LIGHT-WEIGHT SALIENT FOREGROUND DETECTION WITH ADAPTIVE MEMORY REQUIREMENT

Mauricio Casares, Senem Velipasalar

University of Nebraska-Lincoln, Dept. of Electrical Engineering, Lincoln, NE, 68588
mauricio.casares@huskers.unl.edu, velipasa@engr.unl.edu

ABSTRACT

Designing algorithms, which require less memory and consume less power, is very important for the portability to embedded smart cameras, which have limited resources. We present a light-weight and efficient algorithm for salient foreground detection that is highly robust against lighting variations and non-static backgrounds such as scenes with swaying trees. Contrary to traditional methods, memory requirement for the data saved for each pixel is very small in the proposed algorithm. Moreover, the total memory requirement is adaptive, and is decreased even more depending on the amount of *activity* in the scene. As opposed to existing methods, we treat each pixel differently based on its history. Instead of requiring the same amount of memory for every pixel, we allocate less memory for stable background pixels. The plot of the required memory at each frame also serves as a tool to find the video portions with high *activity*.

Index Terms— foreground detection, background subtraction, salient motion, light-weight algorithm, memory.

1. INTRODUCTION

Embedded smart cameras have limited processing power and memory. Thus, it is very important to design light-weight algorithms that require less memory for storage, and consume less power. This paper presents a light-weight and efficient algorithm for salient foreground detection that is highly robust against lighting variations and non-static backgrounds such as scenes with swaying trees, water fountains, rippling water effects and rain. The memory requirement of the proposed method is significantly less compared to the traditional methods and our previous work [1]. Moreover, this is achieved without sacrificing accuracy.

Many methods have been introduced for foreground object detection [2]-[13]. However, much less attention has been paid to the memory requirement and the portability of the algorithms to an embedded processor.

Adaptive Mixture of Gaussians (MoG), introduced by Stauffer and Grimson [11], is one of the most commonly used background subtraction (BGS) methods to model complex and non-static backgrounds. However, a few Gaussian distributions are usually not sufficient to accurately model backgrounds having fast variations. Zivkovic [13] proposed an improved adaptive MOG model to constantly update the parameters of a Gaussian mixture and to simultaneously select the appropriate number of components for each pixel.

This work has been funded by NSF grant CNS0834753.

Kim et al. [6] proposed an algorithm for background modeling, where sample background values at each pixel are quantized into codebooks during training. This algorithm performs well when background is non-static. However, its performance on different video sequences is dependent on the choice of several threshold values.

In tracking applications, we are interested in *salient* motion. Thus, we need to handle the challenging cases of lighting variations and non-static backgrounds, and separate those from the salient motion regions. This increases the algorithm complexity, and thus memory requirements. For instance, in the case of adaptive mixture of Gaussians [11], three to five Gaussian distributions are saved for each pixel. Each distribution is represented by its mean and variance, and these values are saved as floats. In the codebook-based method [6], each codeword for each pixel has nine entries. Some of these entries need to be floats, and on the average 6.5 codewords are needed for a pixel. In our previous work [1], we presented a method for which the memory requirement is significantly less compared to the state-of-the-art algorithms. Yet, the same amount of memory is allocated for every pixel.

In this paper, we present a light-weight salient foreground detection method, wherein the total memory requirement per frame is adaptive. The total memory requirement is decreased even more depending on the amount of activity in the scene. As opposed to existing methods, we treat each pixel differently based on their histories. Instead of requiring the same amount of memory for every pixel, we allocate much less memory for stable background pixels. Thus, the memory requirement of the proposed method is significantly less compared to the traditional methods and our previous work [1]. As in our previous work, the algorithm presented here differentiates between salient and non-salient motion based on the reliability or unreliability of a pixel's location, and by considering neighborhood information. The concept of reliability will be summarized below. The background model is selectively updated with an automatically adaptive rate, thus can adapt to rapid changes as well. For instance, if a location is deduced to be reliable, a higher update rate is used, i. e. this location is incorporated to the background faster. Also, less memory is used for these locations. The algorithm gives very reliable results with gray level images.

