

# Evaluation of Background Subtraction Algorithms with Post-processing

Donovan H. Parks and Sidney S. Fels

University of British Columbia

Electrical and Computer Engineering, UBC, Vancouver, Canada

donovanp@ece.ubc.ca

## Abstract

*Processing a video stream to segment foreground objects from the background is a critical first step in many computer vision applications. Background subtraction (BGS) is a commonly used technique for achieving this segmentation. The popularity of BGS largely comes from its computational efficiency, which allows applications such as human-computer interaction, video surveillance, and traffic monitoring to meet their real-time goals.*

*Numerous BGS algorithms and a number of post-processing techniques that aim to improve the results of these algorithms have been proposed. In this paper, we evaluate several popular, state-of-the-art BGS algorithms and examine how post-processing techniques affect their performance. Our experimental results demonstrate that post-processing techniques can significantly improve the foreground segmentation masks produced by a BGS algorithm. We provide recommendations for achieving robust foreground segmentation based on the lessons learned performing this comparative study.*

## 1. Introduction

Background subtraction (BGS) is a widely used real-time method for identifying foreground objects in a video stream. It is the first significant step in many computer vision applications, including human-computer interaction, traffic monitoring, and video surveillance. This has prompted the development of a wide range of different BGS algorithms, along with a number of post-processing techniques that aim to improve their performance.

The most common paradigm for performing BGS is to build an explicit model of the background. Foreground objects are then detected by calculating the difference between the current frame and this background model. A binary foreground mask can be constructed by classifying a pixel with an absolute difference above a threshold as being from a foreground object.

Unfortunately, there are a number of factors which make

obtaining high accuracy foreground masks difficult. For example, a BGS algorithm should be robust to changing illumination conditions, able to ignore the movement of small background elements, and capable of incorporating new objects into the background model. All this must be achieved in a computationally efficient manner that allows real-time requirements to be met.

Post-processing of the foreground mask produced by a BGS algorithm is required to gain robustness to many of these factors. This post-processing can range from explicit noise removal operating at the pixel-level to connected component labeling which identifies object-level elements. These post-processing techniques can significantly improve the quality of a foreground mask.

The contributions of this paper are three-fold. First, a comparative study of seven popular, state-of-the-art BGS algorithms is performed. We believe this is the most extensive comparison of BGS algorithms that has been conducted to date in terms of the number of test sequences considered. Secondly, we examine the affect of different post-processing techniques on a select subset of these BGS algorithms. We do not know of any other studies that compare the performance of BGS algorithms after post-processing has been performed. Finally, we compile a list of recommendations based on the lessons we have learned performing this comparative study.

## 2. Related Work

We distinguish between surveys which compare the theoretical and qualitative aspects of BGS algorithms and comparative studies that focus primarily on quantitative experimental results.

### 2.1. Surveys

Recently, Radke *et al.* [15] performed a comprehensive survey of *image change detection* algorithms. This survey considers a wide-range of methods for detecting changes in images and discusses issues related to pre-processing of input data, post-processing of foreground masks, and methodologies for evaluating performance. Much of this discussion

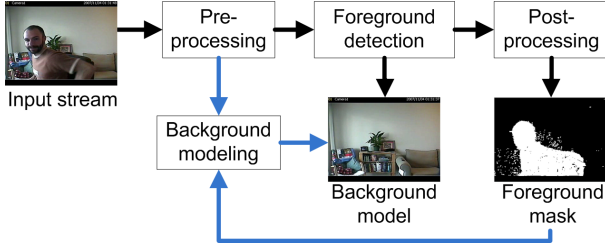


Figure 1. Data flow diagram of a typical BGS algorithms.

is directly related to BGS algorithms.

Piccardi [12] investigated the computational complexity, memory requirements, and theoretical accuracy of seven BGS algorithms. Although Piccardi argues effectively for why some algorithms *should* be more accurate than others, he does not support his arguments with experimental results. The comparative study conducted in this paper directly addresses this issue.

## 2.2. Comparative studies

A comprehensive comparison of BGS algorithms was performed at VSSN06 [1]. The major strength of the VSSN06 comparison is that all implementations were provided by the algorithms' authors. Unfortunately, many popular, state-of-the-art algorithms were absent from this comparison. Our work considers state-of-the-art algorithms that are representative of the field.

Cheung *et al.* [4, 5] provide a comparison of six BGS algorithms in the context of traffic monitoring. This comparison is performed on four grayscale test sequences of road scenes with different environmental conditions. Our comparison considers all the significant algorithms evaluated by Cheung *et al.* on an extensive test set of thirteen colour video sequences.

