BACKGROUND SUBTRACTION WITH EIGEN BACKGROUND METHODS USING MATLAB

[1]Ilmiyati Sari
[2]Nola Marina

[1]Pusat Studi Komputasi Matematika, Universitas Gunadarma
e-mail: ilmiyati@staff.gunadarma.ac.id

[2]Pusat Studi Komputasi Matematika, Universitas Gunadarma
e-mail: nola@staff.gunadarma.ac.id

ABSTRACT

Most background models are pixel-based and it is not robust to unstable background. One frame-based background model is eigen background. In this paper, we present eigen background model and to support the discussion of the theory, we perform some simulations using the software Matlab. The result of experiment is to obtain good background we just use one eigen vector corresponding one largest eigen value. We obtain foreground image with add one parameter, i. e threshold. The used threshold is 60.

I. INTRODUCTION

Detection of foreground objects in video often requires robust techniques for background modeling. The basic idea of background modeling is to maintain an estimation of the background image which should represent the scene with no foreground objects and must be kept updated frame by frame so as to model both static and dynamic pixels in the background. Moving objects can then be detected by a simple subtraction and threshold procedure.

Many background modeling methods for background subtraction have been proposed. Most background models are pixel-based and very little attention is given to region-based or frame-based methods. Typical models, among many others, include Kalman filters [1], Gaussians [2], mixture of Gaussians (MoG) [3, 4] and kernel density estimation [5]. Despite being capable of solving many background maintenance issues, pixel-based models are less efficient and effective in handling light switching and time-of-day problems [6]. Rather, frame-based background models may potentially handled such a problem more efficiently. One frame-based background model is eigen background.

In this paper, we present eigen background model and to support the discussion of the theory, we perform some simulations using the software Matlab.

The paper is organized as follows. In section II, we describe eigen space model, efficient implementation of eigen space model and background subtraction with eigen background while results and conclusion are presented in section III and IV.

II. EIGEN BACKGROUND

In this section a description of the eigen background algorithm will be given. We will start with the definition of eigen space. The key element of this method lies in its ability of learning the background model from unconstraint video sequence, even when they contain moving foreground objects. Eigen takes into account neighboring statistics. It thus has a more global definition on background which, hopefully, make it more robust to unstable backgrounds [7].

II.1 EIGEN SPACE MODELS

We will first introduce the concept of eigen value decomposition applied to the case of a sequence of images. We follow mainly the definitions of [8]. The following notation will be used: *italics* with subscripts to indicate vectors and matrixes ($A_{m,n}$ is a matrix of *m* rows and *n* columns), bold letters with subscripts for images ($\boldsymbol{B}_{h,w}$ is an image with height *h* and width *w*).

Given an image $\boldsymbol{I}$, of size h, w (height, width) it can be rewritten as a column vector x,

$$\boldsymbol{I}_{h,w} \rightarrow x_{n,1}, \tag{1}$$

where $n = hw$. Given a sequence of *N* images $\boldsymbol{x}_{n,1}^1, \boldsymbol{x}_{n,1}^2, \cdots, \boldsymbol{x}_{n,1}^N$, the average image is computed as:

$$\bar{x}_{n,1} = \frac{1}{N}\sum_{i=1}^{N} x_{n,1}^i. \tag{2}$$

The covariance matrix of the sequence of images is given by:

$$C_{n,n} = \frac{1}{N}\sum_{i=1}^{N}\left(x_{n,1}^i - \bar{x}_{n,1}\right)\left(x_{n,1}^i - \bar{x}_{n,1}\right)^T. \tag{3}$$

The same covariance matrix can be obtained, by rearranging the sequence of image in single matrix $X_{n,N}$ in which each column contains one image:

$$X_{n,N} = [x_{n,1}^1 \quad x_{n,1}^2 \quad \cdots \quad x_{n,1}^N]. \tag{4}$$

From this $C_{n,n}$ can be obtained by:

$$C_{n,n} = \frac{1}{N}\left(X_{n,N} - \bar{x}_{n,1} II_{1,N}\right)\left(X_{n,N} - \bar{x}_{n,1} II_{1,N}\right)^T, \quad (5)$$

in which is a matrix in which all elements are set to 1.

By definition a covariance matrix is real and symmetric. Its eigenvalue decomposition is guaranteed to exist then, and its given by:

$$C_{n,n,} U_{n,n} = U_{n,n,} A_{nn}, \quad (6)$$

where $U_{n,n}$ are eigenvectors, and $A_{n,n}$ is a diagonal matrix of eigenvalues, ordered form the most significant one to the least relevant one. The eigenvectors are orthonormal, so that $U_{n,n}^T U_{n,n} = I_{n,n}$, the identity matrix.

