An Analysis of Background Subtraction on Embedded Platform Based on Synthetic Dataset

I. Iszaidy1, R. Ngadiran2, R.B Ahmad3, N. Ramli4, M.I. Jais5, V. Vijayasarveswari6

School of Computer and Communication Engineering
Universiti Malaysia Perlis
Pauh Putra, Perlis, Malaysia

iszaidy@unimap.edu.my1, ruzelita@unimap.edu.my2, badli@unimap.edu.my3, nuraminah@unimap.edu.my4, mohdilman@unimap.edu.my5, vijaya@unimap.edu.my6

Abstract. Background subtraction is a preliminary technique used for video surveillance and a widely used approach for indexing moving objects. A range of algorithms have been introduced over the years, and it might be hard to implement the algorithms on the embedded platform because the embedded platform comes up with limited processing power. The goal of this study is to provide a comparative analysis of available background subtraction algorithms on the embedded platform: Raspberry Pi. The algorithms are compared based on the segmentation quality (precision, recall, and f-measure) and hardware performance (CPU usage and time consumption) using a synthetic video from BMC Dataset with different challenges like normal weather, sunny, cloudy, foggy and windy weather.

1. Introduction

Today’s digital cameras capture images and a smart camera capture high-level descriptions of a scene and analyse what they see. These devices could support a wide variety of applications, including human and animal detection, video surveillance, motion analysis, and facial identification.

In this context, video surveillance systems provide the basic functionality needed to transform the security paradigm from investigation to the identification. Video surveillance requires reliable background subtraction for further operation, and it can be used to collect important information data such as object localization, tracking, and recognition.

However, the technological evolution of background subtraction has now faced with important issues such as the illuminating problem, camera jitter, low frame rate video, and overlapping problem [1], [2].

An embedded vision system is a combination of an image sensor with a surveillance system in an embedded board to work as a motion detection tool. The software design considers the start of the process from image grabbing up to identifying a moving object. The image sensor is part of the hardware components of the embedded vision system [3]–[5].

Identifying moving objects using an embedded vision system is a fundamental and critical task. Background subtraction algorithms are commonly used to detect and track moving objects. Each frame is compared to the background frame, and the differences between the two frames are the moving object.
Then, the differences are further processed for object localization and tracking purposes [6]. Although many background subtraction algorithms have been used as the first step in many computers visions processing, the problem of identifying objects in a complex environment with limited computer resources such as the embedded board is still far from being completely solved.

The recent background subtraction algorithms are accurate because it used a complex model which required high computational resources [7]. For a background subtraction to be useful in an embedded system, it must be both accurate and computationally efficient because it has limited computational resources. In this paper, the performance analysis of the recent background subtraction algorithm is investigated on an embedded board with multiple events of synthetic video datasets.

This paper is structured as follows: In Section II, a description of the recent background subtraction algorithms is presented. Section III presents the experimentation setup for the background subtraction to be analysed in detail. In Section IV, the result and discussion are presented, while in Section V, the conclusion part is arranged.

2. Theory of Background Subtraction

2.1. Frame Difference
Frame difference is a simple, fast, and basic background modeling technique, which only uses a single previous frame as the reference frame. It is a commonly used approach for detecting moving objects [8]. The disadvantage of this technique is it could not identify the interior pixel of a large, uniformly colored moving object. This technique is fast because it only involves two differenced image subtraction for the foreground mask [9], [10].

2.2. Adaptive Background Learning
The adaptive background learning (ABGL) technique is also one of the basic background modeling techniques based on the frame difference method [11]. It presents equations that are used to model each frame and based on this background model, the background subtraction of the frames is done. In simple terms, the algorithm generates the current background based on the segmentation results produced from the frame difference method.

