Background modelling and subtraction
for object detection in video

Niel Joubert
14428008

Final year project 2009

Abstract

Background modelling/subtraction in surveillance and security applications is very important, because this is the first step in detecting and identifying objects or people in videos. In this work we discuss and compare numerous background modelling/subtraction methods in an effort to determine which one is the best for object detection in videos. We discus four methods in depth: basic background subtraction, mixture of Gaussians, kernel density estimators and eigenbackgrounds. These are then also used to perform tests on three video sequences. The results of the tests are compared to find the best method for detecting objects in videos.

Contents

1 Introduction

Background modelling/subtraction in surveillance and security applications are very important, because this the the first step in detecting and identifying objects or people in the videos. It is normally used to roughly identify objects and people as the first step in finally detecting and identifying a person.

A model of the scene background is built and each pixel in the image is analyzed. A pixel's deviation in colour and/or intensity values is used to determine whether the pixel belongs to the background or the foreground. This information is then grouped together to form regions in the image.

There are various problems in background modelling that must be addressed or handled. A few are listed below:

- gradual and sudden changes in lighting conditions;
- if a moving object is present during the initialization of the background scene;
- any shadows that is present at any time;
- movement through cluttered areas;
- objects that overlap in the image;
- effects of moving elements of the scene background (e.g. swaying tree branches);
- slow-moving objects;
- if a foreground object becomes motionless, it would appear to be the same as a background object that moves and then becomes motionless;
- background objects that are inserted or removed from the scene that become/was part of the background;
- if an object that was part of the background is moved, both it and the part of the background that was behind the object appear to change;
- if a foreground object's pixel characteristics are almost the same as the background.

In this work, a few of the known background subtraction techniques are discussed and compared. Figure 1 provides an overview. In section 2 we discuss the basic and simple methods of frame differencing (FD) and running Gaussian averaging (RGA). In section 3 a mixture of Gaussians (MoG) is used to approximate the background distribution at every pixel. Section 4 discusses the method of kernel density estimation (KDE) and section 5 the method of eignbackgrounds (EB). In section 6 we list and briefly discuss

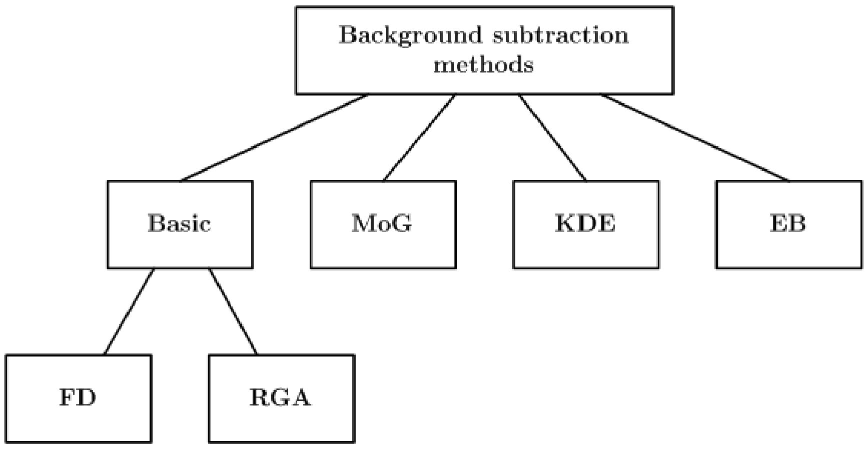

Figure 1: Techniques for background modelling and subtraction considered in this work: basic frame difference (FD) and running Gaussian average (RGA), mixture of Gaussians (MoG), kernel density estimation (KDE), and eigenbackgrounds (EB).

a few other techniques, for the sake of completeness. In section 7 some experimental results are presented, and we conclude in section 8.

2 Basic background subtraction

This section contains a few basic methods for background subtraction. Although simple to implement, they rarely work well in practise. The methods do, however, provide a basis from which more sophisticated methods can be developed.

2.1 Frame differencing

The background image can be calculated by subtracting the current frame from the previous frame or from the average image of a number of frames. It only works well in particular conditions of object speed and frame rate and is very sensitive to the threshold.

According to this scheme, pixels belongs to foreground if

$$|I_i(x,y) - I_{i-1}(x,y)| > T, \tag{1}$$

where I_i is the current frame, I_{i-1} is the previous frame and T is a chosen threshold.

The median/average method uses the average or the median of the previous n frames as the background image, B. It is quick but very memory consuming. The required memory is $n \times \text{size(frame)}$.

The running average can be calculated as follows:

$$B_i = \alpha I_{i-1}(x,y) + (1-\alpha)B_{i-1}(x,y), \tag{2}$$

where α is the adapt or learning rate (usually chosen as 0.05).

The running average can be expanded to accommodate selectivity, as follows:

$$B_i(x,y) = \begin{cases} \alpha I_{i-1}(x,y) + (1-\alpha)B_{i-1}(x,y), & \text{if } I_{i-1}(x,y) \text{ is background,} \\ B_{i-1}(x,y), & \text{if } I_{i-1}(x,y) \text{ is foreground.} \end{cases} \tag{3}$$

Selectivity just means that if the pixel was classified as foreground, it is ignored in the new background model. Using this we prevent the background model to contain a pixel believed to be foreground.

The background model at each pixel location is based on the pixel's recent history. The history can be the average of the previous n frames or the weighted average, where recent frames have higher weight. The weighted average is computed as the chronological average from the pixel's history and no spatial correlation is used between different (neighbouring) pixel locations.

