Speeded up Gaussian Mixture Model algorithm for background subtraction

Conference Paper · October 2011

DOI: 10.1109/AVSS.2011.6027356 · Source: IEEE Xplore

2 authors, including:

Bharadwaj Amrutur
Indian Institute of Science
156 PUBLICATIONS **1,450** CITATIONS

SEE PROFILE

8th IEEE International Conference on Advanced Video and Signal-Based Surveillance, 2011

Speeded up Gaussian Mixture Model Algorithm for Background Subtraction

Pushkar Gorur, Bharadwaj Amrutur
Department of Electrical Communication Engineering
Indian Institute of Science, Bangalore, India
{pushkar,amrutur}@ece.iisc.ernet.in

Abstract

Adaptive Gaussian Mixture Models (GMM) have been one of the most popular and successful approaches to perform foreground segmentation on multimodal background scenes. However, the good accuracy of the GMM algorithm comes at a high computational cost. An improved GMM technique was proposed by Zivkovic to reduce computational cost by minimizing the number of modes adaptively. In this paper, we propose a modification to his adaptive GMM algorithm that further reduces execution time by replacing expensive floating point computations with low cost integer operations. To maintain accuracy, we derive a heuristic that computes periodic floating point updates for the GMM weight parameter using the value of an integer counter. Experiments show speedups in the range of 1.33 - 1.44 on standard video datasets where a large fraction of pixels are multimodal.

1. Introduction

Background subtraction is often the first step in static camera video surveillance applications. It reduces the computation required by the downstream stages of the surveillance pipeline which usually comprises of object detection and tracking. Consequently, it constitutes the most active/resource demanding stage of the surveillance pipeline since it processes each incoming pixel in the video stream.

Background subtraction also reduces the search space in the video frame for the object detection unit by filtering out the uninteresting background. Hence an accurate background subtraction algorithm is essential to reduce the overall false positive rate of the surveillance system. Shaking trees, foliage, sunlight intensity changes due to active cloud motion, rain have been the main sources for reduced accuracy of simple background subtraction algorithms. The Gaussian mixture model(GMM) scheme proposed by Stauffer and Grimson [7] has been one of the most successful techniques that works well in such uncontrolled outdoor environments. However the original GMM algorithm [7] suffered from slow learning rates during the initial phase. KaewTraKulPong and Bowden [3] corrected this using a two stage learning scheme where the GMM is updated initially using the sufficient statistics based equations and is later switched to a 'L-recent window' version. Lee [4] further improved upon this by using a modified schedule that gradually switches between the two update modes.

The good accuracy of the GMM approach comes at the cost of significantly high computation and memory bandwidth requirements. Benezeth *et al*. [2] found that the GMM algorithm with 3 modes is about 3.7 times slower compared to the single Gaussian scheme. Zivkovic [8] described a significant improvement to reduce the computation time and memory bandwidth. He formulated a Bayesian approach to select the required number of Gaussian modes for each pixel in the scene. In scenes with static background (traffic sequence in their paper), this approach assigns a single mode Gaussian to model most of the pixels which helps to reduce average processing time by 32%. However in the outdoor video (trees sequence), results show only a 2% improvement since a significantly large portion of the scene requires a multimodal model.

Although real time performance of the adaptive GMM scheme has been demonstrated on native PC's, an increasing demand to move the analytics onto the camera itself requires embedded platforms with low compute resources to support the algorithms. It is also increasingly common to have a large number of cameras streaming video into a central workstation in networked surveillance systems. The growth in the adoption of high resolution, wide FOV(Field of view) surveillance cameras is also increasing the computational requirements of video analytics algorithms. Hence reducing computation time continues to be important even with improved compute systems.

In this paper, we propose an orthogonal approach that provides computation time reduction by minimizing floating point computations. We also combine the fast learning of [4] with the automatic selection of number of modes in [8] to obtain a highly efficient and accurate scheme.