2.3. Mixture of Gaussians
The Mixture of Gaussians (MOG) model is the most widely used algorithms for background subtraction and usually used for comparison purposes [12]. The MOG approach models each pixel history as a cluster of Gaussian-type distributions and get through an online approximation to update its parameters. Based on this step, the background is determined as the expected value of the distribution corresponding to the most populated cluster [13]. This methodology is significantly enhanced on the grounds of performance by allowing recursive equations to adaptively update the parameters of the Gaussian model [14]. But the MOG approach cannot handle fast variations with accuracy using a few Gaussians. Therefore this approach has problems for sensitive detection (changes in illumination condition, background trembling, and so on) of the foreground region [15]. The high computational complexity of the Gaussian based method acts as a bottleneck in the real-time implementation in a surveillance system with low computational power [7], [16].

2.4. Pixel Based Adaptive Segmenter
Hofmann et al. [17] is a recent study that has yielded the best subtraction results. A new background subtraction with feedback is introduced and named as a pixel-based adaptive segmenter (PBAS). PBAS formulates the foreground decision by using a history of recently observed pixel values as the background model. Although pixel-based background modelling methods can effectively obtain detailed shapes of foreground objects, they are easily affected by noise, dynamic backgrounds, and illumination changes [18]. Moreover, Nurhadiyatna, Hardjono, & Wibisono (2013) mentioned that the PBAS method demanded a high level of power consumption.
2.5. Codebook
Another popular algorithm for background subtraction and frequently used for the multimodal background is the so-called Codebook [12]. The Codebook is the most sensitive colour-based background subtraction method and can be applied both indoor and outdoor scenes. Based on a training sequence and without probabilities evaluation, the method allocates to each background pixel a series of key colour values (called code-words) gathered in a codebook without making parametric assumptions. These codewords will take over colour in a certain period. For instance, a pixel in a stable area may be summarised by only one codeword, whereas three values could summarise a pixel located over a tree shaken by the wind; green for the foliage, blue for the sky, brown for the bark. With the assumption that shadows correspond to the brightness shifts and real foreground moving objects to chroma shifts, the original version of the method was designed to eliminate false positives caused by illumination [20], [21].

3. Experimentation
The hardware evaluation experiments were conducted on a Raspberry Pi 2 board, and the background subtraction algorithm investigation is focused on Recall, Precision, and F-measure based on provided datasets.

Since the main objective is to evaluate the result of the background subtraction technique correctly, ground truth is necessary for all videos forming the database allowing the evaluation of true positive (TP), true negative (TN), false positive (FP) and false negative (FN) numbers [7], [11], [18], [21]–[24].

Precision is the measurement of positive predictive value, and recall is a measurement of background subtraction sensitivity based on the value of TP, TN, FP, and FN. The equation for precision is defined by (1) and (2) is for recall.

\[
\text{Precision} = \frac{TP}{TP + FP} \quad (1)
\]
\[
\text{Recall} = \frac{TP}{TP + FN} \quad (2)
\]

From the above equation, the best algorithm is the one that can produce all together with a small number of FP and FN. These will produce a high precision and recall value. Based on the value of precision and recall, the harmonic mean (F-measure) is defined by (3).

\[
F = \frac{2(\text{Recall} + \text{Precision})}{(\text{Recall} + \text{Precision})} \quad (3)
\]

| Method                     | Parameter Settings (Author)                      |
|----------------------------|--------------------------------------------------|
| Frame Difference           | \( \tau = 15 \) (OpenCV)                        |
| Adaptive Background Learning (ABGL) | \( \tau =15, \alpha = 0.5 \) (OpenCV)         |
| Mixture of Gaussian (MOG)  | \( \tau = 5, \alpha = 0.01 \) [14]            |
| Pixel Based Adaptive Segmenter (PBAS) | Original default parameter from [17] |
| CodeBook                  | Codewords and codebooks framework from [20]    |
In this subsection, all algorithms with the determined parameter values are used for comparison based on a dataset obtained from BMC [25], where the same event type and scene are selected. Still, it works by 500 frames without event, then event (normal, sunny, foggy, cloudy and windy) for about 500 frames, and finally 500 frames without event again. For evaluation for each algorithm with the dataset, each algorithm’s threshold, $\tau$ is set as shown in Table 1.