At each new frame, each pixel is classified as either foreground or background. The selective background model ignores any pixels which are classified as foreground by the classifier.

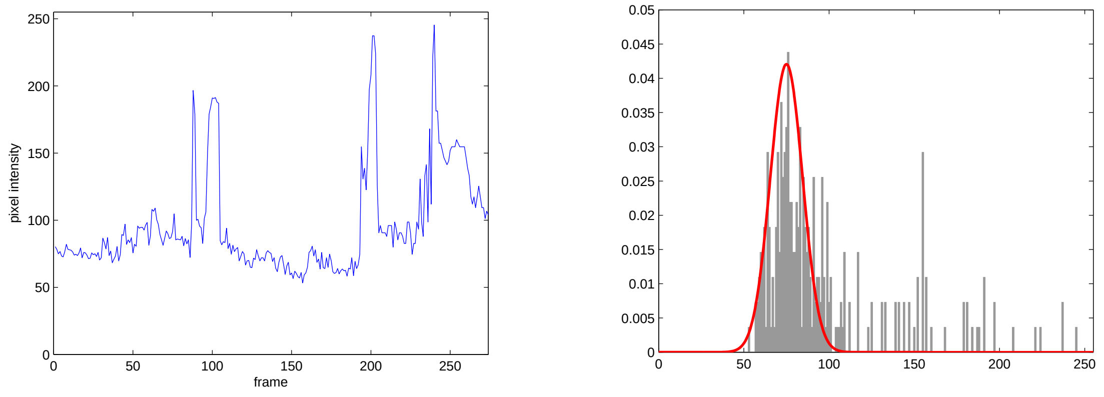

Figure 2: The history of a pixel's intensity value over a number of frames (left) and a normalized histogram of that history with Gaussian background model (right).

The limitations of basic background subtraction are that it does not provide an explicit method to choose the threshold and it can not cope with multiple modal background distributions (for a single value of the threshold).

2.2 Running Gaussian average

PFinder, a method proposed in [1], works by fitting one Gaussian distribution (μ, σ) over the histogram of the history of each pixel, which is then used as the background probability density function (PDF) for a particular pixel. See Figure 2 for a visual representation.

At the next frame, the background PDF is updated by using a running average as follows:

$$\begin{aligned} \mu_i(x,y) &= \alpha I_{i-1}(x,y) + (1-\alpha)\mu_{i-1}(x,y), & (4) \\ \sigma_i^2(x,y) &= \alpha(I_{i-1}(x,y) - \mu_{i-1}(x,y))^2 + (1-\alpha)\sigma_{i-1}^2(x,y). & (5) \end{aligned}$$

The update is followed by testing if $|I_i(x,y) - \mu_i(x,y)| > T$, where the threshold T can be chosen as some constant multiple of $\sigma_i(x,y)$.

A disadvantage of this method is that it does not cope with multimodal background distributions, where a single Gaussian PDF would be insufficient.

3 Mixture of Gaussians

A single Gaussian, as used in the previous section, may be enough to model the pixel value over time if each pixel came from the same surface under the same lighting. This would also have accounted for acquisition noise. Also, a single, adaptive Gaussian may be enough if the lighting is the only thing that changes over time.

As we know, images usually consist of multiple surfaces and lighting conditions change. Thus, multiple adaptive Gaussians may be necessary.

A mixture of adaptive Gaussians can be used to approximate this process, as proposed in [2]. Each time the parameters are updated, the Gaussians are evaluated and a simple heuristic is used to decide which are most likely to be part of the background.

3.1 Approach

The values of a particular pixel over time is modelled as a mixture of Gaussians, instead of modelling the values of all the pixels as one particular type of distribution. We determine which Gaussians may correspond to background colours. The pixel values that do not fit the background are assumed to be part of the foreground, until a Gaussian that includes them with sufficient, consistent evidence is reached.

The incorporation of slowly moving objects into the background takes longer, because their colours have a larger variance than the background. Repetitive variations are also learned, and a model for the background distribution is maintained, which leads to faster recovery when objects are removed from further frames.

3.2 Method

Consider a particular pixel (x, y). For notational convenience we define $\mathbf{c}_i$ to be the value of the pixel in frame i, i.e.

$$\mathbf{c}_i = I_i(x, y), \tag{6}$$

where I is the image sequence and I_i is the current frame.

Note that $\mathbf{c}_i$ is assumed to be a 3-element vector containing, for example, the red, green and blue components of the pixel.

The history of pixel (x, y), at any given time t, is therefore given by the ordered set

$$\{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_t\}. \tag{7}$$

An aspect of variation occurs if moving objects are present in the scene. A moving object will normally produce more variance than a stationary object.

We model this history by a mixture of K Gaussian distributions, so that the probability of observing the current pixel value is

$$P(\mathbf{c}_t) = \sum_{i=1}^{K} \omega_{i,t} \eta(\mathbf{c}_t; \boldsymbol{\mu}_{i,t}, \boldsymbol{\Sigma}_{i,t}), \tag{8}$$

where K is the number of distributions, $\omega_{i,t}$ is an estimate of the weight (what portion of the data is accounted for by this Gaussian), $\boldsymbol{\mu}_{i,t}$ is the mean value, $\boldsymbol{\Sigma}_{i,t}$ is the covariance matrix of the ith Gaussian in the mixture at time t, and η is a Gaussian probability density function:

$$\eta(\mathbf{c}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{\frac{d}{2}} |\boldsymbol{\Sigma}|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{c}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{c}-\boldsymbol{\mu})}. \tag{9}$$

Here d is the number of dimensions of $\mathbf{c}$, in this case $d = 3$.

