

# A Statistical Background Modeling Algorithm for Real-Time Pixel Classification

Ricardo Acevedo-Ávila, Miguel González-Mendoza, Andrés García-García

Tecnológico de Monterrey, Department of Postgraduate Studies,  
Campus Estado de México,  
Mexico

{ricardo.acevedo, mgonza, garcia.andres}@itesm.mx

**Abstract.** This paper introduces a statistical background pixel classifier intended for real-time and low-resource implementation. The algorithm works within a smart video surveillance application aimed to detect unattended objects in images with fixed backgrounds. The algorithm receives an input image and builds an initial background model based on image statistics. Using this information, the algorithm identifies new objects that do not belong to the original image. The algorithm categorizes image pixels in four possible classes: shadows, midtones, highlights and foreground pixels. The classification stage produces a binary mask where only objects of interest are shown. The pixel classifier processes Quarter VGA (320 x 240) gray-scale images at a nominal processing rate of 30 frames per second. Higher resolutions such as VGA (640 x 480) have been also tested. We compare results with traditional statistical background modeling methods. Our experiments demonstrate that our approach achieves successful background segmentation at a minimal resource consumption while maintaining real-time execution.

**Keywords.** Background modeling, embedded computer vision, statistical pixel modeling, image processing, object detection.

## 1 Introduction

One key function to be performed in computer vision applications is background segmentation. Background segmentation is used to build a reference model of a background scene in order to facilitate higher level computer vision tasks like *Binary Large Object* (Blob), detection, pattern and object recognition [1]. It is essential in applications such as video surveillance, automatic event monitoring, and object tracking [2].

The background segmentation task must be able to distinguish foreign objects that do not belong to the original scene. Moreover, the background model must also offer an accurate representation of the scene while being unaffected by environmental noise such as illumination and weather changes [3]. Different techniques have been extensively studied and are widely available through literature [4]. However, algorithm implementation is not only constrained by the application, but also by the underlying hardware driving the processing tasks [5].

Typically, complex general purpose algorithms are computationally heavy and resource demanding, while simpler, lighter algorithms are very case-specific and not reliable. Known background segmentation techniques range from basic image subtraction to elaborated implementations of fuzzy *Mixture of Gaussians*.

The former method involves direct subtraction between the current image and the initial background model [6], while the latter is used to model dynamic composite surfaces with multiple textures [7, 8]. Surveys of different algorithms can be found in the literature [9, 10, 11]. In essence, background modeling can be classified in two major categories: Non-statistical Background Modeling and Statistical Background Modeling [12]. On this work we will focus on statistical background modeling.

Statistical methods involve the classification of a pixel based on historical knowledge built through time. A probability function of the background pixels is first estimated, then, the likelihood of each pixel belonging to this function is computed.

This information is used to classify each incoming pixel according to a prior-defined classification rule. Statistical modeling has been approached numerous times throughout the development of computer vision applications [13], as they offer a good compromise between useful results and resource consumption. The main disadvantage that the method presents is the (often large), memory used to store historical data. It is possible to minimize memory consumption if the image information is semantically compressed, stored, classified, and then parsed back to pixel-level data. This approach should offer us small memory footprint while maintaining robustness commonly not found on non-statistical segmentation techniques.

In this work we present a statistical method for background modeling that prioritizes two factors: minimum data storage and robust real-time processing. The proposed algorithm is part of a full embedded smart surveillance system that is deployed in in-door environments. The system's goal is to track a limited number of objects and ultimately identify those that are left unattended at certain areas of interest.

The remainder of this paper is organized as follows, Section 2 presents related work, Section 3 discusses the main ideas behind the statistical background modeling algorithm, and Section 4 presents and analyzes results obtained with this method. Finally, in Section 5 conclusions are presented.

## 2 Related Work

Statistical Background Modeling is a broad topic. It spans multiple techniques, commonly implemented as software algorithms running on personal computers. Some authors systematically classify statistical-based methods in three categories: the first category includes *Single Gaussian*, *Mixture of Gaussians* and *Kernel Density Estimation*. A second category mainly involves *Support Vector* techniques; finally, a third category features *General Gaussian Theory* along matrix-based approaches [10].

