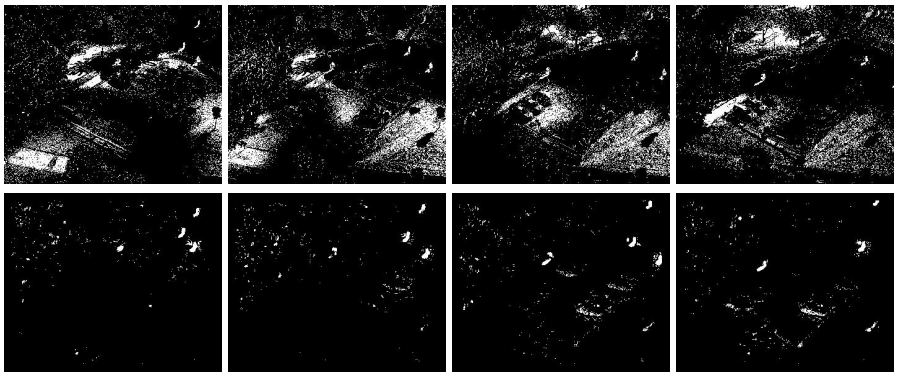

Figure 2: The first row shows the result from the method in [17] and to the second row the result from the proposed method when applied to the frames in Figure 1.

algorithm in [17] that processes four frames per second on the same computer and input sequence.

In order to obtain quantitative comparison between the two algorithms a simple application was implemented on top of them. It is a first stage in a parking lot monitoring system that counts the number of parked cars in real time. The application detects the event that a car is entering or exiting the parking. This is done by counting the number of foreground pixels, N, in a predefined rectangle covering the entrance of the parking lot. If $N > \sigma_1$, where σ_1 is some threshold, an event is detected and the image is saved. Then when $N \leq \sigma_2$, for some $\sigma_2 < \sigma_1$ a new event is triggered. The saved images are then inspected and compared with the ground truth.

The cloudy sequence mentioned above contains four events, all consisting of a car exiting the parking lot. And the proposed algorithm found all four of them and no false positives. The four images saved are shown in Figure 3. Executing the same event detection, based on [17], on the same input sequence resulted in 18 events detected. The four cars exiting are still detected and the addition 14 false positives are cloud shadows moving past the parking lot entrance. The first four detections are shown in Figure 4.

As a final test the proposed event detector were tested on a 16 hour image sequence acquired from 16:30 in the afternoon until 08:30 the next morning. The sequence contains 32 events, both cars exiting and entering, and quite a lot of lighting variations as it contains both a sunset and a sunrise. The proposed system detected 34 events, including the 32 correct ones. The first additional false detection consisted of five pedestrians simultaneously entering the parking lot, and the second of a car driving past the entrance from

Figure 3: All events detected from the parking lot monitoring algorithm based on the proposed algorithm.

Figure 4: First four events detected from the parking lot monitoring algorithm based on [17].

one place in the parking lot to another. In both cases the proposed background/foreground segmentation algorithm generated the expected data, and the misstates were made by the simplistic event detection algorithm. Four detected events chosen from the entire sequence are shown in Figure 5.

Figure 5: Four detected events chosen from the events detected by the parking lot monitoring algorithm based on the proposed algorithm when tested on a 16 hour sequence.

6 Conclusions

This paper introduces a novel method for estimating the foreground and background in image sequences which could either gray scale or color. This is done by introducing a class of features that are independent of changes in lighting under mild assumptions. The probability functions for the values of the features are approximated by the sampled histogram which are dynamically updated at each frame. This results in an algorithm with low computational cost and consequently high execution speed. It is shown that the proposed method can estimate the foreground reliable in situations where traditional methods fail. Experiment have been conducted with a system for detecting when cars are parking on a parking lot. The experiment was done on a image sequence that contained

a lot of moving clouds casting shadows in the scene. Four cars were detected which was correct according to manual inspection. The method [17] detected 18 cars in this image sequence.

References

[1] M. Bonnisegna and A. Bozzoli. A tunable algorithm to update a reference image. *Signal Processing: Image Communication*, 1998.

[2] A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. In *6th European Conference on Computer Vision*, 2000.

[3] D. Farin et al. Robust background estimation for complex video sequences. In *International Conference on Image Processing*, pages 145–148, 2003.

[4] D.S. Gao, J. Zhou, and L.P. Xin. A novel algorithm of adaptive background estimation. In *International Conference on Image Processing*, 2001.

[5] G. Gordon, T. Darrell, M. Harville, and J. Woodfill. Background estimation and removal based on range and color. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1999.

[6] I. Haritaoglu, D. Harwood, and L. Davis. A real time system for detecting and tracking people. *Image and Vision Computing Journal*, 1999.

[7] E. Hayman and J-O. Eklundh. Background subtraction for a mobile observer. In *Proc. 9th Int. Conf. on Computer Vision, Nice, France*, 2003.

[8] J. Heikkila and O. Silven. A real-time system for monitoring of cyclists and pedestrians. In *Second IEEE Workshop on Visual Surveillance*, pages 74–81, 1999.

[9] C. Hydén. *The development of a method for traffic safety evaluation: The Swedish traffic conflicts technique*. PhD thesis, Institutionen för trafikteknik, LTH, Lund, 1987.

[10] P. Kumar, K. Sengupta, and A. Lee. A comparative study of different color spaces for foreground and shadow detection for traffic monitoring system. In *The IEEE 5th International Conference on Intelligent Transportation Systems*, pages 100– 105, 2002.

[11] G. Medioni, I. Cohen, F. Bremond, S. Hongeng, and R. Nevatia. Event detection and analysis from video streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 873 – 889, 2001.

[12] J. Orwell, P. Remagnino, and G.A. Jones. Multi-camera colour tracking. In *Second IEEE Workshop on Visual Surveillance*, pages 14–21, 1999.

[13] R. Pless, J. Larson, S. Siebers, and B. Westover. Evaluation of local models of dynamic backgrounds. In *Conference on Computer Vision and Pattern Recognition*, page 73, 2003.

[14] A. Prati, I. Mikic, C. Grana, and M. M. Trivedi. Shadow detection algorithms for traffic flow analysis: a comparative study. In *IEEE Intelligent Transportation Systems*, pages 340 – 345, 2001.

[15] C. Ridder, O. Munkelt, and H Kirchner. Adaptive background estimation and foreground detection using kalman-filtering. In *Proceedings of International Conference on recent Advances in Mechatronics*, pages 193–199, 1995.

[16] Huwer S. and Niemann H. Adaptive change detection for real-time surveillance applications. In *Third IEEE International Workshop on Visual Surveillance*, pages 37–46, 2000.

[17] Chris Stauffer. Adaptive background mixture models for real-time tracking. In *Proc. Conf. Computer Vision and Pattern Recognition*, pages 246–252, 1999.

[18] R Wren and et al. Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 780–785, 1997.