A comparison of ten different BGS algorithms was conducted by Toyama *et al.* [18]. This comparison was performed on seven test sequences that each consider a specific factor such as sudden illumination changes. They demonstrate that the design of their Wallflower algorithm allows superior foreground masks to be obtained under these different factors. For example, to deal with sudden illumination changes the Wallflower algorithm keeps several background models and switches between these when a sudden illumination change occurs. We consider techniques such as this to be a post-processing mechanism since they are independent of the background model. In our comparison, we evaluate the effectiveness of different post-processing techniques under a range of background models.

## 3. BGS Algorithms

BGS algorithms typically follow the data flow diagram illustrated in Figure 1 which consists of four major processing blocks: pre-processing, background modeling, fore-

ground detection, and post-processing. The survey by Radke *et al.* [15] provides a useful summary of pre-processing techniques. Background modeling, foreground detection, and post-processing are discussed in this section.

### 3.1. Background modeling

The defining characteristic of a BGS algorithm is how it defines and updates its background model. In this section, we describe the different background modeling techniques considered in our comparative study.

Throughout the remainder of this paper  $I_t^c(x,y)$  and  $B_t^c(x,y)$  are used to denote the value of channel  $c$  of the pixel at location  $(x,y)$  at time  $t$  for the incoming video stream and the background model, respectively. The pixel location  $(x,y)$  is dropped when the spatial location is irrelevant.

#### 3.1.1 Recursive techniques

Recursive techniques maintain a single background model that is updated with each new video frame. These techniques are generally computationally efficient and have minimal memory requirements.

**Running Gaussian average (RGA):** If we consider the background to be nearly static, then the main source of variation in a pixel's colour will be due to camera noise. Since it is common to model camera noise as being Gaussian, it is natural to model each pixel in the background model as a Gaussian distribution [19].

**Gaussian mixture model (GMM):** In order to model multi-modal backgrounds, Stauffer and Grimson [16] proposed modeling each channel of a pixel as a mixture of  $K$  Gaussians. We refrain from giving a formal treatment of GMMs due to space constraints and because GMMs have been well explored in the literature [16, 13, 4, 20]. Our implementation uses the update equations suggested by Powers and Schoonees [13].

**GMM with adaptive number of Gaussians (AGMM):** The Stauffer-Grimson algorithm uses a fixed number of Gaussians to model each pixel. An interesting extension has recently been proposed by Zivkovic [20], which shows how to automatically adapt the number of Gaussians being used to model a given pixel. This extension reduces the algorithm's memory requirements, increases its computational efficiency, and can improve performance when the background is highly multi-modal [20].

**Approximated median filtering (AMF):** All of the above methods model pixels using Gaussian distributions. An alternative proposed by McFarlane and Schofield [10] uses a recursive filter to estimate the median using the following update equation:

$$B_{t+1}^c = \begin{cases} B_t^c + 1 & \text{if } I_t^c > B_t^c \\ B_t^c - 1 & \text{if } I_t^c < B_t^c \\ B_t^c & \text{if } I_t^c = B_t^c \end{cases} \quad (1)$$

The major strengths of this approach are its computational efficiency, robustness to noise, and simplicity. A notable limitation is that it does not model the variance of a pixel.

### 3.1.2 Non-recursive techniques

Non-recursive techniques maintain a buffer  $L$  of  $n$  previous video frames and estimate a background model based solely on the statistical properties of these frames. This causes non-recursive techniques to have higher memory requirements than recursive techniques. However, since they have explicit access to the most recent  $n$  video frames they can model aspects of the data not possible with recursive techniques.

**Median filtering:** Median filtering sets each channel of a pixel in the background model to be the median value as determined from the buffer of video frames. We consider a slight modification of the technique proposed by Calderara *et al.* [3], which extends its use to colour images. Our algorithm works as follows:

1. Sort an *extended* buffer  $E^c(x,y) = (L_1^c(x,y), L_2^c(x,y), \dots, L_n^c(x,y), B_{t-1}^c(x,y))$  so values are in ascending order.
2. Set the background model to the median of this extended buffer:  $B_t^c(x,y) = E_{\frac{n+1}{2}}^c(x,y)$
3. Estimate a measure of variance:  $\Psi^c(x,y) = \lambda \left( E_{\frac{n+1}{2}+k}^c(x,y) - E_{\frac{n+1}{2}-k}^c(x,y) \right)$ , where  $k$  is a user specified parameter

Extending the buffer to include the last background model value makes the algorithm more robust to noise when small buffer sizes are used [3].

**Mediod filtering:** Instead of independently finding the median of each channel, the mediod of a pixel can be estimated from the buffer of video frames as proposed by Cucchiara *et al.* [6]. This has the advantage of capturing the statistical dependencies between colour channels. Mediod filtering is the only background modeling technique we consider that does not treat each colour channel independently. A notable short-coming of this approach is that it does not produce a measure of variance.