Most of these methods offer reliable results, however, only a sub-set of these algorithms are eligible for real-time implementation on embedded

platforms. An example of an embedded statistical-based algorithm is found on [14]. The authors introduce a real-time algorithm that maps each input pixel value to a histogram built from a number of past frames. The histogram serves as an approximation of the pixel's color probability distribution through time. The background model is obtained from each histogram, and can be subtracted from the current input image every processing step. A total of five bins are used for every histogram constructed.

Another embedded implementation is presented on [15]. The algorithm is proposed with two variants: The first one operates on the *Red-Green-Blue* (RGB) color space while the second one operates at a 255-bit grayscale level. The approach is based on the popular algorithm developed by [16], based on Mixture of Gaussians (MoG). The basic idea is to model the color values of each pixel as a mixture of Gaussian distributions. The background model is, thus, built with the most persistent intensity values presented over time. This allows for composite and dynamic surfaces to be modeled, where a pixel can describe more than one texture at a given frame.

The traditional MoG model sets a maximum number of Gaussian Distributions; however, texture conditions on a complex region of interest can vary. Authors in [17] introduce an *Adaptive MoG* algorithm that dynamically creates and removes Gaussians distributions on the fly. The algorithm is set with an initial number of Gaussians. During running time, an update procedure checks every Gaussian weight in the mixture model. If a weight is smaller than a threshold and the current number of distributions is below a minimum preset, a new distribution is created. Otherwise, the Gaussian has expired and must be removed from the model.

Individual pixel distribution modeling can be computationally expensive. One possible method that minimizes memory consumption is found on [18]. The approach involves describing pixels not as distributions but as compressed statistical ranges depicted with intensity clusters. Each incoming pixel is matched against a list of existing clusters. If a positive match is found, the relevant cluster self-updates in order to include more persistent values, effectively grouping pixels with similar intensity data.

An alternative statistical model is introduced by [19]. In this paper, the background segmentation method is presented as a machine learning algorithm. The algorithm builds an initial statistical model during training. The constructed model is consulted by each incoming pixel during the classification phase. This approach offers a low memory and low latency approach for embedded implementations.

The algorithm works in the RGB color space. Each background pixel is modeled using four parameters: *expected color value*, *color standard deviation*, *brightness distortion* and *chromaticity distortion*. Using these parameters is possible to identify a single color under different light conditions, making the algorithm robust against luminance noise. After parameter computation, each pixel can be classified as background or foreground according to a simple classification rule.

In this paper we developed a background modeling implementation that is part of a full embedded surveillance system. We use a statistical approach focused on fixed-point precision and robust real-time processing. The complete system is deployed out-doors and monitors areas of interest with low to medium traffic. Images are acquired using a fixed-camera. Image processing is achieved on a 32-bit Intel Nios II Embedded Processor with DSP capabilities.

Natural light variations such as shadows and highlights are present and must be tolerated by the background model. The system is focused on real-time identification of static and moving objects that do not belong to the original scene.

A Standard Definition (SD) frame is preferred over High Definition (HD) images. The system processes frames of 320 x 240 pixels. This size is typically known as Quarter VGA (QVGA) resolution. QVGA frames lower storage consumption while also offering enough information for standard image processing tasks (e.g., blob detection and feature extraction based on object geometry).

## 3 Statistical Algorithm

### 3.1 Overview

The background model has been developed as a supervised pixel classifier. As such, the algorithm must first run through a training process prior to active classification. During training, the classifier builds a statistical model based on background information. Once training has been completed, historical data will be retrieved and used for new pixel classification. The model samples pixel data from an unknown population in order to build a reference model of the real population. The reference model will be used to further classify pixel color values according to their historical mean and standard deviation, final pixel classification will be decided using a set of rules defining four different pixel classes.

In this application, the estimated population correlates to the pixel grayscale intensity distribution gathered over time. The *Central Limit Theorem* (CLT) states that given a large enough sample size; it is possible to make inferences about the population mean regardless of its shape. The distribution of the sample mean must be approximated to normal distribution [20, 21].

The proposed algorithm estimates the population distribution for each sample average and stores it as the following tuple:  $N(\mu, \sigma)$ , (pixel mean and pixel standard deviation). The sample distribution will also show variance equal to the variance of the population divided by the sample size. The final model will be described with *imageWidth* x *imageHeight* tuples. For QVGA frames that's a total of  $320 \times 240 = 76,800$  tuples.