In the proposed method, the memory requirement at each frame is adapted according to the amount of activity in the scene at that frame. For instance, if the static background is observed for N frames, then the minimum possible memory will be allocated for each pixel during these N frames. If a car enters the scene, then more memory will be allocated for those pixels that are affected. Thus, if we plot the total memory requirement versus the frame number, the changes and peaks in this plot will indicate the portions of the video with *activity*. Thus, this plot can serve as a tool for activity summary. We will revisit this important functionality in Section 3, where we will also present the foreground detection results obtained with different challenging sequences, and compare them with the results of different BGS methods.

2. THE PROPOSED METHOD

The details of the proposed algorithm will be explained by referring to the pseudo-code provided in Table 1. In the pseudo-code, M and M_{prev} denote the current and previous background models, respectively. $s(i, j)$ is a binary variable, and denotes the state of a pixel at location (i, j). It is 1 when the pixel is classified as foreground, or vice versa. $m(i, j)$ is another binary variable that determines the amount of memory to use. $m(i, j)$ is set to be 1 if the state of the pixel has not changed in the last 50 frames. This indicates that this pixel location is stable, and can be allocated less memory. The counter $h(i, j)$ holds the number of times a pixel changes its state $s(i, j)$ in the last 100 frames, i. e. $h(i, j)$ keeps the number of times a pixel's state changes from 0 to 1, or vice versa. $h(i, j)$ is used to determine the *reliability* or *unreliability* of a pixel's location. The motivation is that the lower the value of $h(i, j)$, the more stable and *reliable* that pixel location is.

An important point to note about the computation of $h(i, j)$ is the following: At any frame t, we want the number of changes in a pixel's state in the last 100 frames, and this requires to save the frame number for each change of state. For locations with non-salient motion, this requires an array with a potentially high dimension for each pixel. Instead, we divide the 100-frame window into 4 intervals, and keep a counter $CC_k, k \in \{1, 2, 3, 4\}$, for each interval for pixel (i, j). A temporary variable p keeps the number of changes in each 25-frame interval. At every k^{th} 25-frame interval, CC_k is replaced by p, and then p is set to 0. This avoids saving the instances of changes. Then $h(i, j) = CC_1 + \ldots + CC_4$.

The background model M is built by using a temporal difference method [1]. In order to detect slow motions or stopping objects, a weighted accumulation, I_t^{ac}, is used for temporal difference. At pixel location (i, j), I_t^{ac} is defined as:

$$I_t^{ac}(i, j) = (1 - w_{ac})I_{t-1}^{ac}(i, j) + w_{ac}|I_t(i, j) - I_{t-1}(i, j)|$$

where t is the current frame number, I_t is the current image frame, $w_{ac} = 0.5$, and I_0^{ac} is set to be an empty image. At the beginning $M(i, j) = -1$ for every pixel location (i, j). $M(i, j)$ is set to be $I_t(i, j)$, if $I_t^{ac}(i, j) < T$, where T is a difference threshold. Thus, as moving objects that exist in the scene change their location, the M will gradually be filled. If M is not complete, the difference image (I_{diff}) is set to be I_t^{ac}. If the M is complete, then $I_{diff} = I_t^{md}$, where $I_t^{md} = |I_t - M|$.

T, seen in Table 1, is a difference threshold. When the model M is not complete, and I_{diff} is based on temporal difference method $T = T_d = 15$ is used. When the model M is complete, and I_{diff} is set to be I_t^{md}, then $T = T_m = 25$ is employed. The same values have been used in the experiments for all the video sequences. Since temporal difference is based on consecutive frames, and tends to give smaller differences, T_d has a smaller value than T_m.

2.1. Adaptive memory allocation

As described above, $h(i, j)$ is determined by dividing a 100-frame window into 4 intervals, and keeping a counter $CC_k, k \in \{1, 2, 3, 4\}$, for each interval for pixel (i, j). For each CC_k, 1 byte is allocated. Rather than saving many values for each pixel location, such as averages for three color values, multiple Gaussian distribution means and variances, multiple codewords with multiple entries, we use CC_k to form a very compact summary of a pixel's history. Let $h_{50}(i, j) = CC_3 + CC_4$ be the number of changes in the pixel's state in the last 50 frames. Every 25 frames, we check the value of $h_{50}(i, j)$. If $h_{50}(i, j) \leq 1$, it means that the state of this pixel has not changed during the last 50 frames, i. e. this location is very reliable and stable. If the current state of this location is 0, i. e. it is a background pixel, $m(i, j)$ will be set to 1, and all CC_k values, each of which was allocated 1 byte, will be deallocated. The reasoning is the following. Since the pixel has been reliable and stable in its most recent history, we do not need to allocate extra memory to check reliability until the pixel's state changes again.