Moreover, in this subsection, the hardware evaluation, such as CPU and execution time, will also be discussed. A built-in software in GNU/Linux, ‘top’ command was used to evaluate the CPU occupancy and the execution time. It can sort the task by CPU usage and runtime [26].

4. Result and Discussion

The BMC dataset contained five (5) different frame sequences in which every type of frame comes with different events of street scenes. Every frame sequence were in RGB format and contained 1500 frames with a resolution of 640×480. For segmentation quality purpose, private ground truth was also included together and only had 100 selected frames from an overall 1500 frames. The evaluation of BGS algorithm performance started by using dataset 112, and five (5) algorithms are involved. The event in dataset 112 was a street in cloudy conditions without acquisition noise as the normal mode.

Table 2. Segmentation result of dataset 112

| Algorithm          | Recall  | Precision | F-measure | CPU (%) | Elapsed Time (s) |
|--------------------|---------|-----------|-----------|---------|-----------------|
| Frame Difference   | 0.532   | 0.030     | 0.057     | 13.20   | 15.21           |
| ABGL               | 0.701   | 0.035     | 0.067     | 12.11   | 25.37           |
| MOG2               | 0.914   | 0.035     | 0.068     | 19.39   | 29.46           |
| PBAS               | 0.961   | 0.044     | 0.084     | 12.31   | 87.96           |
| Codebook           | 0.911   | 0.040     | 0.077     | 12.31   | 25.51           |

Figure 1. Snapshot of dataset 112 at 195th frame: (a) Input image; (b) Ground truth; (c) Output by Frame Difference; (d) Output by ABGL; (e) Output by MOG2; (f) Output by PBAS; (g) Output by Codebook.

The result in Table 2 and Figure 1 show that PBAS scored the highest score in F-measure compared to other algorithms. Frame difference algorithm scored the lowest result in F-measure compared to others. Still, this algorithm’s processing time is faster than the others algorithm. It also had an almost similar CPU usage with ABGL, PBAS, and Codebook.
The next evaluation of the background subtraction algorithm analysis is based on dataset 212, which consists of a street scene with cloudy conditions. The event contained salt and pepper noise during the whole sequences. This frame sequence also had 1500 input colored frames with 100 private ground truths. All five (5) algorithms were evaluated based on 212 datasets. The result presented in Table 3 and Figure 2 illustrated the segmented output by five (5) algorithms. From the evaluation result, the PBAS algorithm scored highest in all evaluation except it used a lot of processing time. F-measure score dropped for Codebook because it was sensitive to the color-changing created by the salt and pepper noise.

|                | Recall | Precision | F-measure | CPU (%) | Elapsed Time (s) |
|----------------|--------|-----------|-----------|---------|------------------|
| Frame Difference | 0.438  | 0.020     | 0.039     | 11.48   | 18.23            |
| ABGL           | 0.640  | 0.031     | 0.059     | 12.85   | 25.42            |
| MOG2           | 0.752  | 0.030     | 0.057     | 21.39   | 29.05            |
| PBAS           | 0.961  | 0.044     | 0.084     | 12.47   | 92.01            |
| Codebook       | 0.799  | 0.014     | 0.027     | 13.08   | 24.40            |

Figure 2. Snapshot of dataset 212 at 300th frame: (a) Input image; (b) Ground truth; (c) Output by Frame Difference; (d) Output by ABGL; (e) Output by MOG2; (f) Output by PBAS; (g) Output by Codebook.