K is determined by the available memory and the computational power, normally between 3 and 5 is used. For computational reasons the covariance matrix is assumed to be of the form:

$$\boldsymbol{\Sigma}_{k,t} = \sigma_{k,t}^2 \mathbf{I}, \tag{10}$$

where $\mathbf{I}$ is the 3×3 identity matrix.

This assumes that the red, green and blue pixel values are independent and have the same variances. It is most probably not true, but this assumption allows us to avoid a costly matrix inversion at the expense of some model accuracy.

Thus, by virtue of (8), the distribution of recently observed values of each pixel in the scene is characterized by a mixture of Gaussians. A new pixel value will normally be represented by one of the components of the mixture model and can be used to update the model.

Every new pixel value, $\mathbf{c}_t$, is checked against the existing K Gaussian distributions until a match is found. It is a match if the pixel value is within 2.5 standard deviations of the mean of that distribution. This threshold can be changed slightly with little effect on performance.

If none of the K distributions match the current pixel value, the least probable distribution is replaced that has a distribution with the current value as its mean. The distribution initially has high variance and low prior weight. The prior weights of the K distributions at time t, $\omega_{k,t}$, are adjusted as follows:

$$\omega_{k,t} = \alpha M_{k,t} + (1 - \alpha)\omega_{k,t-1}, \tag{11}$$

where α is the learning rate and $M_{k,t}$ is 1 for the model that matched and 0 for the rest.

After this approximation, the weights are normalized. The time constant $1/\alpha$ determines the speed at which the distribution parameters change.

The μ and σ parameters for the unmatched distributions remain the same, but the parameters of the distribution that matches the new observation are updated as follows:

$$\begin{aligned} \boldsymbol{\mu}_{k,t} &= \rho \mathbf{c}_t + (1-\rho)\boldsymbol{\mu}_{k,t-1}, & (12) \\ \sigma^2_{k,t} &= \rho(\mathbf{c}_{k,t} - \boldsymbol{\mu}_{k,t})^T(\mathbf{c}_{k,t} - \boldsymbol{\mu}_{k,t}) + (1-\rho)\sigma^2_{k,t-1}, & (13) \end{aligned}$$

where

$$\rho = \alpha\, \eta(\mathbf{c}_{k,t}; \boldsymbol{\mu}_{k,t-1}, \boldsymbol{\Sigma}_{k,t-1}). \tag{14}$$

Equations (12), (13) and (14) are effectively a type of low-pass filter, except that only the data which matches the model is included in the estimation.

An advantage of using this method is that the existing background is not destroyed if something is allowed to become part of the background. The original background colour remains in the mixture until it becomes the Kth most probable and a new colour is observed. An example of how this is useful is when an object remains motionless, just long enough to become part of the background, and it then moves again. The distribution describing the previous background does still exist with the same mean and variance, but with a lower weight and will quickly be re-incorporated into the background.

4 Kernel density estimators

In this section we discuss kernel density estimation (KDE) for application in background modelling [3]. If we keep track of the intensity values of a single pixel over time, with no background movement, the intensity can be modelled with a Gaussian kernel, given that the image noise, over the same time, can be modelled by a Gaussian distribution with zero mean.

4.1 Density estimation

The KDE model obtains the most recent information about the image sequence and continuously updates this information to obtain fast changes in the background (swinging branches etc.). The intensity distribution of a pixel can change quickly so we estimate this density function of this distribution at any time, by only using the most recent history information.

Let $\mathbf{c}_1$, $\mathbf{c}_2$, ..., $\mathbf{c}_N$ be the most recent sample of intensity values of a pixel. Using these, the probability density function of the specific pixel at time t, that will have a value $\mathbf{c}_t$, can be non-parametrically estimated [4] with the kernel estimator K as:

$$P(\mathbf{c}_t) = \frac{1}{N}\sum_{i=1}^{N} K(\mathbf{c}_t - \mathbf{c}_i). \tag{15}$$

We now choose the kernel estimator function K to be a Gaussian function with zero mean and covariance matrix Σ, so that:

$$P(\mathbf{c}_t) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{c}_t-\mathbf{c}_i)^T\Sigma^{-1}(\mathbf{c}_t-\mathbf{c}_i)}. \tag{16}$$

The covariance matrix Σ is referred to as the kernel function bandwidth.

If we can assume that the different colour channels are independent but have different kernel bandwidths, σ_j^2 for the jth color channel, then

$$\Sigma = \begin{pmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{pmatrix}. \tag{17}$$

Now the density estimation reduces to:

$$P(\mathbf{c}_t) = \frac{1}{N}\sum_{i=1}^{N}\prod_{j=1}^{d} \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{1}{2\sigma_j^2}(c_{t,j}-c_{i,j})^2}, \tag{18}$$

where $c_{i,j}$ indicates the jth component of $\mathbf{c}_i$.

By using this probability it can be decided that the pixel is part of the foreground if $P(\mathbf{c}_t) < T$, where T is the threshold that can by adjusted to achieve a desired percentage of false positives.

Each group of N samples is considered to be a zero mean Gaussian distribution by itself. This allows the estimation to be done more accurately and to depend only on the most recent information from the image sequence.

Another advantage of doing this is that the model ignores ("forgets") the previous samples and focuses more on the more recent samples.