If $m(i, j) = 1$ for a pixel, and the pixel's state changes to foreground at frame t, then $m(i, j)$ is set back to zero, and $1-$byte memory is allocated for each CC_k. The reason is that this pixel is not a stable background location anymore, and counting of the number of changes in its state starts again. We perform the memory checks every 25 frames so that each CC_k corresponds to a complete 25-frame interval.

The comparison of the memory requirements of this adaptive method, and our previous work is provided in Section 3.

2.2. Adaptive Model Update

The update of the background model M is performed in a selective way, and with an automatically adaptive rate. The motivation is that when a pixel's location is deduced to be consistently reliable, then the value at that location is incorporated into the background model with a higher weight. Thus,

Table 1. Salient foreground detection algorithm

```
Set $M(i,j) = -1$ and $s(i,j) = 0$ for all $i, j$
Set $I_1$ = first frame, $I_t^{md}(i,j) = -1$ for all $i, j$
for every frame $t > 1$
    Set $I_t = t^{th}$ frame and set $I_{outp}$ all zeros
    if $\exists\ i, j$ for which $M(i,j) = -1$
        compute $I_t^{ac}$, set $I_{diff} = I_t^{ac}, T = T_d$
    else
        set $model\_complete = true$
        compute $I_t^{md}$, set $I_{diff} = I_t^{md}, T = T_m$
    for all $i, j$
        if $I_{diff} \geq T$
            if $s(i,j) = 0$
                update counters and set $s(i,j) = 1$
        else
            if $s(i,j) = 1$
                update counters and set $s(i,j) = 0$
            if $M(i,j) = -1$
                $M(i,j) = I_t(i,j)$
        Update $h(i,j)$
        if $model\_complete = true$
            if $I_t^{md}(i,j) > T$
                if $h(i,j) < T_p$
                    $s(i,j) = 1; m(i,j) = 0;$
                else
                    Set $neighb(i,j)$ to be $3 \times 3$ neighb. of $h(i,j)$
                    if N $> 0.7(2w+1)^2$ (details in [1])
                        $s(i,j) = 1; m(i,j) = 0;$
                    else
                        $s(i,j) = 0$
                        $M(i,j) = \alpha I_t(i,j) + (1-\alpha) M_{prev}(i,j)$
            else
                if t is a multiple of 25
                    if $h_{50}(i,j) \leq 1$
                        $m(i,j) = 1; h(i,j) = 0;$
                    $M(i,j) = 0.5 \times I_t(i,j) + 0.5 \times M_{prev}(i,j)$
                else
                    $M(i,j) = \alpha I_t(i,j) + (1-\alpha) M_{prev}(i,j)$
    Set $I_{prev}^{ac} = I_t^{ac}$ , $M_{prev} = M$
return $s(i,j)$
```

if $s(i,j) = 0$ and $m(i,j) = 1$, then this pixel will be incorporated to the background model with 50% weight.

As opposed to traditional model-based BGS approaches, in the proposed scheme satisfying $I_t^{md}\ (i,j) > T$ is not enough for the pixel location (i,j) to be classified as foreground. Instead, reliability constraints are employed to differentiate between salient and non-salient motion. A pixel location satisfying $I_t^{md}(i,j) > T$ is classified as foreground, and its state is set to 1 only if its counter $h(i,j)$ satisfies $h(i,j) < T_p$, where $T_p = 15$ is the percentage threshold. The reasoning is that if $h(i,j) < 15$, then it means that the status of the pixel at this location changed less than 15% of the time during the last 100 frames making this location a reliable one. In other words, this location is not likely to be in a non-salient motion region. Thus the intensity difference greater than T is caused by a salient motion with high probability.

3. EXPERIMENTAL RESULTS

We first compared the proposed method with our previous work in terms of the total memory requirement for each frame. The plot of the memory requirement (in bytes) versus frame number is displayed in Fig. 1, where each frame is 240x320. The flat part at the beginning corresponds to the model building period, where each pixel is allocated the same amount of memory (1 byte for the intensity value, 1 byte for each CC_k, 1 bit for the state variable). The total memory requirement per frame during this period is equal to the memory requirement of our previous work (393.6 Kb). After the background model is built, the memory requirement of the proposed method drops significantly to 100 Kb, since the memory allocation is adapted according to the activity in the scene. Between frames 2100 to 2300, a person walks in front of the camera, affecting many pixels. This increases the allocated memory and causes the peak in Fig. 1.