The paper is organized as follows: In the next section,

978-1-4577-0845-9/11/$26.00

we review the modification proposed by D.S.Lee followed by a very brief description of the improved AGMM algorithm proposed by Zivkovic. We refer to [4] & [8] for a detailed discussion of the algorithms. We also present our proposed improvisation to the GMM algorithm that significantly reduces the computation time. Detailed experimental results of the proposed algorithm are discussed in section 3.

2. Gaussian mixture model

2.1. Adaptive Mixture Learning with fast convergence

Each pixel in a frame is modeled as a stochastic process. The parameters of a Gaussian mixture model (usually with 3 modes) are estimated using an online version of the EM algorithm. To prevent the foreground pixels from corrupting the background model, [7] proposed to use modes with low weights to model the foreground. The description of the algorithm is given below:

The modes of the GMM are arranged in decreasing order of their weights. A predefined fraction of the weights is used to determine the modes that model the background. This favors modes with higher weight to be selected as the background. A match of an incoming pixel to any of the modes is defined to occur if the Mahalanobis distance from the pixel is less than a predefined threshold T_σ. If the match occurs on one of the background modes, the pixel is labeled as background, else it is classified as foreground. The following update equations are applied for the parameters of the Gaussian mode 'k' with the highest weight that matched the incoming pixel $x(t)$:

$$w_k(t) = (1-\alpha)w_k(t-1) + \alpha \quad (1)$$

$$\mu_k(t) = (1-\eta_k)\mu_k(t-1) + \eta_k x(t) \quad (2)$$

$$\sigma_k^2(t) = (1-\eta_k)\sigma_k^2(t-1) + \eta_k(x(t)-\mu_k(t-1))^2 \quad (3)$$

where η_k is the adaptive learning rate given by:

$$\eta_k = \frac{1-\alpha}{c_k} + \alpha \quad (4)$$

c_k is a counter which is maintained independently for each mode. Its value is initialized to 1 for a new mode and is incremented whenever a match with an incoming pixel occurs. η_k is a parameter that controls the learning rate of the modes.

As can be observed from Eq. (4), the learning rate is initially set to match the sufficient statistics based update. As time progresses, it converges to a L-recent window based update mode with a fixed learning rate of α. The weight for the remaining modes is updated using Eq. 5.

$$w_k(t) = (1-\alpha)w_k(t-1) \quad (5)$$

If none of the modes match, then the mode with the least weight is replaced with a new mode having low initial weight and large variance.

2.2. Automatic selection of number of components

In the GMM algorithm described above, the weights of the Gaussian mixture represent the fraction of the data samples '$x(t)$' that belongs to the particular mode in the model. Defining n_m to represent the number of samples that belong to the m^{th} mode, the weights of the GMM can be considered to define a multinomial distribution for the n_m's. Instead of using the ML estimate that results in the original GMM update equation, Zivkovic used a Dirichlet prior with negative coefficients. This is done with an intention of accepting a class only if there is enough evidence from the data samples for the existence of the class. Solving for the MAP(Maximum a posteriori posterior) estimate, the final adaptive update Eqs. (1) & (5) are modified as follows:

$$w_k(t) = (1-\alpha)w_k(t-1) + \alpha - \alpha c_T \quad (6)$$

$$w_k(t) = (1-\alpha)w_k(t-1) - \alpha c_T \quad (7)$$

c_T is a parameter that represents the minimum fraction of samples required to support the existence of a mode (set to be equal to 0.01 in [8]). We need to normalize the weights after each update so that they add up to one. The modes whose weights become negative are discarded. New modes are initialized with mean set to be equal to the pixel values that didn't match any of the existing modes. The variance is initialized to a large value. The mean and the variance updates are similar to Eqs. (2) & (3) with η_k defined to be equal to $\alpha/w_k(t)$ instead of (4). This division by the weight significantly improves the learning rate compared to [7], however as Lee mentions in [4], η_k is unbounded and hence might lead to divergence.

2.3. Proposed Algorithm

From the description provided in sections 2.1 & 2.2 we list the main steps involved in the GMM algorithm as: (A) Sort the Gaussian modes (B) Match the pixel to the modes and (C) Update the parameters of the modes.