4.2 Kernel width estimation

There are at least two reasons why a pixel's intensity value changes. Firstly, different objects are projected to the same pixel at different times. Secondly, the pixel can be a projection of the same object, but may change as a result of image blurring.

The kernel bandwidth indicates the local variance in pixel intensity because of the local variation from image blur and not the intensity jumps (when different objects are projected to the same pixel). The local variation may change over time and can also differ between colour channels in the kernel calculation.

To estimate the kernel bandwidth σ_j^2, for the jth colour channel of a given pixel, we calculate the median absolute deviation over the sample for consecutive intensity values. In other words, we calculate the median m of $|c_{i,j} - c_{i-1,j}|$ for each consecutive pair $(c_{i-1,j}, c_{i,j})$ in the sample. This median is calculated separately for each colour channel.

Since the deviations are integer values, linear interpolation can be used to obtain more accurate median values.

4.3 Suppression of false detection

There are mainly two sources of false detections in outdoor environments. Firstly, false detection can occur due to random noise. This is usually homogeneous over the entire image. Secondly, false detection can result from small movements in the background, that are not represented by the background model. This can happen, for example, when tree branches move further than it did when we generated the model. Another example of small movements would be due to camera movement. It is usually spatially clustered in the image and difficult to estimate using morphology and noise filtering, because these operations also affect small and/or occluded targets.

4.3.1 Removing unmodelled movements

If a background object moves to another pixel, that is not part of that pixel's model, then it will be detected as a foreground object. This object will however have a high probability of being part of the background distribution at the original pixel.

Assuming that only small displacements can happen between consecutive frames, we look at the distributions in a small neighbourhood of the detection.

Suppose that $\mathbf{c}_t$ is the observed pixel that was detected as a foreground pixel at time t. Let $P_N(\mathbf{c}_t)$ be the maximum probability that $\mathbf{c}_t$ belongs to the background distribution of a point in the neighbourhood N of the pixel. That is,

$$P_N(\mathbf{c}_t) = \max_{j \in N} P(\mathbf{c}_t | B_j), \tag{19}$$

where B_j is the background sample for pixel j and the probability estimation $P(\mathbf{c}_t|B_j)$ is calculated using the kernel function estimation (18).

Now we can threshold $P_N(\mathbf{c}_t)$ to obtain the detected pixels. True detections could also be made, because they might be similar to the background of a nearby pixel. This typically happens on grayscale images.

To avoid eliminating pixels that may accidentally be similar to the background of a nearby pixel (also known as true detections), we check that the entire detected foreground object has moved from a certain location, and not only some of its pixels. This is known as component displacement.

The component displacement probability, P_C, is the probability that a detected connected component C has moved from another location. It is estimated by:

$$P_C(\mathbf{c}_t) = \prod_{\mathbf{c} \in C} P_N(\mathbf{c}_t). \tag{20}$$

P_C will be very small if a real object has moved from the background.

Finally, a detected pixel is part of the background only if $(P_N(\mathbf{c}_t) > T_1) \wedge (P_C(\mathbf{c}_t) > T_2)$, where T_1 and T_2 are two chosen thresholds.

4.4 Updating the background

The sample that was used in the previous sections contains N intensity values that could be taken over a window of time with size W. The kernel bandwidth estimation requires that the sample consist of N consecutive intensity values. Thus $N = W$.

This sample can be updated continuously to adapt to changes in the scene. The sample is updated in a FIFO manner. The new sample is chosen randomly from each interval of length $\frac{W}{N}$ frames.

There are two ways to update the background:

1. **Selective update:** add a new sample to the model only if it is classified as a background sample.
2. **Blind update:** simply add a new sample to the model.

4.4.1 Selective update

The selective update enhances detection of the targets, since the targets are not added to the model. Here a decision needs to be made whether each pixel value belongs to the background or not. To do this we use the decision result as an update decision. If an incorrect decision is made here it will cause a deadlock situation [5].

So if, for example, a tree branch might have moved and stayed still in the new location for a long time it would be detected for the total duration.

4.4.2 Blind update

The blind update does not suffer from the deadlock situation, because no update decisions need to be made. It does allow intensity values to be added to the model that do not belong to the background. In turn, this leads to the detection of more false positives, because they become part of the model. This can be reduced if we increase the time window, such that a smaller portion of the target pixels is included in the sample.

More false positives will occur if the time window is increased, because the adaptation to changes is slower and rare events are not as well represented in the sample.

To build a background model that adapts quickly to the changes, supports sensitive detection and a low rate of false positives, we combine the result of two background models, namely a short- and a long-term model. By combining these models we can try to take the best features of each one.

4.4.3 Short-term model

The short-term model consists of the most recent N background sample values. It adapts to changes quickly, to allow for sensitive detection. A *selective-update* mechanism is used to update the sample. The update decision is based on a mask $M(x, t)$, where

$$M(x,t) = \begin{cases} 1, & \text{if the pixel } i \text{ should be updated at time } t, \\ 0, & \text{otherwise.} \end{cases} \tag{21}$$

The mask is driven by the final result of combining the two models.

4.4.4 Long-term model

The long-term model consists of N background sample values taken from a larger window than the short-term model. It captures a more stable representation of the background and adapts to changes slowly (based on a ratio of W/N). A *blind-update* mechanism is used to update the sample, so that all the new samples are added to the model. This model is expected to contain more false positives because it is not the most recent background model, and contain more false negatives as the target pixels may be included in the sample.