The memory requirement for such an algorithm are quite demanding. Given a set of N images of n pixels each, the covariance matrix obtained, $C_{n,n,}$ will have a size of n^2. Let's as an example, consider an image acquired with a VGA camera, considered nowadays a low resolution device. The amount of memory required to store such a matrix is then:

$$w = 640, h = 480 \ \rightarrow n = 307200 \ \rightarrow n^2 = 9.40e10.$$

Moreover, the computation of an eigen value decomposition for a matrix of size n^2 with such large values is absolutely prohibitive.

Clearly, the simplest solution is to greatly reduce the adopted image resolution, with the obvious side effect of a lower quality representation of the scene.

II.2 Efficient Implementaion of Eigen Space Model

Fortunately a much more efficient implementation is obtained by simply observing the following set of equations [8,9]. Let's once again rearrange a sequence of N images as $X_{n,N}$ given by equation (4). We can then compute matrix as:

$$D_{N,N} = \frac{1}{N}\left(X_{n,N} - \bar{x}_{n,1} II_{1,N}\right)^T \left(X_{n,N} - \bar{x}_{n,1} II_{1,N}\right). \quad (7)$$

If we define:

$$A_{n,N} = \left(X_{n,N} - \bar{x}_{n,1} II_{1,N}\right). \quad (8)$$

We can rewrite $C_{n,n}$ and $D_{N,N}$ as:

$$C_{n,n} = A_{n,N} A_{n,N}^T. \quad (9)$$

$$D_{N,N} = A_{n,N}^T A_{n,N}. \quad (10)$$

As for , an eigen value decomposition for $D_{N,N}$ exists and it's given by:

$$D_{N,N}\, V_{N,N} = V_{N,N} \Lambda_{\mathrm{N,N}}. \quad (11)$$

Let's now take any of the eigen vectors of $D_{N,N}$, $V_{N,1}^1$ with its respective eigen value δ_i. By definition of eigen value decomposition, the following equations hold:

$$A_{n,N}^T A_{n,N} V_{N,1}^i = \delta^i V_{N,1}^i, \tag{12}$$

$$A_{n,N} A_{n,N}^T A_{n,N} V_{N,1}^i = \delta^i A_{n,N} V_{N,1}^i , \tag{13}$$

$$\left(A_{n,N} A_{n,N}^T\right) A_{n,N} V_{N,1}^i = \delta^i A_{n,N} V_{N,1}^i \tag{14}$$

If we define:

$$P_{n,1}^i = A_{n,N} V_{N,1}^i. \tag{15}$$

And using equation (9), we can then rewrite equation (14) as:

$$C_{n,n} P_{n,1}^i = \delta^i P_{n,1}^i. \tag{16}$$

By definition of eigen decomposition, $P_{n,1}^i$ is an eigen vector of $C_{n,n}$ and δ^i is the respective eigen value. It is then clear that we can obtain the needed eigen decomposition of matrix $C_{n,n}$ by solving the decomposition of matrix $D_{N,N}$.

The advantage is that, in typical applications, the number of observations *N* is much smaller than the number of pixels in the image *n*. We can thus write:

$$N << n,$$

$$P_{n,N} = A_{n,N} V_{N,N},$$

$$C_{n,n} P_{n,N} = P_{n,N} \Lambda_{N,N}.$$

Matrix $P_{n,N}$ is guaranteed to be orthogonal, but it is not orthonormal. We then need to normalize each vector. Two methods are possible:

$$U_{n,1}^i = P_{n,1}^i / \left\| P_{n,1}^i \right\|, \tag{17}$$

$$U_{n,1}^i = P_{n,1}^i / \sqrt{\delta_i}. \tag{18}$$

The second method (18), can be derived following equation in [7]. The advantage of using (18) when compared to (17) is that it allows to save some computation, because term does not need to be computed. Even if this can appear as minor detail, in a real time implementation it can speed up the computation.

$$C_{n,n} U_{n,N} = U_{n,N} \Lambda_{N,N}. \tag{19}$$

Equation (19) solves the same problem of equation (6), reducing greatly the amount of computation and memory needed, with the only side effect of returning the *N* eigen vectors out of the total available *n*, corresponding to the *N* most signigicant eigen values. This does not create problems though, since the number of significant eigen vectors, even for a scene with many lighting variations is seldom higher than few tens.

II.3 Background Subtraction with Eigen Background

In this section we describe how to exploit the decomposition of a sequence of images to build a background model and achieve a simple background subtraction algorithm.