The next dataset that been used for the algorithm’s performance evaluation was dataset 312. Dataset 312 also had 1500 colored frames with 100 private ground truths. This dataset consisted of a street scene with sunny weather, which generates moving cast shadows. Table 4 explained the details of the performance of five (5) algorithms based on dataset 312 and Figure 3 depicted the snapshot of segmented output for every algorithm.

As the moving cast shadow exists in dataset 312, the result of a certain algorithm showed some drop, especially in F-measure such as Codebook. The performance of the F-measure score for Codebook was poor because the pixel intensity was different every time the shadow moved. The difference in comparison of the pixel intensity from codeword produced low F-measure scored. For the other background subtraction like frame difference is not affected its score because the changing pixel intensity still within its threshold. Among the other algorithm, MOG2 shows the highest usage of CPU, and PBAS consumed the highest processing time.
In dataset 312, the street scene is used, but the foggy weather created noise and made both background and foreground challenging to analyze. This dataset contained frames with resolution 640×480 and 100 frames of ground truth. The evaluation of all five (5) algorithms are continued by using dataset 312, and the recall, precision F-measure, CPU usage and elapsed time are compared. The results are recorded in Table 4, and the segmented output from every algorithm is depicted in Figure 3.

The poor performance of Codebook continues in dataset 312 evaluation was due to the high sensitivity of pixel intensity changing. Other algorithms performances such as ABGL and PBAS gave better results as their algorithms were able to adapt (adaptive threshold) to the changing pixel intensity.

In dataset 412, the street scene is used, but the foggy weather created noise and made both background and foreground challenging to analyze. This dataset contained frames with resolution 640×480 and 100 frames of ground truth. The evaluation of all five (5) algorithms are continued by using dataset 412, and the recall, precision F-measure, CPU usage and elapsed time are compared. The results are recorded in Table 5, and the segmented output from every algorithm is depicted in Figure 4.

The poor performance of Codebook continues in dataset 412 evaluation was due to the high sensitivity of pixel intensity changing. Other algorithms performances such as ABGL and PBAS gave better results as their algorithms were able to adapt (adaptive threshold) to the changing pixel intensity.

### Table 4. Segmentation result of dataset 312

|                   | Recall | Precision | F-measure | CPU (%) | Elapsed Time(s) |
|-------------------|--------|-----------|-----------|---------|-----------------|
| Frame Difference  | 0.423  | 0.020     | 0.038     | 10.30   | 20.28           |
| ABGL              | 0.562  | 0.027     | 0.051     | 10.98   | 28.30           |
| MOG2              | 0.763  | 0.018     | 0.035     | 21.18   | 29.26           |
| PBAS              | 0.860  | 0.044     | 0.083     | 12.40   | 93.93           |
| Codebook          | 0.822  | 0.002     | 0.004     | 12.68   | 33.73           |

**Figure 3.** Snapshot of dataset 312 at 555th frame: (a) Input image; (b) Ground truth; (c) Output by Frame Difference; (d) Output by ABGL; (e) Output by MOG2; (f) Output by PBAS; (g) Output by Codebook.

### Table 5. Segmentation result of dataset 412

|                   | Recall | Precision | F-measure | CPU (%) | Elapsed Time(s) |
|-------------------|--------|-----------|-----------|---------|-----------------|
| Frame Difference  | 0.456  | 0.019     | 0.036     | 10.54   | 19.48           |
| ABGL              | 0.590  | 0.029     | 0.055     | 11.29   | 26.09           |
| MOG2              | 0.656  | 0.011     | 0.023     | 20.66   | 28.28           |
| PBAS              | 0.680  | 0.036     | 0.068     | 12.58   | 42.86           |
| Codebook          | 0.786  | 0.001     | 0.001     | 12.40   | 94.40           |
Figure 4. Snapshot of dataset 412 at 1170th frame: (a) Input image; (b) Ground truth; (c) Output by Frame Difference; (d) Output by ABGL; (e) Output by MOG2; (f) Output by PBAS; (g) Output by Codebook.