By combining these two models the false positives from the short- and from the long-term models are removed. The only false positives that are left will be the rare events that are not represented in either model. The long-term model should adapt to these rare events if they continue to appear, and suppress them from the result later on.

The true positives in the short-term model will be suppressed if they are false negatives in the long-term model. This happens because the long-term model adapts to targets if they are standing still or moving slowly. To overcome this, all the pixels that are detected by the short-term model that are adjacent to the pixels that are detected by the combination of the two, are include in the final result.

5 Eigenbackgrounds

In this section we describe the method of using eigenbackgrounds [6] to model the background image. We adaptively build an eigenspace that models the background. This eigenspace model describes the range of variations in intensity values that have been observed.

If we take frame i with size $w \times h$, where w is the width and h is the height, we must transform it into a $wh \times 1$ column vector $\mathbf{x_i}$. This is done by placing the first column into a new vector and then placing the second column into the vector after the first one, and so forth.

The model is formed by taking a sample of N images. The mean image $\mathbf{m}$ is calculated by:

$$\mathbf{m} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i. \tag{22}$$

The mean normalized image vectors are then packed into a $wh \times N$ matrix X

$$X = [\ \mathbf{x}_1 - \mathbf{m} \quad \mathbf{x}_2 - \mathbf{m} \quad \cdots \quad \mathbf{x}_N - \mathbf{m}\ ]. \tag{23}$$

The columns of X, corresponding to frames in the video, all lie on a wh-dimensional space. Because the frames are very similar it is very likely that these columns can be represented in some lower dimensional subspace.

For this reason we calculate the singular value decomposition (SVD) of X:

$$X = U \Sigma V^T, \tag{24}$$

where U is an orthogonal $wh \times wh$ matrix and V^T is an orthogonal $N \times N$ matrix. The singular values of X are contained in the $wh \times N$ diagonal matrix Σ, in non-increasing order.

If the first r singular values are non-zero, while the rest is zero, the first r columns of U gives an orthogonal basis for the column space of X. Moreover, if the first r singular values are non-zero and the rest are sufficiently close to zero, then the first r columns of U form an approximate basis for the column space of X. In our case the columns of X should be fairly correlated, implying that the singular values decrease quickly.

We can therefore approximate the subspace spanned by the columns of X quite well by considering, as a basis, only the first r columns of U, where $r \ll N$.

This is known as principal component analysis (PCA). We keep only the first r columns of U, which in this case is also referred to as the eigenbackgrounds, in a matrix U_r.

Any new image $\mathbf{y}$ can now be projected onto the reduced subspace as

$$\tilde{\mathbf{y}} = U_r \mathbf{p} + \mathbf{m}. \tag{25}$$

Since U_r is orthogonal, $\mathbf{p}$ is easily obtained as

$$\mathbf{p} = U_r^T(\mathbf{y} - \mathbf{m}), \tag{26}$$

Moving objects do not have a significant contribution to this model, because they hopefully do not appear in the same location in the N sample images and they are typically small. The portions of an image that contain a moving object can not be well described by this eigenspace model (except in very unusual cases). Whereas the static portions of the image can be accurately described as a linear combination of the various eigenbasis vectors. To summarize, the eigenspace provides a robust model of the PDF of the background, but not for the moving objects.

Once the eigenbackground images are obtained, as well as the mean $\mathbf{m}$, we can project each input image onto the space spanned by the eigenbackground images, by Equation (25), to model the static parts of the scene that belongs to the background.

Therefore, by computing and thresholding the absolute difference between the input image and the projected image we can detect the moving objects present in the scene as follows:

$$|y_i - \tilde{y}_i| > T, \tag{27}$$

where T is a given threshold and y_i is the ith element of $\mathbf{y}$.

To get the final image in the correct $w \times h$ form, we place the first h elements from $\mathbf{y}$ into the first column of F and the second h elements into the second column of F and so forth.

6 Other methods

In this section we discuss a few other background subtraction methods found in the literature. It is important to know about as many methods as possible when designing a system for object detection.

6.1 Mean-shift based estimation

Mean-shift based estimation [7] is a gradient-ascent method that is able to detect the modes of a multimodal distribution, along with their covariance matrices. The method is iterative, thus it is very slow and the memory requirements are very high. A way to overcome the computation issue is to use this method only for detecting the background PDF modes at initialization time and then use a method that is computationally lighter, such as mode propagation.

6.2 Combined estimation and propagation (sequential kernel density approximation)

This method uses mean-shift mode detection, but only at initialization time. After initialization the modes are propagated by adapting them with the new samples. This method is also faster than kernel density estimation and has low memory requirements. See [8] for more details.

6.3 Optical flow

The optical flow method involves estimating the displacement field between two images. It is usually used when correspondences between pixels are needed. The aim of this method is to approximate a projection of the three dimensional velocities of surface points onto a two dimensional surface. There are also a few techniques for determining optical flow. See [9] for a detailed analysis of the various techniques, as well as their performances. Since this method determines what pixels are moving, we can identify the moving objects.

7 Results

In this section we apply various background subtraction methods on three video sequences. The methods used are:

- frame differencing (FD), Section 2.1;
- average median (AM), Section 2.1;
- mixture of Gaussians (MoG), Section 3;
- and eigenbackgrounds (EB), Section 5.

Test were run on three video sequences: *walking*, *reflections* and *hockey*. The camera is stationary in all of the video sequences, except near the end of the *hockey* video where the camera moves a bit.