**Eigenbackgrounds (EigBG):** All the other approaches we have considered model each pixel in the background model independently. The approach proposed by Oliver *et al.* [11] captures spatial correlations by applying principal component analysis to a set of  $N_L$  video frames that do not contain any foreground objects. This results in a set of basis functions of which only the first  $d$  are required to capture the primary appearance characteristics of these frames. A new frame can then be projected into the eigenspace defined by these  $d$  basis functions and then back projected into the original image space. Since the basis functions only model

the static part of the scene when no foreground objects are present, the back projected image will not contain any foreground objects. As such, it can be used as a background model.

The major limitation of this approach is that computing the basis functions requires a set of video frames without foreground objects. As such, it is not clear how the basis functions can be updated over time if foreground objects are continually present in the scene.

### 3.2. Foreground detection

Pixels in a new video frame that cannot be adequately explained by the background model are considered to be from a foreground object. The major distinction that defines how this comparison is performed is whether or not the background model is statistical in nature. Algorithms that lack a statistical framework (*e.g.*, AMF, Mediod, EigBG) classify a new pixel as being from the foreground whenever  $|I_t(x,y) - B_t(x,y)| > T$ , where  $T$  is a user defined threshold. The major limitation of this approach is that it uses a single threshold for all pixel models even though some pixels may exhibit more variation than others.

As such, methods that provide a measure of variance for each pixel are preferable. Algorithms which model pixels as a probability density function (*e.g.*, RGA, GMM, AGMM, Median) classify a new pixel as being from the foreground whenever  $p(I_t^c | B_t^c) < T^c = \eta \Psi^c$  for any channel  $c$ . The threshold  $T^c$  is set proportional to the estimated variation,  $\Psi^c$ , in order to ensure a pixel is classified as being from the foreground only when it is outside the normally observed level of variance.

### 3.3. Post-processing Techniques

We consider a number of post-processing techniques that can be used to improve upon the foreground masks that result from foreground detection. Figure 2 gives a data flow diagram that indicates the order in which post-processing techniques are performed in our comparative study.

**Noise removal:** Due to camera noise and limitations of the background model, the foreground mask typically contains numerous small “noise” blobs. These erroneous blobs can be removed by applying a noise filtering algorithm to the foreground mask. Removing these erroneous blobs early is desirable since they can interfere with later post-processing stages.

**Blob processing:** Most applications using BGS are interested in identifying foreground objects. As such, connected-component labeling is almost always performed in order to identify object-level blobs. In our comparative study, we examine how morphological closing and area thresholding can improve the blobs identified in a foreground mask. Morphological closing is used to fill internal holes and small gaps whereas area thresholding is used to remove blobs that are insufficiently large to be of interest.

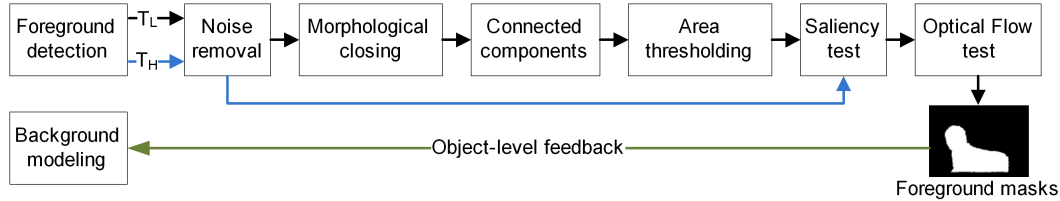


Figure 2. Data flow diagram of the post-processing techniques considered in our comparative study.

**Saliency test:** Saliency testing works on the premise that at least some portion of a foreground object should be poorly explained by the background model [6]. If we believe this premise, then we can verify that a blob represents a valid foreground object by checking that it contains a certain percentage of pixels that are highly salient compared to the background.

**Optical flow test:** When a foreground object stops moving within a scene it will eventually be incorporated into the background model. If the object now begins to move, the area it previously occupied will be incorrectly detected as a foreground blob commonly referred to as a *ghost*. This ghost will remain until the background model adapts to the newly exposed background. This problem can be alleviated by removing any blobs that have an average optical flow near zero since ghost blobs have no motion [6, 4]. The limitation of this method is that a valid foreground object that comes to rest for any period of time will be incorrectly classified as a ghost.

**Object-level feedback:** Elgammal [7] noted that there are two contrasting approaches for updating a background model. In *unconditional updating* every pixel in the background model is updated, whereas in *conditional updating* only pixels that have been identified as being from the background (as indicated by the most recent foreground mask) are updated.