We also note 4 observations in [4] and [8] that suggest our modification:

1. The weights of the Gaussian modes change slowly with time constant of roughly $\approx 1/\alpha$ which is typically of the order of a few hundred frames
2. Set of Background modes also doesn't change rapidly since the weights change slowly
3. The mean and variance update Eqs. (2) & (3) are independent of the weight values

4. A newly formed mode takes a minimum of a few tens of cycles to be removed. For $c_T = 0.01$ in [8], it takes about the order of $1/c_T$ frames (100 frames) for a newly formed mode to be removed in the case where none of the pixels match that mode.

Based on the above observations, we propose to update the weights only once in T_w frames where T_w is a constant set to be equal to 16. The details for the choice of T_w is discussed in the results section. We refer to T_w as the 'weight update interval'. The set of modes that belong to the background are also determined only once in T_w frames. New modes are allowed to be created for all the frames, however we determine mode deletions only once in T_w frames. We refer to the cycle when we perform the true weight update as the 'fine update cycle'. To ensure that learning is unaffected, we need to perform accurate weight updates based on the values of the pixels in the past T_w frames. We enable this by using a low resolution (4 bit) integer counter to count the number of matches to a mode that occurs in the T_w frames. The counter values are used to perform an accurate update during the 'fine update cycle' using a heuristic derived below. Since we now update the floating point weight values and determine the set of background modes only during the 'fine update cycle' (once in T_w frames), the computational complexity is reduced. Expensive floating point computations during the remaining $T_w - 1$ cycles are replaced by simple integer increment operations. The heuristic for the weight update is described below.

Assume that all T_w consecutive pixel samples $x(t)$ matched the same mode 'k' with weight w_k. The cumulative weight update at the end of the T_w^{th} frame is:

$$w_k(t+T_w) = [[[w_k(t)(1-\alpha)+\alpha](1-\alpha)+\alpha]....] \quad (8)$$
$$\approx w_k(t)(1-\alpha)^{T_w} + T_w\alpha \quad (9)$$
$$\approx w_k(t)(1-T_w\alpha) + T_w\alpha \quad (10)$$

Similarly for the case where none of the T_w pixel samples $x(t)$ matched the mode 'k', the cumulative weight update at the end of the T_w^{th} frame is:

$$w_k(t+T_w) \approx w_k(t)\,(1-T_w\alpha) \quad (11)$$

Now, we propose to ignore the order in which the pixel samples $x(t)$ have arrived in the T_w frames. Hence for the case when N_k matches occur to a pixel mode 'k' in T_w frames, we can multiply the two Eqs. (10) & (11) with suitably modified count values to obtain the final heuristic in Eq. (12). Here N_k is equal to $weightCount_k$ which is the number of pixels in the T_w window that matched the k^{th} mode. We also append the αc_T term from [8] to enable dynamic selection of number of modes in the GMM.

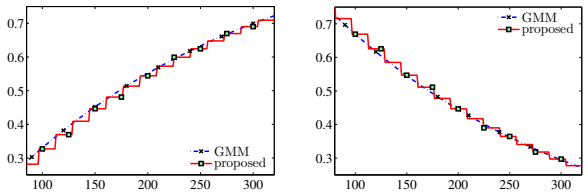
Algorithm 1: Proposed scheme for a single pixel x