In the typical scenario, a background model is constructed using a sequence of frames captured with a video camera, while no object which should be detected as foreground appears in the scene. These images are arranged in matrix $X_{n,N}$ following equation (4) and the eigen value decomposition is obtained through equation (26). In order to be robust to illumination changes taking place in the scene, the best option is to select a sequence of images in which the expected variations is mostly present. The model consists thus of the following matrices:

$\bar{x}_{n,1}$ = average model image

$\Lambda_{k,k} = k$ largest eigen values

$U_{n,k}$= corresponding k eigen vectors.

The detection of foreground is obtained using the following steps. First, given a captured image, rearranged as a vector $x_{n,1}^i$, the reconstruction of such image through the background model is :

$$r_{k,1} = U_{n,k}^T\left(x_{n,1}^i - \bar{x}_{n,1}\right), \tag{20}$$

and then reconstructed approximation image $\left(x_{n,1}^i\right)'$ as follows

$$\left(x_{n,1}^i\right)' = U_{n,k} r_{k,1} + \bar{x}_{n,1}. \tag{21}$$

Finally, foreground moving pixels are detected by computing the distance between the input image $x_{n,1}^i$ and reconstructed one $\left(x_{n,1}^i\right)'$

$$\chi_{n,1}^i = \begin{cases} 1 & if\ d_0\left(x_{n,1}^i, \left(x_{n,1}^i\right)'\right) > \tau \\ 0 & otherwise \end{cases}, \tag{22}$$

where τ is a threshold and d_0 is defined as follows

$$d_0 = \left|x_{n,1}^i - \left(x_{n,1}^i\right)'\right|.$$

III. Experimental Results

As an important part of this paper, we also present the experimental results of our implementation. Our experiments are employed on Matlab R2009a on 32-bit windows system .

We first analyse the effect of including in the model images containing foreground, such as car moving of the street. Figure 1 shows some of the frames that were used to compute the background model. Figure 2 shows the obtained average image with 1 largest eigen vectors and Figure 3 shows the background subtraction obtained with different eigen values.

Figure 1: Frames of the sequence that were used to compute the background model. In total 51 frames were used to compute the model.

Figure 2: Image and eigen vectors, corresponding to1 largest eigen values, of the background model obtained with the sequence shown in Figure 1.

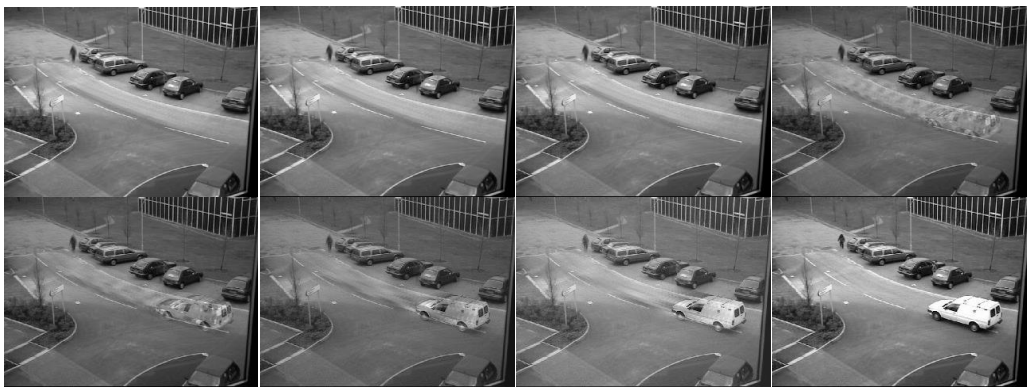

Figure 3: Top: form left to right: the background image use 5, 10, 20, and 30 largest eigen value
Bottom: from left to right: background image use 40, 45, 48, and 51 (all) largest eigen value

The background image in Figure 2 and 3, we reconstruct it with formula in equation (21). From Figure 2 it is clear, we can get background from each sequence image only use 1 largest eigen value. Figure 3 show the effect of eigen value against the background image in more detail. Growing number of the largest eigen values are used, then it will be more visible foreground.

Based on section 2.3, foreground image we can obtain from equation (22). Then to change graylevel image that we get in equation (22), we use threshold to get foreground image of binary image. Figure 4 show foreground image with different eigen value.