Conditional updating enhances the detection of foreground objects since the background model will not become polluted with foreground pixel information. Furthermore, it removes the problem of ghosts since valid foreground objects will not be incorporated into the background model. The danger of conditional updating is that an incorrectly identified foreground pixel will continually be misclassified since the background model will never adapt to it. When using conditional updating, application specific testing may be required to ensure foreground objects that remain in the scene for an extended period of time are eventually incorporated into the background model.

In our comparative study, we examine the use of object-level conditional updating. Specifically, we performed conditional updating using the foreground mask produced at the end of our post-processing chain where all remaining blobs are highly likely to be from actual foreground objects.

## 4. Experimental Results

In this section, a quantitative comparison of the BGS algorithms discussed in Section 3.1 is performed. We then explore how the post-processing techniques in Section 3.3 affect the performance of a select subset of these BGS algorithms.

Our comparison of BGS algorithms is performed using a diverse set of 7 outdoor and 6 indoor video sequences. The outdoor sequences present a significant challenge as they contain moving background elements (*i.e.*, video 4, video 7 [1]; fountain, water surface [9]; waving trees [18]), objects moving at varying speeds (*i.e.*, busy road [9]), and objects of varying sizes (*i.e.*, video 7 [1], campus [14]). The indoor sequences are less challenging (*i.e.*, video 8 [1], intelligent room [14], camouflage [18]), but do exhibit examples of shadows (*i.e.*, laboratory [14]), slowly varying lighting conditions (*i.e.*, time of day [18]), and large poorly textured objects that can give rise to the foreground aperture problem (*i.e.*, foreground aperture [18]). For evaluating the post-processing techniques we make use of the aforementioned videos from [14] and [9] since these cover a range of interesting scenarios and are all real footage.

Experimental results are reported using precision-recall plots in order to allow our results to be compared with [4, 5] and [1]. To determine which parameter values produce the best results, we calculate the average F-measure [2] over recall = {0.7, 0.95}.

### 4.1. Evaluation of BGS algorithms

The results of our comparative study are given in Figure 3. We generated these precision-recall curves by systematically changing the threshold parameter,  $T$ . Table 1 lists the range of parameter values considered. For the GMM-based algorithms, we considered parameter values based on the recommendations in [13] and [4]. For the non-recursive techniques, we determined the range of parameter values to test based on informal experiments and our experience with these algorithms.

Due to space constraints we are unable to provide precision-recall curves for each test sequence and instead provide averaged results in Figure 3. As expected, the performance varies considerably between test sequences, but as suggested by Figure 3 the difference between the performance of these algorithms is typically less than 10%. The

Algorithm	Fixed parameters	Test parameters
RGA	None	Learning rate, $\alpha = \{5e^{-4}, 0.001, 0.005, 0.01, 0.02, 0.1\}$
GMM, AGMM	Initial weight, $w_o = \alpha$	Learning rate, $\alpha = \{5e^{-4}, 0.001, 0.005, 0.01, 0.02, 0.1\}$ Gaussians, $K = \{\mathbf{3}, 4, 5\}$ Initial variance, $\sigma_o^2 = \{11, \mathbf{36}\}$ Weight threshold, $\Gamma = \{\mathbf{0.75}, 0.9\}$
AMF	None	Sampling rate, $S_r = \{1, 3, 5, 9, 13\}$
Median	Variance parameter, $k = 4$	Sampling rate, $S_r = \{1, 3, 5, 9, 13\}$ Buffer size, $n = \{7, \mathbf{21}, 51\}$
Mediod	None	Same as Median
EigBG	Learning frames, $N_L = 100$	Eigenspace dimensionality, $d = \{10, 20, 30\}$

Table 1. Parameter settings tested during evaluation of BGS algorithms. All plots in this paper use the parameter values shown in bold. The bold values give the best or nearly the best performance while not unnecessarily increasing the computational requirements of the algorithm. The most critical parameter of a BGS algorithm (apart from the threshold) is the learning rate for recursive techniques and the sampling rate for non-recursive rates. For these parameters and the eigenspace dimensionality, we find the value resulting in the best performance independently for each test sequence.

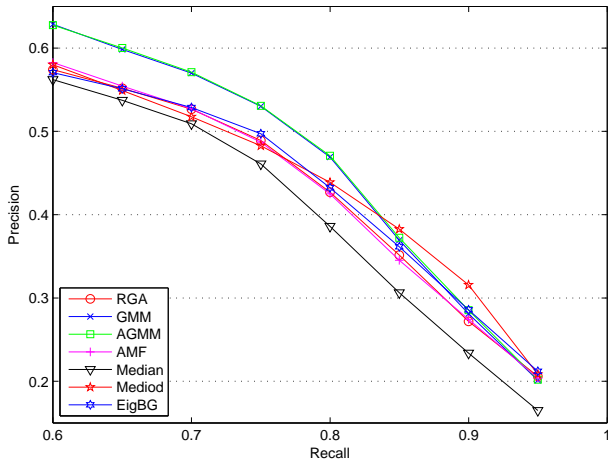


Figure 3. Precision-recall plots for 7 popular, state-of-the-art BGS algorithms. Results are the average over the 13 test sequences considered.