The *walking* sequence consists of 83 frames at 180×144 resolution, and depicts a man walking from the left side of the frame to the right in front of a brick wall. The first frame is not an empty background, since the man is already present.

The *reflections* sequence consists of 380 frames at 640×480 resolution, and depicts a man walking from the left side of the frame to the right. The floor that he walks on is reflective and he walks past a cabinet with a glass door.

The *hockey* sequence consists of 1500 frames at 352×288 resolution, and depicts a group of hockey players practicing at night. The floodlights are on and thus there are shadows present.

The methods were implemented in Matlab and were executed on a 3500+ AMD Athlon 64bit processor with 2GB of RAM.

7.1 Visual comparison of results

It is very difficult to precisely determine how accurate each method is. Some methods will work better on certain images while on other images it may be inadequate. The only way to test how good each method works is to physically mark the outlines in each image. Then the outlines of the subtracted images must be compared to the original images to determine the actual accuracy of the method.

Figure 3 shows the differences between frame differencing (FD), average median (AM), mixture of Gaussians (MoG) and eigenbackgrounds (EB), that was applied on a frame from each of the three video sequences. EB does the best job of the four methods at finding moving objects, but also detects may false positives. The shadows are detected in the *walking* (shadow on the ground) and *hockey* (shadows that are a result of the

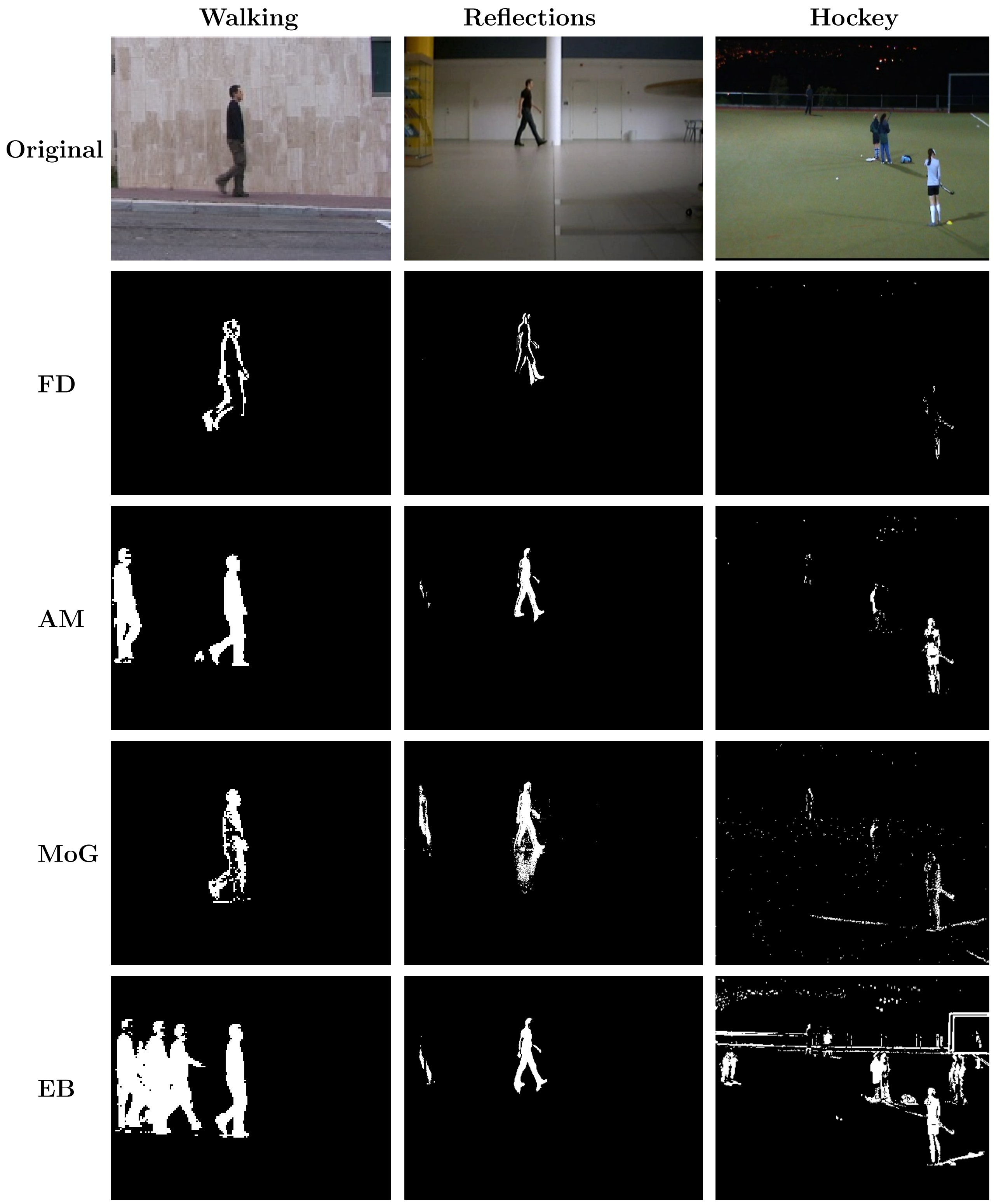

Figure 3: Three frames from the middle of each video sequence on which the following methods have been applied, frame differencing (FD), average median (AM), mixture of Gaussians (MoG) and eigenbackgrounds (EB).