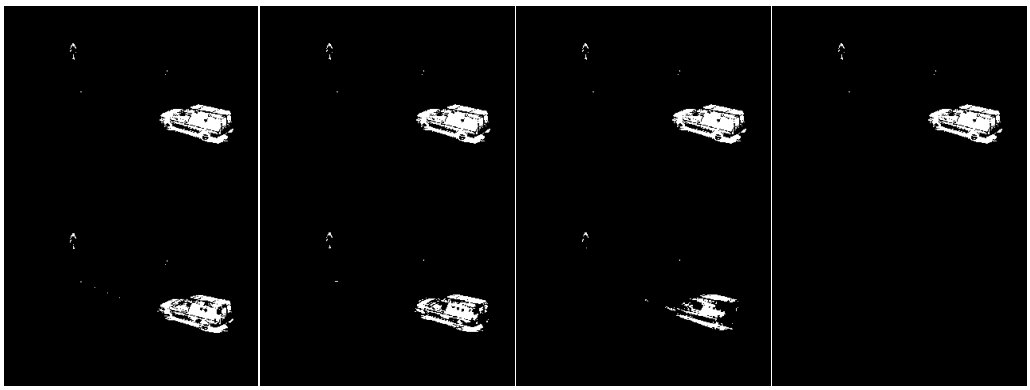

Figure 4: Foreground image and eigen vectors, corresponding to *k* largest eigen values, of the background model obtained with the sequence shown in Figure 1. Used threshold is 60.

From Figure 4, it is clear that the largest number of eigen values can be used to obtain the good foreground image is 40. In Figure 4, used threshold is 60, because we have done simulation and we obtain 60 is the best threshold value for all images. Figure 5 shows foreground image reconstruct with 1 largest eigen value and different threshold.

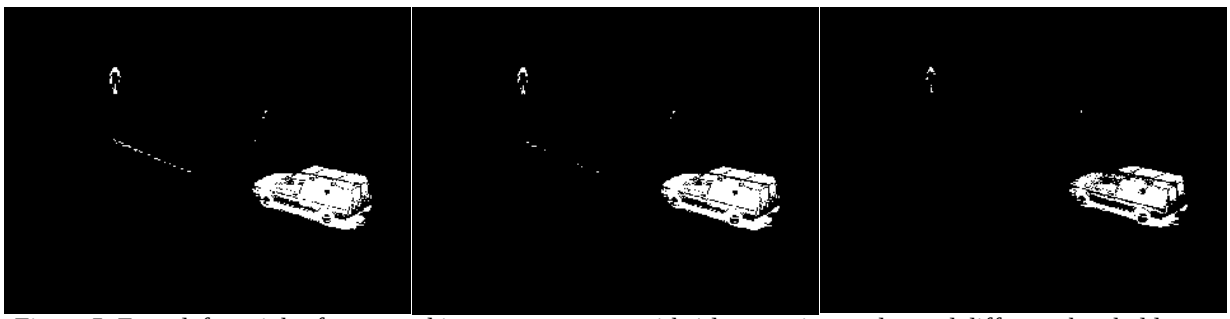

Figure 5. From left to right. foreground image reconstruct with 1 largest eigen value and different threshold (τ). $\tau = 50$, $\tau = 55$, and $\tau = 70$.

IV. Conclusion

We presented the theory and implementation of background subtraction based on eigen spaces. Effects of most parameters (number of *k* largest eigen vector and threshold) choice have been illustrated, to provide the reader with the insights needed to make an

informed choice when selecting these parameters. The final choice is strongly dependent on the application.

From experiment, it can be concluded that to obtain the background image, we only use 1 eigen vector corresponding 1 largest eigen value. Foreground image is obtained by equation (21) with 1 eigenvector corresponding 1 largest eigen value and threshold equals 60. This threshold can be used for each sequence image.

References

[1] Koller, D., Weber, J., Huang, T., Malik, J., Ogasawara, G., Rao, B., Russell, S.: Towards robust automatic traffic scene analysis in real-time. In: Proceedings of the International Conference on Pattern Recognition, Israel (1994) 126–131

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

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

[4] Lee, D.S.: Effective gaussian mixture learning for video background subtraction.IEEE Transactions on Pattern Analysis and Machine Intelligence **27**(5) (2005) 827–832

[5] Elgammal, A., Duraiswami, R., Harwood, D., Davis, L.S.: Background and foreground modeling using non-parametric kernel density estimation for visual surveillance. Proceedings of the IEEE **90**(7) (2002) 1151–1163

[6] Toyama, K., Krumm, J., Brumitt, B., Meyers, B.: Wallflower: Principles and practice of background maintenance. In: ICCV (1). (1999) 255–261

[7] Benezeth, Y., Jodoin, P. M., Emile, B., Laurent, H., Rosenberger, C.: Comparative study of background subtraction algorithms. Journal of Electronic Imaging (19). (2010).

[8] Karaman, M., Goldmann, D., Yu, D., Sikora, T.: Comparison of static background segmentation methods. Visual Communications and Image Processing . (2005).

[9] Turk, M., Pentland, A.: Eigen for Recognition. Journal of Cognitive Neuroscience. (1991).