fountain, busy road, video 4, foreground aperture, and time of day sequences were challenging for all the BGS algorithms. The median filtering algorithm performed considerable worse on the camouflage and waving trees sequences compared to the other algorithms which is why it has the worst average performance. As reflected in Figure 3, the mediod algorithm generally outperformed the other algorithms when the *recall* > 0.85. On video 7, the GMM and AGMM algorithms performed extremely well compared to the other algorithms when the *recall* < 0.85. This largely accounts for the relatively high average performance of these algorithms in Figure 3 when the *recall* < 0.85.

There are a number of useful results that can be obtained from Figure 3:

- None of the algorithms consistently produce high quality foreground masks. This suggests that there is a significant opportunity to improve foreground masks using post-processing techniques.
- Modeling a pixel as a mixture of Gaussians improves

performance compared to using a single Gaussian.

- The AGMM extension has little impact on performance. However, we did observe that this extension makes the algorithm more computationally and memory efficient.
- Mediod filtering clearly outperforms median filtering indicating it is advantageous to model the dependencies between colour channels.
- AMF performs relatively well given its low computational and memory requirements.
- The relatively strong results obtained by the EigBG algorithm are notable considering that it does not update the background model.

#### 4.2. Evaluation of post-processing techniques

The effect of different post-processing techniques on the performance of a select subset of BGS algorithms is examined in this section. Based on the results in Section 4.1 and in order to cover a range of background modeling techniques we have chosen to evaluate the AGMM, AMF, Mediod, and EigBG algorithms. We set the parameter values of these algorithms to those specified in Table 1.

We quantitatively evaluated these post-processing techniques in two different manners. In Figure 4 we examine the effect different parameter values have on the post-processing techniques being considered. The performance of each BGS algorithm after each stage of the post-processing pipeline is given in Figure 5.

**Noise removal:** We applied a density-based noise removal method to the foreground masks which discards a foreground pixel if it has less than  $\rho$  foreground pixels amongst its 8-connected neighbours. Figure 4 shows that if the noise filtering is too aggressive performance will decrease. When set appropriately, noise removal can moderately increase the performance of a BGS algorithm as illustrated by the results in Figure 5.