### 3.2 Pixel Distribution Model

Pixel color distribution has been known to follow a normal distribution in a static scene that evolves over time [16, 22]. Consider an image pixel  $p$  acquired by a camera and located at coordinates  $(x, y)$  on the screen. During a fixed time period of  $n$  frames acquired, pixel  $p$  will show variations due to illumination effects. Our model defines four specific illumination ranges where pixel  $p$  moves across time.

The first range is referred to as the *midtones* range.

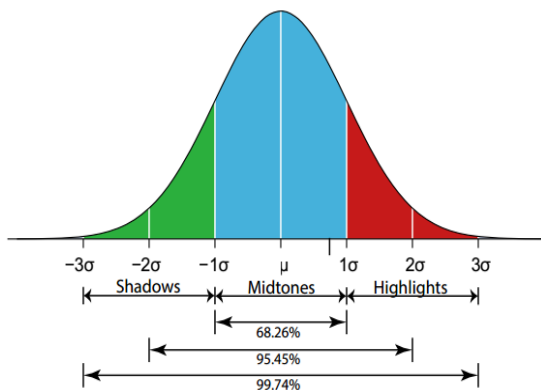


Fig. 1. Normal distribution  $N(\mu, \sigma)$  of a pixel  $p(x,y)$

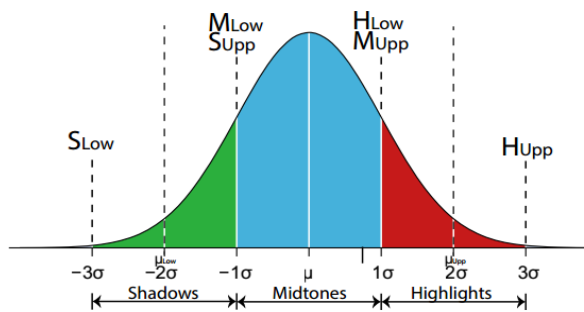


Fig. 2. Pixel Distribution with defining each tone range

Table 1. Sample size implemented as a fixed-point arithmetic shift

Sample Size	Arithmetic Right Shift
32	>>5
64	>>6
128	>>7

The mean intensity value of pixel  $p$  will be located in this range most of the time. The range below the mean value is named the *shadows* range. The following intensity range in the distribution is defined as the *highlights* range, and contains intensity values above the intensity mean of pixel  $p$ . A final range is defined outside of the three main possible ranges. This is the *foreground* range, where pixel intensity values differ drastically from previous gathered data.

Figure 1 shows the pixel normal distribution and the three proposed area ranges. Everything falling outside the  $[-3\sigma, 3\sigma]$ , interval will be labeled as a *foreground* object, not belonging to the original scene.

### 3.3 Sample Distribution with Unknown Population Parameters

For this particular application, both population mean and standard deviation are unknown quantities. Nevertheless, the algorithm must estimate a normal population with unknown mean  $\mu$  and unknown standard deviation  $\sigma$  using the sample mean  $\bar{X}$  and sample standard deviation  $S$ .

A typical procedure is to replace  $\sigma$  with the sample standard deviation  $S$  [21]. However, the sampling distribution of the sample mean will no longer be normally distributed. Instead of using the random variable  $Z$ , that is, the standard normal distribution, it is possible to use a distribution called the *t-distribution*. The effect of replacing  $\sigma$  by  $S$  on the distribution of the random variable  $T$  is negligible if the sample size  $n$  is large enough. Commonly, a sample size between 30 and 40 is considered large enough [23]. The larger the sample size, the closer the *t-distribution* is to the *z* distribution. The shape of the *t-distribution* is very similar to the *z-distribution*: the contour is bell-shaped, the mean is zero, but the spread is larger.

### 3.4 Sample Size

Sampling is carried out during the training phase. During this stage the classifier gathers a pixel sample of a defined size  $n$  for parameter estimation. The condition of  $n > 30$  is a common guideline if population distribution is not too far from normal [24]. As mentioned before, having an unknown population variance  $\sigma^2$  forces the algorithm to model the sample distribution as *t*-shaped. Nevertheless, the *t-distribution* resembles a normal distribution as sample size increases.