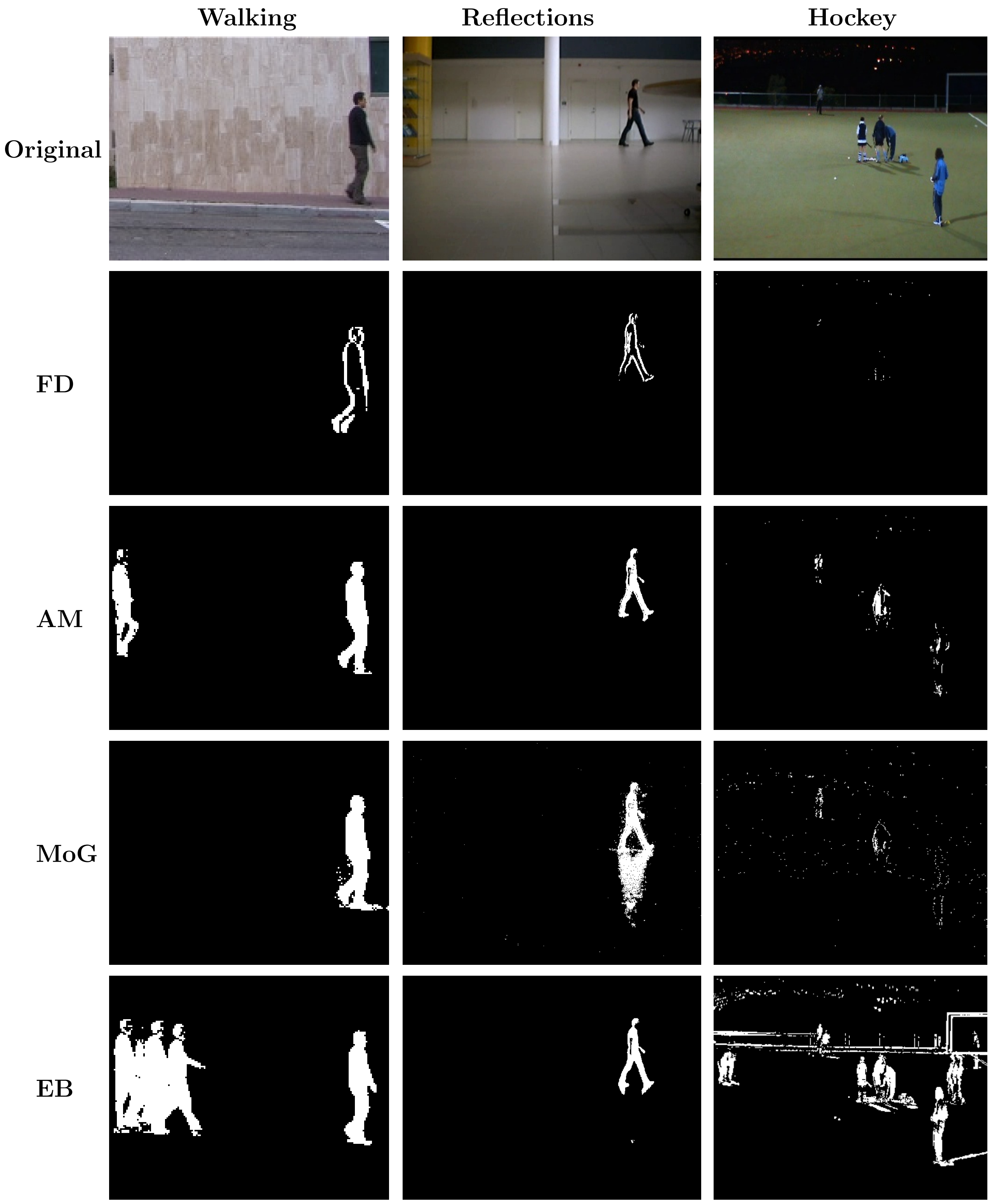

Figure 4: The final frames from each video sequence on which the following methods have been applied, frame differencing (FD), average median (AM), mixture of Gaussians (MoG) and eigenbackgrounds (EB). This shows how well each method works over the whole image sequence.

	Walking		**Reflection**		**Hockey**	
	AVG	FPS	AVG	FPS	AVG	FPS
FD	0.002	595.530	0.026	38.604	0.010	95.638
AM	0.004	243.052	0.061	16.462	0.020	49.208
MoG	1.142	0.8758	14.41	0.0694	6.060	0.1650
EB	0.004	274.459	0.088	11.382	0.026	38.687

Table 1: Execution times (in seconds) of various methods including their average time per frame (AVG) and their frames per second (FPS). The methods used are frame differencing (FD), average median (AM), mixture of Gaussians (MoG) and eigenbackgrounds (EB).

floodlights). In the *reflections* image, the reflections against the wall and tile floor are detected.

The object on the right in the *walking* AM image is there, because it is present during the initialization of the background image. The FD method calculates the images as shown, because the colours of the pixels inside the outline stays the same throughout the video. The left part of the EB *walking* image, that looks like a blurred man walking, is there because of the initialization of EB that uses a certain amount of frames to determine the background image.

When looking at Figure 4 we see that in *hockey*'s final images EB shows all the objects present and MoG shows the outlines of three big objects, but also some noise. FD shows no significant objects and AM shows three significant objects too, but not as clearly as the MoG image.

Looking at *reflections*'s final images we see that EB shows the object with a little bit of its shadow on the floor and MoG shows the object with almost its complete shadow on the floor with some noise. FD shows the outline of the object and AM shows the object too and this time, more clearly than the MoG image.