Init: BG = { }, $numModes$ = 1, Reset $maxModeFlag$, $\forall i \in \{1..maxNumModes\}$
$\mu_i = \infty$, $\sigma_i = \sigma_{init}$, $w_i = \alpha$
Data: input pixel $x(t)$
while *New Data* $x(t)$ **do**
  for $i \in \{1....numModes\}$ **do**
    if $x(t)$ *matches mode* i **then**
      if $i \in BG$ **then**
        $x(t)$ is Background
      else
        $x(t)$ is Foreground
      $c_i \longleftarrow c_i$ + 1 (refers to c_i from D.S.Lee)
      Update μ_i, σ_i using Eqs. (2), (3) & (4)
      $weightCount_i \longleftarrow weightCount_i$ + 1
  if $\forall i \in \{1..numModes\}$, $x(t)$ *doesn't match mode* i **then**
    $x(t)$ is Foreground
    if $numModes < maxNumModes$ **then**
      $numModes \longleftarrow numModes$ + 1
      Initialize new mode j;
    else
      Replace mode j where j = arg min$_i\{w_i\}$
    $c_j \longleftarrow 1$, $weightCount_j \longleftarrow 1$
    $\mu_j \longleftarrow x(t)$, $\sigma_j \longleftarrow \sigma_{init}$
    if $numModes = maxNumModes$ **then**
      Set $maxModeFlag$

  Once in T$_w$ frames:
  if t *is a multiple of* T_w **then**
    if $maxModeFlag$ *is Set* **then**
      $w_i \longleftarrow \alpha$ where i = arg min$_i\{w_i\}$
      Reset $maxModeFlag$
    for $i \in \{1....numModes\}$ **do**
      Update w_i using Eq. (12)
      if $w_i < 0$ **then**
        delete mode
        $numModes \longleftarrow numModes$ - 1
    Normalize w
    Arrange modes in decreasing order of w_i's
    Determine Set of Background Modes, BG
    BG = $\{1....n_{BG}\}$ where n_{BG} = arg min$_b\{\sum_{k=1}^{b} w_k > T_{BG}\}$

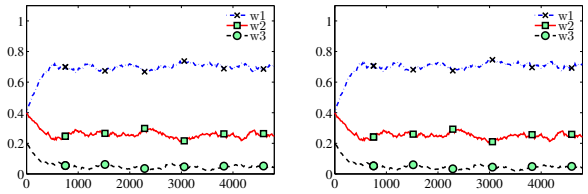
$$w_k(t+T_w) = [w_k(t)(1-N_k\alpha)+N_k\alpha][1-(T_w-N_k)\alpha] - T_w\alpha c_T \quad (12)$$

In Fig. 1, the weight update is plotted using the original GMM update equation and the proposed heuristic for the case where the weights are monotonically increasing and decreasing with T_w = 16. The heuristic is also shown applied to a realistic scenario in Fig. 2. Here data points generated from a synthetic distribution with mass function = [0.7, 0.25, 0.05] are used to update the weights using both the original GMM Eqs. (6) & (7) and the proposed heuristic Eq. (12). The initial weight is set arbitrarily to [0.4, 0.4 ,0.2]. We observe that the learning rate and weight values obtained using the proposed heuristic matches well with those obtained from the original GMM update equations.



(a) (b)

Figure 1: Weight update using proposed heuristic for a monotonically (a) increasing and (b) decreasing case. The weights are plotted on the y axis with respect to time



(a) (b)

Figure 2: Weight update using (a) original GMM update equations and (b) proposed heuristic for a GMM with weights=[0.7, 0.25, 0.05] and T_w=16. The weights are plotted on the y axis with respect to time

The complete pseudo code is described in Algorithm 1. Here $maxModeFlag$ is used to indicate that the mode with least weight was replaced during the T_w window. This is used during the fine update cycle to reset the true weights of that mode. BG is a set that contains the list of modes that belong to the Background model. This set is updated during the 'fine update cycle'. The integer weight counters are represented by $weightCount_i$'s. The 'fine update cycle' is staggered across pixels and in time such that only n_{pixels}/T_w recieve the fine update during each cycle (where n_{pixels} is the number of pixels in the frame). This ensures that the processing time is uniform across all the frames.