**Morphological closing:** We performed morphological closing using a square kernel with varying widths,  $w$ , as

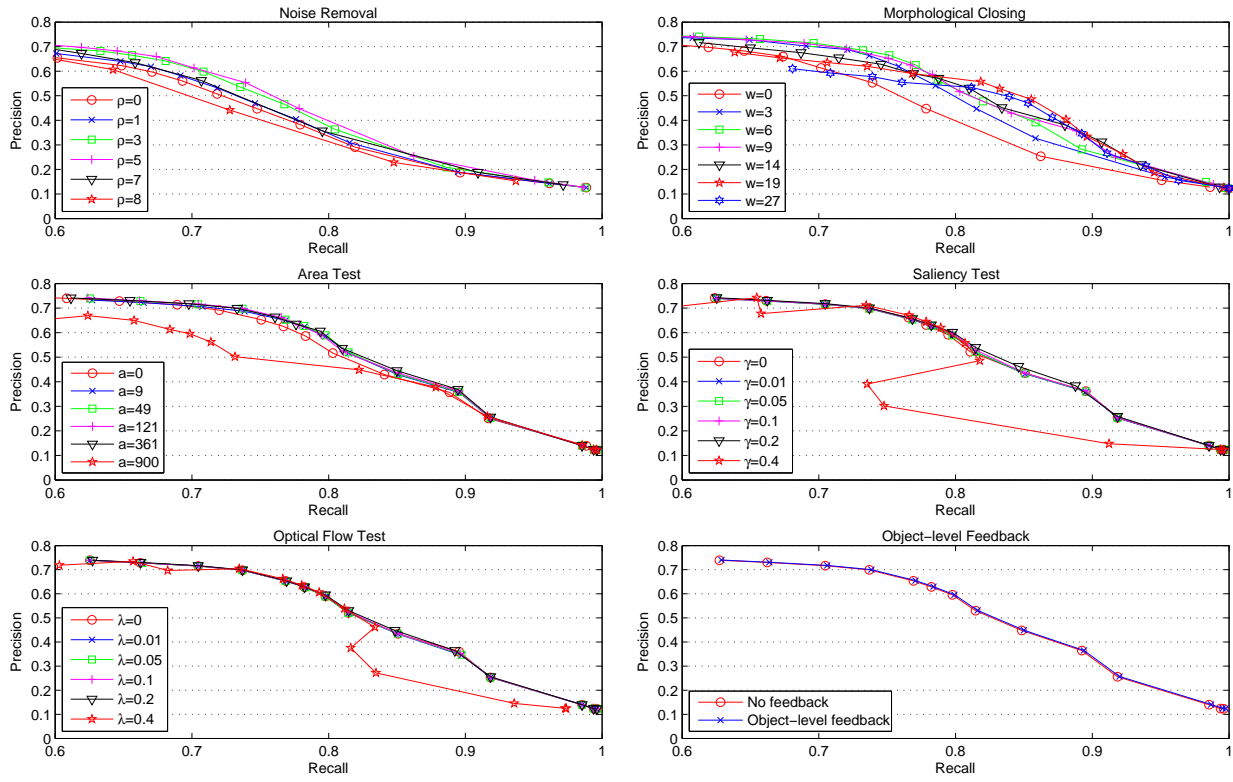


Figure 4. The affect different parameter values have on the post-processing techniques. These plots are the average over 13 test sequences. Results are only shown for the AGMM algorithm as the trends are similar for all the BGS algorithms. We use the best parameter values found for all previous post-processing stages when evaluating the current stage.

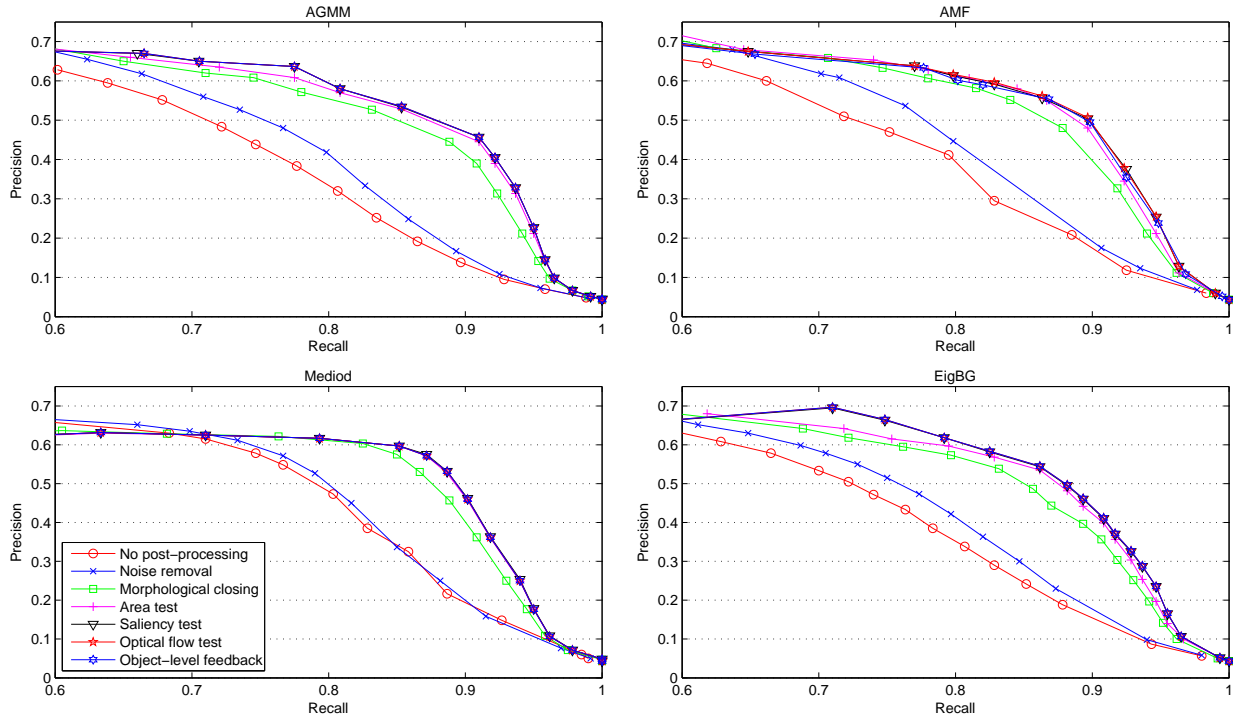


Figure 5. Precision-recall plots for each BGS algorithm after each stage of the post-processing pipeline. These plots are the average over the 6 test sequences from [14] and [9].

shown in Figure 4. These results indicate that performance is improved for a range of values, but that when  $w$  is set too large performance can decrease. Figure 5 indicates that the performance of all algorithms can be significantly increased by applying morphological closing.