Lastly, the evaluation of dataset 512, which had 1500 coloured frames, 100 frames of ground truth, and its resolution was 640×480 with a street scene of windy weather, is investigated. The windy weather with the noise produced a moving background, so it was difficult for the algorithm to conduct background modelling. Table 6 listed the score of five evaluation methods based on dataset 512. From the Table, PBAS obtained the highest score for the F-measure score and followed by MOG2. Figure 5 shows the segmented output from every algorithm.

Table- I:

| Method             | Recall | Precision | F-measure | CPU (%) | Elapsed Time(s) |
|--------------------|--------|-----------|-----------|---------|-----------------|
| Frame Difference   | 0.510  | 0.013     | 0.026     | 10.71   | 19.88           |
| ABGL               | 0.683  | 0.014     | 0.027     | 11.14   | 29.83           |
| MOG2               | 0.842  | 0.020     | 0.040     | 20.51   | 29.27           |
| PBAS               | 0.849  | 0.033     | 0.063     | 12.40   | 93.57           |
| Codebook           | 0.886  | 0.003     | 0.007     | 12.62   | 30.90           |

Figure 5. Snapshot of dataset 512 at 1185th frame: (a) Input image; (b) Ground truth; (c) Output by Frame Difference; (d) Output by ABGL; (e) Output by MOG2; (f) Output by PBAS; (g) Output by Codebook.
5. Conclusion
According to the results for all datasets, it can be concluded that PBAS was a very stable algorithm and can adapt to any event without dropping its segmentation performance. On the contrary, Codebook globally underperformed due to its sensitiveness. In terms of processing time, PBAS consumed a lot of time for processing every frame while the frame difference performed the fastest. MOG2 consumed higher CPU usage compared to the other algorithm. The result obtained based on the synthetic dataset (112, 212, 312, 412, and 512) was not accurate as the evaluation process of synthetic dataset only compares 100 frames of ground truth out of 1500 frames in the whole video. The private ground truth was exclusively focused on a certain frame without considering the other’s frame.

6. References
[1] A. Sofwan et al., “Implementation of Vehicle Traffic Analysis Using Background Subtraction in The Internet of Things (IoT) Architecture,” 2018 6th Int. Conf. Inf. Commun. Technol., vol. 0, no. c, pp. 24–27, 2018.
[2] R. Kalsotra and S. Arora, “Morphological based Moving Object Detection with Background Subtraction Method,” in 4th IEEE International Conference on Signal Processing, Computing and Control (ISPCC 2k17), 2017.
[3] L. Guo, D. Zhou, J. Zhou, S. Kimura, and S. Goto, “Lossy Compression for Embedded Computer Vision Systems,” IEEE Access, vol. 6, pp. 39385–39397, 2018.
[4] A. Mhalla, T. Chateau, S. Gazzah, N. Essoukri, and B. Amara, “An Embedded Computer-Vision System for Multi-Object Detection in Traffic Surveillance,” IEEE Trans. Intell. Transp. Syst., pp. 1–13, 2018.
[5] S. Perri, F. Frustaci, F. Spagnolo, and P. Corsonello, “Design of Real-Time FPGA-based Embedded System for Stereo Vision,” 2018 IEEE Int. Symp. Circuits Syst., 2018.
[6] S.-C. Cheung and C. Kamath, “Robust techniques for background subtraction in urban traffic video,” in Proceedings of Video Communications and Image Processing, SPIE Electronic Imaging, 2004, pp. 881–892.
[7] Y. Shen et al., “Real-time and robust compressive background subtraction for embedded camera networks,” IEEE Trans. Mob. Comput., vol. 15, no. 2, pp. 406–418, 2016.
[8] P. Ramya and R. Rajeswari, “A Modified Frame Difference Method Using Correlation Coefficient for Background Subtraction,” Procedia Comput. Sci., vol. 93, no. September, pp. 478–485, 2016.
[9] B. Wang and P. Dudek, “A Fast Self - tuning Background Subtraction Algorithm,” IEEE Conf. Comput. Vis. Pattern Recognit., pp. 4321–4324, 2014.
[10] O. Domínguez et al., “Fast Background Subtraction For Real-Time Motion Detection,” in Congreso Internacional De Ingeniería Electrónica Memoria Electro, 2011, vol. 33, pp. 212–215.
[11] A. Sobral and A. Vacavant, “A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos,” Comput. Vis. Image Underst., vol. 122, no. May, pp. 4–21, 2014.
[12] N. a. Mandellos, I. Keramitsoglou, and C. T. Kiranoudis, “A background subtraction algorithm for detecting and tracking vehicles,” Expert Syst. Appl., vol. 38, no. 3, pp. 1619–1631, 2011.
[13] C. Stauffer and W. E. L. Grimson, “Adaptive background mixture models for real-time tracking,” Proc. 1999 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Cat No PR00149, vol. 2, no. c, pp. 246–252, 1999.
[14] Z. Zivkovic, “Improved adaptive Gaussian mixture model for background subtraction,” Proc. 17th Int. Conf. Pattern Recognition, 2004. ICPR 2004., vol. 2, no. 2, pp. 28–31, 2004.
[15] C. H. Manaswini, “Efficient Vehicle Counting and Classification using Robust Multi-Cue Consecutive Frame Subtraction,” Glob. J. Comput. Sci. Technol. Graph. Vis., vol. 13, no. 8,
[16] N. Shah, A. Pingale, V. Patel, and N. V George, “An Adaptive Background Subtraction Scheme for Video Surveillance Systems,” *2017 IEEE Int. Symp. Signal Process. Inf. Technol.*, pp. 13–17, 2017.