3. Experimental results

We initially describe experiments done to determine the optimum weight update interval T_w. Later, we discuss results of experiments performed to obtain KL divergence on 1-dimensional synthetic datasets showing good learning performance of the proposed scheme. Next, we present a quantitative evaluation of the proposed algorithm on a set of 10 standard videos (5 outdoor and 5 indoor): video4, 6 & 7 from [1], fountain, hall, lobby, shoppingMall, bootstrap & campus from [5] and HighwayI from [6]. Dataset [5] provided 20 frames of manually segmented foreground masks from each video set. VSSN 2006 provides foreground truth for all the frames. Groundtruth was generated manually for 10 randomly chosen frames in the HighwayI sequence. Precision-Recall curves for the proposed algorithm is compared with those obtained using [4] and [8].

We measure the frame rate on a Core i5 processor running at 2.53Ghz with 4GB of system memory. All the programs are single threaded and have been compiled in Release mode using Microsoft Visual C++. The following parameter values were found to work well on all the videos: $\alpha = 0.004$ & T_{BG} = 0.8. The $maxNummodes$ was set to 3 (two for the background and one for the foreground).

3.1. Weight update interval Experiment

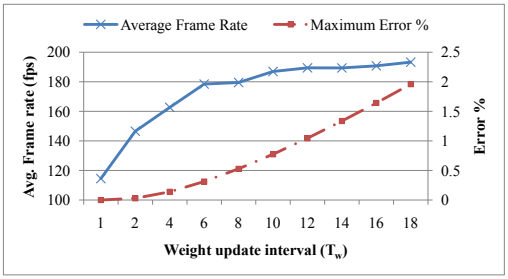
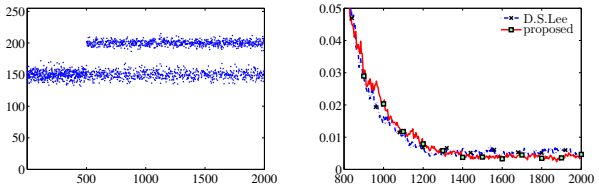


Figure 3: Frame rate (fps) and error % are plotted with respect to the weight update interval T_w

Fig. 3 shows the measured frame rate plotted as a function of T_w for the VSSN06 dataset videos using the heuristic. We observe that the speedup saturates for T_w in the range of $\approx$ 12 - 18. The maximum error % of the proposed heuristic is also plotted where the error is defined as the maximum difference between the weights obtained using 'per cycle update' Eqs. (6) & (7) and the weight obtained using the proposed heuristic Eq. 12 at the end of T_w frames. All possible combinations of matches during the T_w frames are considered and the maximum deviation is plotted. The initial weight at the beginning of the T_w frames is set to a realistic value w_{init} = [0.7, 0.25, 0.05]. We can observe that the error increases linearly as T_w is increased.

Since speedup saturates for T_w in the range of $\approx$ 12 - 18, we choose T_w to be 16. Choosing a higher T_w increases the error without any benefit. On a dedicated hardware system, a weight update interval of 16 results in compact 4 bit counters for the coarse weight updates. We find that the small error in weight doesn't have any impact on the accuracy in real dataset videos. Detailed accuracy data is described below for the chosen weight update interval of 16.

3.2. Adaptive Mixture Learning Experiment



(a) (b)

Figure 4: (a) Synthetic distribution based on commonly observed surveillance videos (b) KL divergence achieved by the proposed and the original method [4]

The accuracy of the proposed method is first validated on one dimensional synthetic data. Fig. 4a shows a typical pixel intensity distribution (plotted against frame count) observed in surveillance videos. Here, a pixel which was initially unimodal (*e.g.* pixel belongs to the 'sky') changes to a multimodal process (*e.g.* wind causes tree leaves to vacillate on a static 'sky' background). Fig. 4b shows the KL divergence achieved by the original update equations in [4] and by the proposed method during the phase where the model is learning the parameters for the new mode. We find the learning achieved by the proposed method to be very similar to that obtained using the original update equations. The divergence has been computed using Monte Carlo sampling averaged over 5 datasets. Similar experiments performed on slowly varying illumination models showed that the accuracies of the proposed method matched well with the original scheme in [4].