**Area thresholding:** Figure 4 shows the results when all blobs consisting of less than  $a$  pixels are discarded. Setting  $a$  too large results in a significant decrease in performance. Even when  $a$  is less than the minimum size of all objects of interest, poor results will typically be observed since morphological closing sometimes fails to close all gaps between blobs belonging to the same object. For this reason, we recommended setting  $a$  to a value well below the size of the smallest object of interest (*e.g.*, 25%).

**Saliency test:** Saliency testing is used to remove any blobs where at least  $\gamma$  percent of its pixels are not highly salient. We define a highly salient pixel to be one that is identified as a foreground pixel when  $T_H$  is set to twice the value of  $T_L$  (Figure 2). Figure 4 clearly demonstrates that if  $\gamma$  is set too large, performance can decrease significantly. Even when set appropriately, saliency testing has little discernible affect on performance (Figure 5) since at this stage in the post-processing pipeline most frames contain only highly salient blobs. Nevertheless, we recommend its use since it is computationally inexpensive and will occasionally remove low saliency, false positive blobs.

**Optical flow test:** Figure 4 shows the results when all blobs with an optical flow less than  $\lambda$  are removed from the foreground mask. These results demonstrate that the performance decreases rapidly if  $\lambda$  is set too high. Even with a suitable  $\lambda$  value, optical flow testing results in a minimal increase in performance (Figure 5). However, it can be a useful technique if all objects moving below a certain speed should be ignored.

**Object-level feedback:** Object-level feedback results in a nominal change in performance as indicated in Figures 4 and 5. Nevertheless, we recommend this post-processing technique as a computationally inexpensive method for suppressing the formation of ghosts.

## 5. Discussion

### 5.1. Parameter tuning

To mitigate the time required to tune a system, it is beneficial to have an *a priori* notion of *reasonable* parameter values. The parameter values specified in Tables 1 and 2 can be used as a starting point for tuning any of the considered BGS algorithm or post-processing techniques.

### 5.2. Recommendations

Based on the lessons we learned while conducting this comparative study, we have compiled the following list of recommendations regarding the use of BGS with post-processing:

- Object-level feedback is recommended over optical flow testing for dealing with ghosts. Optical flow testing is computationally expensive [17] and can easily result in slowly moving objects being discarded from the foreground mask. We recommend optical flow testing only when all objects moving below a well-defined speed should be ignored (*e.g.*, a traffic monitoring application where cars moving less than 5 km/h are not of interest).
- We highly recommend the use of all other post-processing techniques considered. Noise removal, morphological closing, and area testing significantly improve performance as illustrated in Figure 5. Although saliency testing and object-level feedback have little discernible affect on performance, they are both computationally inexpensive and do significantly improve the quality of certain frames. In particular, saliency testing will occasionally remove low saliency, false positive blobs and object-level feedback prevents the formation of ghosts.
- Parameter tuning must be performed on both the BGS algorithm and the post-processing techniques in order to obtain satisfactory performance (see Figure 4).
- The performances of the BGS algorithms are similar after post-processing. When the *recall* > 0.75, the precision between the algorithms is always less than 6.5%. None of the algorithms consistently outperforms any other. We recommend using AMF when computational efficiency or minimizing memory requirements is a high priority. When a more theoretically sound BGS algorithm is desired, AGMM or median filtering can be used at the expense of additional computation and memory.

### 5.3. Limitations and Future Work

Numerous BGS algorithms have been proposed. Since it is only practical to evaluate a small subset of these, we have selected state-of-the-art algorithms that cover a range of popular methodologies. We plan to extend our comparative study to include other methodologies in the future (*e.g.*, [7, 8]). A thoughtful reviewer has also indicated that a statistical test, such as McNemar’s test, could be used to determine if the performance between BGS algorithms or post-processing stages is significant.

A diverse set of video sequences for comparing BGS algorithms has not been established. We overcome this by performing our comparative study on a collection of previously used test sets [1, 9, 14, 18]. Nevertheless, the BGS community would benefit from establishing a *standard* test set which demonstrates canonical problems and facilitates easy quantitative comparisons. Such a test set would also allow parameter values to be determined in a more structured manner (*e.g.*, cross-validation).

	Noise removal, $\rho$	Morphological closing, $w$	Area test, $a$	Saliency test, $\gamma$	Optical flow test, $\lambda$
AGMM	$3.0 \pm 0.0$	$7.3 \pm 2.3$	$324 \pm 0.0$	$0.05 \pm 0.0$	$0.2 \pm 0.0$
AMF	$3.3 \pm 0.8$	$9.0 \pm 0.0$	$307 \pm 160$	$0.043 \pm 0.016$	$0.2 \pm 0.0$
Mediod	$2.7 \pm 0.8$	$7.3 \pm 2.3$	$370 \pm 181$	$0.092 \pm 0.021$	$0.2 \pm 0.0$
EigBG	$3.0 \pm 0.0$	$7.3 \pm 1.5$	$286 \pm 91$	$0.043 \pm 0.016$	$0.2 \pm 0.0$

Table 2. Parameter values resulting in the best performance reported as the mean and standard deviation over all the test sequences.