The choice of a meaningful sample size not only impacts on the classification results, but also consumes a fair amount of memory. During the training period, a number of frames equal to the sample size will be stored in system memory. Consider Table 1, where possible values of  $n$  are examined. It is possible to simplify some

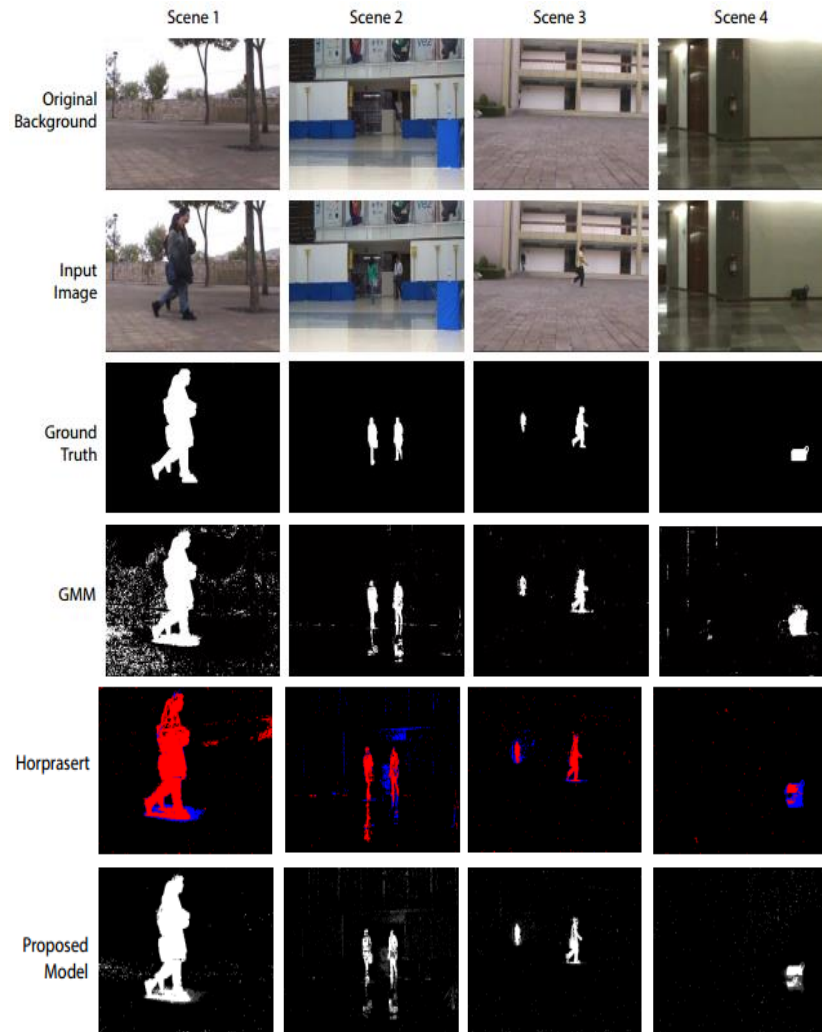


Fig. 3. Qualitative Results Benchmark No. 1

calculations if a sample size is chosen as a power of two (e.g., Sample mean  $\mu$  computation involves a division by  $n$ ). We have set the training sample size  $n$  to 32 ( $2^5$ ) frames.

### 3.5 Pixel Population Mean and Standard Deviation Estimation

After defining a sample size with unknown parameters, population computation can be now formulated. Population expected value  $\mu$  is estimated using an interval of plausible values.

The estimator comes in the form of a Confidence Interval (CI). This parameter is constructed so it offers high confidence that  $\mu$  lies directly within this numeric range. The mean interval is built using a confidence level  $(1 - \alpha)$  of 95 %.

The formal definition of a two-sided confidence interval on  $\mu$  is:

$$\bar{X} - \frac{S}{\sqrt{n}} t_{\frac{\alpha}{2}, n-1} \leq \mu \leq \bar{X} + \frac{S}{\sqrt{n}} t_{\frac{\alpha}{2}, n-1}, \quad (1)$$

where  $\bar{X}$  and  $S$  correspond to the mean and standard deviation of the random sample obtained from a normal distribution. Additionally, the variance  $\sigma^2$  is unknown and the factor  $t_{\alpha=2, n-1}$