3.3. Background subtraction experiment

The precision-recall curves for the 10 videos listed in section 3 have been determined. We observed that the accuracy of the proposed algorithm matches the accuracy of the GMM formulations of Lee [4] and Zivkovic [8] in all the videos. The average precision is plotted against the recall rate in Fig. 5 showing no degradation of accuracy with the proposed scheme. Since a false negative or a 'miss' is undesirable in surveillance applications, we only show recall rates varied from 0.65 to 0.95. However, we verified that the proposed scheme doesn't impact accuracy at lower recall rates as well.

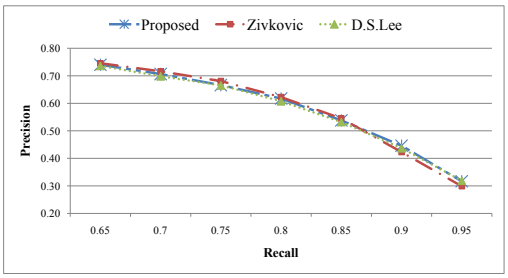
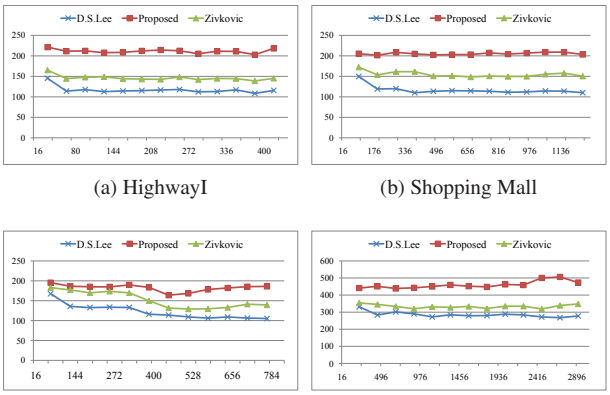


Figure 5: Average precision-recall curves obtained using proposed scheme, [4] & [8] for the 10 dataset videos

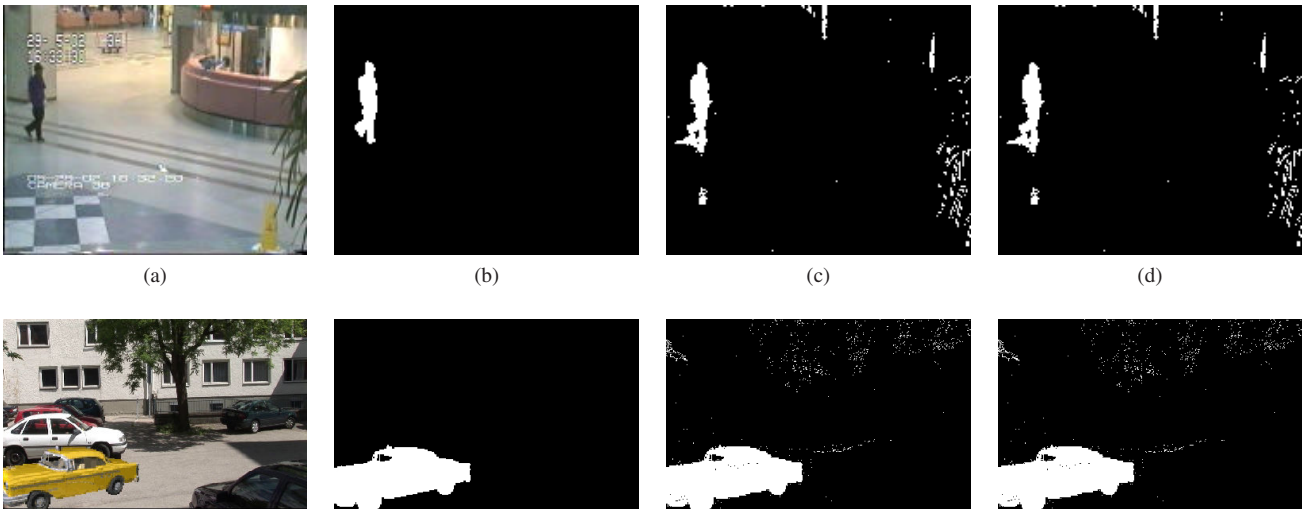


(a) HighwayI (b) Shopping Mall

(c) video4 (d) bootstrap

Figure 6: Instantaneous frame rates plotted against frame count using proposed scheme, [4] & [8]