All the considered post-processing techniques have a computational complexity that is linear in the number of *foreground* pixels. This suggests that applying techniques which remove significant numbers of false positive foreground pixels at low computational cost should be applied early in order to reduce the overall computational complexity of the pipeline. The post-processing pipeline we propose reflects this insight and significantly improves the quality of foreground masks. As future work we intend to evaluate the merits of other reasonable pipeline orderings.

## 6. Conclusions

Numerous computer vision applications depend on BGS to identify foreground objects. We have performed a comparative evaluation of seven BGS algorithms. This evaluation indicates that how the background is modeled does influence performance. However, it also reveals that simple modeling techniques (*i.e.*, approximated median filtering) can perform nearly as well as more complex and theoretically sound techniques (*i.e.*, GMM).

We then examined the effect of different post-processing techniques on a subset of these BGS algorithms. Our results indicate that post-processing can significantly improve the performance of all the BGS algorithms considered when parameter values are set appropriately. To facilitate parameter selection, we have indicated initial parameter values that can be used for rapidly tuning all the BGS algorithms and post-processing techniques considered in this paper.

Finally, we have compiled a list of recommendations based on the lessons learned during this research. This list should be a great aid to researchers interested in incorporating BGS with post-processing into their applications.

## Acknowledgments

This work was supported by a grant from the Natural Sciences and Engineering Research Council of Canada.

## References

- [1] mmc36.informatik.uni-augsburg.de/mediawiki/data/6/65/vssn-algo.pdf.
- [2] S. Agarwal and A. Awan. Learning to detect objects in images via a sparse, part-based representation. *IEEE PAMI*, 26(11):1475–1490, 2004. Member-Dan Roth.
- [3] S. Calderara, R. Melli, A. Prati, and R. Cucchiara. Reliable background suppression for complex scenes. In *VSSN06*, pages 211–214, 2006.
- [4] S.-C. S. Cheung and C. Kamath. Robust techniques for background subtraction in urban traffic video. *SPIE04*, 5308:881–892, 2004.
- [5] S.-C. S. Cheung and C. Kamath. Robust background subtraction with foreground validation for urban traffic video. *JASPO5*, 14:2330–2340, 2005.
- [6] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Detecting moving objects, ghosts, and shadows in video streams. *IEEE PAMI*, 25(10):1337–1342, 2003.
- [7] A. Elgammal, D. Hardwood, and L. Davis. Non-parametric model for background subtraction. In *ECCV00*, pages 751–767, 2000.
- [8] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis. Real-time foreground-background segmentation using codebook model. *Real-time Imaging*, 11:167–256, 2005.
- [9] L. Li, W. Huang, I. Y.-H. Gu, and Q. Tian. Statistical modeling of complex backgrounds for foreground object detection. *IEEE Transactions on Image Processing*, 13(11):2004, 1459–1472.
- [10] N. McFarlane and C. Schofield. Segmentation and tracking of piglets in images. *MVA*, 8:187–193, 1995.
- [11] N. Oliver, B. Rosario, and A. Pentland. A bayesian computer vision system for modeling human interactions. *IEEE PAMI*, 22:831–843, 2000.
- [12] M. Piccardi. Background subtraction techniques: a review. In *SMC04*, volume 4, pages 3099–3104, 2004.
- [13] P. W. Power and J. A. Schoonees. Understanding background mixture models for foreground segmentation. In *Proc. of the Image and Vision Computing*, 2002.
- [14] A. Prati, I. Mikic, M. M. Trivedi, and R. Cucchiara. Detecting moving shadows: Algorithms and evaluation. *IEEE PAMI*, 25:918–923, 2003.
- [15] R. Radke, S. Andra, O. Al-Kofahi, and B. Roysam. Image change detection algorithms: a systematic survey. *IEEE Trans. Image Processing*, 14:294–307, 2005.
- [16] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *CVPR99*, volume 2, page 252, 1999.
- [17] C. Stiller and J. Konrad. Estimating motion in image sequences. *IEEE Signal Processing Magazine*, pages 70–91, 1999.
- [18] K. Toyama, J. Krumma, B. Brumitt, and B. Meyers. Wallflower: principles and practice of background maintenance. In *ICCV99*, page 1999, 1999.
- [19] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland. Pfunder: real-time tracking of the human body. *IEEE PAMI*, 19(7):780–785, 1997.
- [20] Z. Zivkovic and F. van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters*, 27(7):773–780, 2006.