[17] M. Hofmann, P. Tiefenbacher, and G. Rigoll, “Background Segmentation with Feedback : The Pixel-Based Adaptive Segmenter,” in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2012.

[18] Y. Xu, J. Dong, B. Zhang, and D. Xu, “Background modeling methods in video analysis: A review and comparative evaluation,” *CAAI Trans. Intell. Technol.*, vol. 1, no. 1, pp. 43–60, 2016.

[19] A. Nurhadiyatna, B. Hardjono, and A. Wibisono, “Improved Vehicle Speed Estimation Using Gaussian Mixture Model and Hole Filling Algorithm,” in *International Conference on Advanced Computer Science and Information System (ICACSIS)*, 2013, pp. 451–456.

[20] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis, “Real-time foreground-background segmentation using codebook model,” *Real-Time Imaging*, vol. 11, no. 3, pp. 172–185, 2005.

[21] H. M. Desai and V. Gandhi, “A Survey: Background Subtraction Techniques,” *Int. J. Sci. Eng. Res.*, vol. 5, no. 12, pp. 1365–1367, 2014.

[22] O. Barnich and M. Van Droogenbroeck, “ViBe: A universal background subtraction algorithm for video sequences,” *IEEE Trans. Image Process.*, vol. 20, no. 6, pp. 1709–1724, 2011.

[23] L. Kodwani, “Automatic Vehicle Detection, Tracking and Recognition of License Plate in Real Time Videos,” National Institute of Technology Rourkela, 2013.

[24] L. Lacassagne and A. Manzanera, “Motion Detection:Fast And Robust Algorithms For Embedded System,” *Int. Conf. Image Process.*, 2009.

[25] N. Goyette, P.-M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar, “Changedetection.net: A new change detection benchmark dataset,” in *Proc, IEEE Workshop on Change Detection (CDW-2012)*, 2012.

[26] N. F. Kahar, R. B. Ahmad, Z. Hussin, and A. N. C. Rosli, “Embedded Smart Camera Performance Analysis,” in *2009 International Conference on Computer Engineering and Technology*, 2009, pp. 79–83.