Fig. 6 shows the frame rate of the different methods averaged over 5 trials with $T_\sigma = 3$. The average frame rate computed as the reciprocal of the average computation time is listed in Table 1. We find that the proposed scheme provides significant speedup for the case where there are multiple modes required to model a significant fraction of the scene. In Figs. 6a, 6b & 6d, we observe high speedups since frequent foreground motion results in continuous creation of new modes. Similarly, large background motion in the 'campus' sequence causes a significant fraction of pixels to require a multimodal model. Hence, the proposed method provides speedup in this sequence as well. In Fig. 6c, we notice that the initial speedup is low. This is due to the relatively static scene during the initial phase of video4. Since a significant fraction of the scene requires only a single mode, Zivkovic's scheme itself provides high speedup dominating the proposed method. However, we observe that the speedup of the proposed method over Zivkovic's scheme improves beyond frame 400 since the number of multimodal pixels increase (due to shaking leaves and appearance of foreground objects).



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 7: Detection results on the Hall [5] and video7 [1] dataset videos. (a) & (e) are original images from the Hall and video7 datasets respectively. (b) & (f) are the corresponding Ground truth images. (c) & (g) are the segmentation masks obtained using Lee [4]. (d) & (h) are the segmentation masks obtained using the proposed scheme

Table 1: Average frame rate using proposed scheme, [4] & [8]. Average speedup obtained using proposed scheme measured over [8] (Zivkovic)

Average frame rates (fps)				
Dataset	D.S.Lee	Zivkovic	Proposed	Avg. Speedup
HighwayI	116	145	211	1.44
Campus	287	351	432	1.23
Hall	265	326	401	1.22
Lobby	307	375	440	1.17
Mall	115	153	204	1.33
Fountain	342	423	457	1.08
Bootstrap	291	333	458	1.37
video4	119	149	182	1.22
video6	116	143	185	1.28
video7	137	174	191	1.09

4. Conclusions

The computational complexity of modeling the background pixels using adaptive GMM proposed in [8] can be significantly reduced for highly active pixels by our proposed scheme of windowed weight updates. This method reduces processing time without affecting segmentation accuracy. Experimental results shows a speedup of upto 44% in scenes where a large fraction of the pixels require multimodal Gaussian models. The proposed modifications are also quite suitable for a hardware implementation.

References

[1] http://imagelab.ing.unimore.it/vssn06/. 4, 6

[2] Y. Benezeth, P. M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger. Comparative study of background subtraction algorithms. *J. Elec. Imaging*, 19(3):033003+, 2010. 1

[3] P. Kaewtrakulpong and R. Bowden. An Improved Adaptive Background Mixture Model for Realtime Tracking with Shadow Detection. In *Proc. 2nd European Workshop on Advanced Video Based Surveillance Systems*. Kluwer Academic Publishers, September 2001. 1

[4] D.-S. Lee. Effective gaussian mixture learning for video background subtraction. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27:827–832, 2005. 1, 2, 4, 5, 6

[5] L. Li, W. Huang, I. Y. H. Gu, and Q. Tian. Statistical modeling of complex backgrounds for foreground object detection. *IEEE Trans. on Image Processing*, 13(11):1459–1472, 2004. 4, 6

[6] A. Prati, I. Mikic, M. M. Trivedi, and R. Cucchiara. Detecting moving shadows: algorithms and evaluation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 25(7):918–923, June 2003. 4

[7] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. *IEEE Conf. on Computer Vision and Pattern Recognition*, 2:2246, 1999. 1, 2

[8] Z. Zivkovic. Improved adaptive gaussian mixture model for background subtraction. *Int'l Conf. on Pattern Recognition*, 2:28–31, 2004. 1, 2, 3, 4, 5, 6