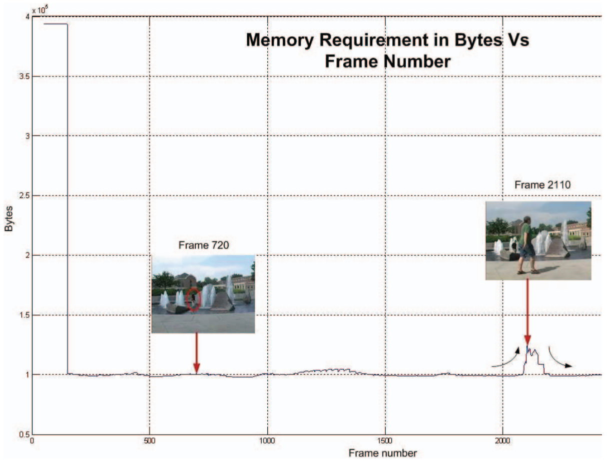
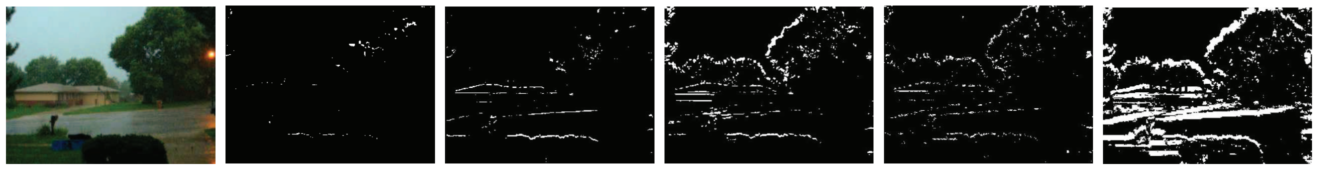


Fig. 1. Total memory requirement vs. frame number plot also serves as a tool for activity summary.

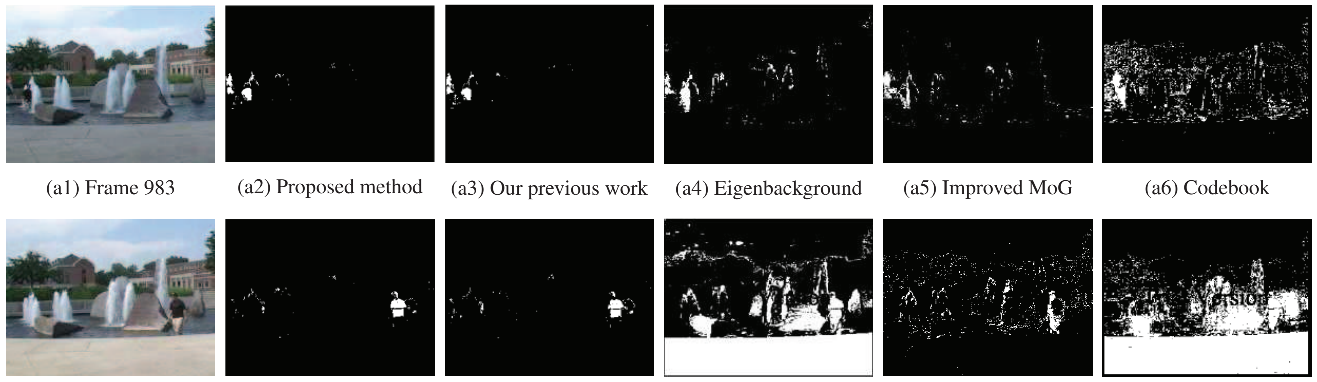
We tested the proposed method on several video sequences, and compared it with six other BGS methods including our previous work. All the displayed outputs images are obtained *without* applying any morphological or post-processing operations. All the results of our algorithm were obtained by using the same threshold values for all videos. Specifically, $T_d = 15, T_m = 25, T_p = 15$, and $\alpha = 0.05$. Figures 2 and 3 show the outputs obtained with the proposed method and different state-of-the-art BGS algorithms. The proposed algorithm performs better in terms of eliminating non-salient motions caused by wind and rain.