Now finally, looking at *walking*'s final images we see that EB shows the object with its shadow, but also an object, that looks like a blurred man walking that is caused by the initialization of EB. MoG shows the object with its shadow on the floor and some noise. FD shows the outline of the object, but not the complete outline. The AM shows the object, but has the same object on the left of the frame, because it was there during initialization.

Figure 4 shows results on different frames of the videos. Observations similar to the ones above can be mad for these results.

7.2 Execution times

Table 1 shows the average time that is used to process a frame and also the frames per second (FPS). Again, the methods tested are frame differencing (FD), average median (AM), mixture of Gaussians (MoG) and eigenbackgrounds (EB). The times were calculated by getting the system time before the algorithm were applied to the frame, and then subtracted from the system time after the algorithm completed.

The training times for EB on the three video sequences, *walking*, *reflections* and *hockey*, are 2.529, 34.154 and 8.052 seconds respectively. The training time is directly related to the resolution of the image, since the whole image must be "reshaped" into a one-dimensional vector from an $m \times n$ matrix and an SVD must then be determined. With the *reflections* images having a resolution of 640×480, it is logical that the training time for this sequence should be the highest and *walking* with a resolution of 180×144 should be the lowest.

We can clearly see that MoG is by far the slowest and FD is the fastest of all the methods, over all three video sequences. This is because the MoG has more complex calculations to do than the FD method. The means and the weights of the components is calculated during the initialization of the MoG method. This goes through each pixel of each color. After the initialization, the difference of each pixel from the mean is calculated after which the Gaussian components (mean and weight) for each component is updated. Now the background image can be calculated and finally thereafter, the foreground can be calculated.

FD, on the other hand, amounts to one frame being subtracted from another, and is therefore much faster.

7.3 Summary

Table 2 is a summary of the various methods with regards to their speed, memory requirements and accuracy. The methods in the part of the table above the line, were discussed in previous sections. The other methods were briefly discussed in section 6.

The following methods were analyzed, normal average, median, running average, mixture of Gaussians, kernel density estimation (KDE), eigenbackgrounds, standard- and optimized mean-shift, sequential kernel density approximation (SKDE) and optical flow.

Note that it is impossible to precisely determine what the accuracy of each method is. An unbiased comparison, with a significant benchmark, is needed.

	Speed	**Memory Requirements**	**Accuracy***
Average	Fast	High	Acceptable
Median	Fast	High	Acceptable
Running average	Fast	Low	Acceptable
Mixture of Gaussians	Intermediate	Intermediate	Good
KDE	Intermediate	High	Better
Eigenbackgrounds	Intermediate	Intermediate	Good
Standard mean-shift	Slow	High	Better
Optimized mean-shift	Intermediate	Intermediate	Better
SKDA	Intermediate	Intermediate	Better
Optical flow	Intermediate	Intermediate	Intermediate

Table 2: Summary of the various methods that was discussed in this work, with regards to their speed, memory requirements and accuracy.

8 Conclusions

A few shortcomings that we came across is in the basic background subtraction (BBS) and the running Gaussian average (RGA) methods. The BBS (section 2) does not provide provide an explicit method to choose the threshold and it can not cope with multiple modal background distributions for a single value of the threshold. As well as RGA (section 2.2) that does not cope with multimodal backgrounds.

By looking at the results of the experiments performed in Section 7.1 on the three video sequences, it is relatively obvious that the eigenbackgrounds method worked the best in detecting objects. The only problem that we could see was that if there was that if there was objects in the initial N frames needed for the initialization of the model, then there would be "ghost" images in the result. For a visual representation, see the *walking* images in Figures 3 and 4.

The mixture of Gaussians method worked second best since it detected the most detail in the video sequences after the eigenbackgrounds methods. It also does not require that the initialization frames must only contain the background image.

If we look at the execution times of the methods in Section 7.2, EB is significantly faster than MoG, and a bit slower than FD and AM. FD, on the other hand, is the fastest by a significant margin, because of the small amount of calculations that has to be done by the technique.

References

[1] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder:Real-time Tracking of the Human Body," IEEE Trans. on Patt. Anal. and Machine Intell., vol. 19, no. 7, pp. 780-785, 1997

[2] Stauffer C, Grimson W. E. L, "Adaptive background mixture models for real-time tracking," in Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149). IEEE Comput. Soc. Part Vol. 2, 1999.

[3] Elgammal A., Harwood D., Davis L, "Non-parametric model for background subtraction," in IEEE ICCV'99 FRAME-RATE WORKSHOP. 1999.

[4] D. W. Scott, "Multivariate Density Estimation," Wiley-Interscience, 1992.

[5] K.-P. Karmann and A. von Brandt, "Moving object recognition using and adaptive background memory, in Time-Varying Image Processing and Moving Object Recognition," Elsevier Science Publishers B.V., 1990.

[6] N. M. Oliver, B. Rosario, and A. P. Pentland, "A Bayesian Computer Vision System for Modeling Human Interactions," IEEE Trans. on Patt. Anal. and Machine Intell., vol. 22, no. 8, pp. 831-843, 2000.

[7] M. Piccardi and T. Jan, "Mean-shift Background Image Modelling", 2004

[8] B. Han, D. Comaniciu, and L. Davis, "Sequential kernel density approximation through mode propagation: applications to background modeling," Proc. ACCV -Asian Conf. on Computer Vision, 2004. Running Gaussian average.

[9] J. L. Barron, D. J. Fleet, and S. Beauchemin (1994), "Performance of optical flow techniques," International Journal of Computer Vision (Springer).