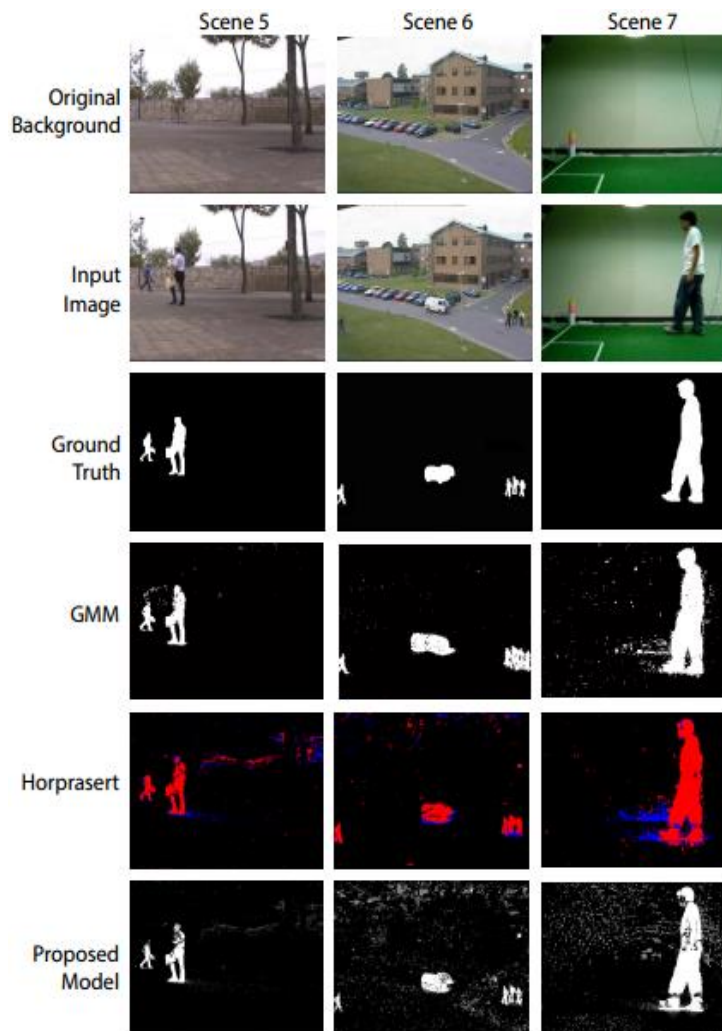


Fig. 4. Qualitative Results Benchmark No. 2

represents the upper percentage point of the t-distribution with  $n-1$  degrees of freedom [20]. For our particular problem, the variables are defined as follows:

$$1 - \alpha = 0.96 \rightarrow \alpha/2 = 0.025, \quad (2)$$

$$n = 32 \rightarrow n - 1 = 31, \quad (3)$$

$$\sqrt{n} = \sqrt{32} \sim 5.6568, \quad (4)$$

$$t_{\frac{\alpha}{2}, n-1} = t_{0.025, 31} \sim 2.04. \quad (5)$$

Finally, it is possible to estimate the parameter  $\sigma$  by relying on the sample data obtained previously. The standard deviation of the population will be estimated by multiplying the square root of the size of the sample  $n$  by the standard deviation of the sample.

### 3.6 Population Distribution Segmentation and Classification

Once each pixel distribution has been estimated, the background model should offer reference information of both the population mean  $\mu$  and

standard deviation  $\sigma$ . This information is stored in a pair of matrices, indexed as the incoming pixels require. The next step in the process is to divide up the whole population distribution in the delimited regions used for pixel identification.

During classification, each pixel is evaluated according to its historical statistics. Each pixel is fitted in four possible classes: Shadows, Midtones, Highlights and Foreground. Figure 2 depicts the segmented pixel distribution.

We define six values used as thresholds for pixel comparison. Note that some threshold values in fact overlap, thus, it is possible to compute only four values instead of six. Each new pixel is labeled according to the following classification rule:

$$Class(p) = \begin{cases} Shadows : \text{if } S_{lower} \leq p < S_{upper}, \\ Midtones : \text{if } M_{lower} \leq p \leq M_{upper}, \\ Highlights : \text{if } H_{lower} < p \leq H_{upper}, \\ Foreground : \text{otherwise.} \end{cases} \quad (6)$$

Additionally, each distribution threshold is computed using the  $\mu$  CI previously built.