Figure 3 was obtained from a video of a scene with a fountain, where the water level goes up and down. Moreover, during the video, lighting changes due to moving clouds, as seen in Figures 3(a1) and 3(b1). The proposed method performs better by having the least amount of noisy pixels and good detection, and requiring significantly less memory at the same time. We also tried the original MoG method and the method by Horprasert et al. [4] on all the videos, but only displayed the outputs of the algorithms giving the better results. While



(a) Frame 296 (b) Proposed method (c) Our previous work (d) Eigenbackground (e) Improved MoG (f) CodeBook

Fig. 2. Foreground detection results of several different algorithms on a challenging video with strong wind and rain.



(a1) Frame 983 (a2) Proposed method (a3) Our previous work (a4) Eigenbackground (a5) Improved MoG (a6) Codebook

(b1) Frame 1800 (b2) Proposed method (b3) Our previous work (b4) Eigenbackground (b5) Improved MoG (b6) Codebook

Fig. 3. Comparison of foreground detection results of several different algorithms on a video of a fountain.

the other algorithms were run with color images, the results of our method were obtained with gray-level images.

4. CONCLUSIONS AND FUTURE WORK

We presented a light-weight salient foreground detection algorithm with adaptive memory requirement. Instead of requiring the same amount of memory for every pixel, less memory is allocated for stable background pixels. Thus, a significant decrease is provided in the total memory requirement per frame. If the total memory requirement versus the frame number is plotted, the changes and peaks in this plot will indicate the portions of the video with *activity*. Thus, this plot can serve as a tool for activity summary. The algorithm achieves the reduction in the memory requirement without sacrificing accuracy. The algorithm can run with gray-level images and handle challenging non-static backgrounds.

5. REFERENCES

[1] M. Casares and S. Velipasalar, "Light-weight salient foreground detection for embedded smart cameras," *Proc. of ACM/IEEE Int'l Conf. on Distributed Smart Cameras*, 2008.

[2] A. Elgammal, D. Harwood and L. Davis, "Non-parametric model for background subtraction," *Proc. of 6th European Conf. on Computer Vision*, pp. 751–767, June/July 2000.

[3] 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*, vol. 22, no. 8, pp. 809-830, August 2000.

[4] T. Horprasert, D. Harwood and L. Davis, "A statistical approach for real-time robust background subtraction and shadow detection," *Proc. of IEEE ICCV Frame-Rate Workshop*, pp. 1–19, 1999.

[5] T. Kanade, R. T. Collins, A. J. Lipton, P. Burt and L. Wixson, "Advances in cooperative multi-sensor video surveillance," Proc. of DARPA Image Understanding Workshop, pp. 3–24, Monterey, CA, November 1998.

[6] K. Kim, T. H. Chalidabhongse, D. Harwood and L. Davis, "Real-time foreground-background segmentation using codebook model," *Real-time Imaging*, v. 11, no. 3, pp. 172–185, June 2005.

[7] Nam T. Nguyen, S. Venkatesh, G. West and Hung H. Bui, "Multiple camera coordination in a surveillance system," *ACTA Automatica Sinica*, vol. 29 (3), pp. 408-422, 2003.

[8] N. Oliver, B. Rosario, and A. Pentland, "A bayesian computer vision system for modeling human interactions," *IEEE Trans. on PAMI*, pp. 831-834, 2000.

[9] I. Pavlidis, V. Morellas, P. Tsiamyrtzis and S. Harp, "Urban surveillance systems: from the laboratory to the commercial world," *Proceedings of the IEEE*, vol. 89, no. 10, pp. 1478–1497, October 2001.

[10] P. L. Rosin and T. Ellis, "Image difference threshold strategies and shadow detection," *Proc. of British Machine Vision Conf.*, pp. 347–356, 1995.

[11] C. Stauffer and W. E. L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Trans. on PAMI*, vol. 22, no. 8, pp. 747–757, August 2000.

[12] C. R. Wren, A. Azarbayejani, T. Darrell, and A.P.Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780-785, July 1997.

[13] Z. Zivkovic, "Improved adaptive Gausian mixture model for background subtraction," *Proc. of the Int'l Conf. on Pattern Recognition*, pp. 28–31, 2004.