The lower mean value ( $\mu_{Lower}$ ), can be used for lower thresholds and the upper mean value ( $\mu_{Upper}$ ) for upper thresholds. Both values can be either positive or negative. Each threshold value is computed as follows:

$$S_{lower} = -3\sigma + \mu_{lower}, \quad (7)$$

$$S_{upper} = M_{lower} = -1\sigma + \mu_{lower}, \quad (8)$$

$$M_{upper} = H_{lower} = 1\sigma + \mu_{upper}. \quad (9)$$

$$H_{upper} = 3\sigma + \mu_{upper}. \quad (10)$$

The prior operations are carried out during the classifier's training phase. Resulting data such as pixel standard deviation mean and threshold information for each image pixel is stored in RAM memory. The classification phase is straightforward, as the background model is queried by each new pixel as the classification rule is applied over the whole image.

## 4 Results

The algorithm is first written and debugged as an embedded Matlab algorithm. Embedded Matlab is a subset of the Matlab computing language that generates C code directly from Matlab scripts. It can also be used to generate a *Hardware Description Language* (HDL), template through Simulink. Further C target-platform optimizations are carried out manually. Matlab offers us great developing tools and a complete insight of background classifier's complexity running on an embedded environment.

We compare results obtained against two other methods often used as reference throughout literature: the Gaussian Mixture Model and the Horprasert algorithm. The latter has been developed by [25] and is used to benchmark of our own model. The entire database of test sequences includes seven video clips of small indoor and outdoor areas focused on low-traffic and varying environmental conditions (e.g., light, surfaces and objects). We have tested out-door and in-door scenes, where backgrounds can be either dynamic (i.e., moving objects such as trees are present) or static. Variable light is observed throughout all the videos, while traffic is labeled as low (1-3 persons) or medium (3-5 persons). Some scenes also feature reflective surfaces.

Figure 3 shows the first four scenes under testing, while Figure 4 depicts the last three scenes evaluated. The database used in the first five scenes is available online [26]. The sixth scene has been taken from the *PETS2001* database [27] and presents an out-door surveillance view of a parking lot where no clean background of the scene is available, thus, the system is trained with cars already parked.

The Gaussian Mixture Model is tested using the following parameters:  $\alpha = 0.005$ ,  $\rho = 0.05$  and three Gaussian distributions. The model does not include a shadow removal stage. MoG-classified pixels are colored white for foreground and black for background. The Horprasert decision parameters  $T_{CD}$ ,  $T_{a0}$ ,  $T_{a1}$  and  $T_{a2}$  are selected threshold values used to determine similarities in chromaticity and brightness between the background model and the current image [18].

Parameter selection can be automatic or manual. In this application, the thresholds that

**Table 2.** Quantitative comparison of PCC in % for each Background (BG) scene

Method	BG 1	BG 2	BG3	BG4	BG5	BG6	BG7
MoG	91.14	98.67	99.45	98.68	99.36	97.37	96.82
Horprasert	97.61	98.84	99.47	99.35	98.23	98.32	97.53
Proposed	98.51	99.17	99.71	99.56	99.23	99.17	98.23

achieved a better segmentation have been manually obtained through experimentation. The parameters used are:  $T_{CD} = 5000$ ,  $T_{alo} = -20$ ,  $T_{a1} = 6$  and  $T_{a2} = -6$ . Horprasert's model classes are rendered in three colors: red for foreground objects, blue for shadows and black for background pixels. Our method's results are shown using four different colors: white for foreground, gray for shadows, light gray for highlights and black for background pixels. All results are presented without any post-processing. Figure 3 and Figure 4 depicts the complete qualitative results of the input test database.

For quantitative evaluation, all seven video clips results are compared using the Percentage of Correct Classification (PCC), defined as follows:

$$PCC = \frac{TP+TN}{TPF} \times 100. \quad (11)$$

In Equation (11),  $TP$  (true positives), represents the number of correctly classified foreground pixels;  $TN$  (true negatives) represents the number of correctly classified background pixels and finally,  $TPF$  represents the total number of pixels in a QVGA image.

The *Ground Truth* image is used as reference for measuring both the  $TP$  and  $TN$  values.

Table 2 depicts all the PCC values obtained for the seven tested scenes under the three different discussed methods. The maximum value of PCC is 100%, which indicates a result identical to the ground truth reference image. A lower result indicates a deviation from the ground truth.

All methods are compared based on foreground and background classification; this means that shadows (or highlights) identified by the Horprasert model and our approach are rendered as background pixels for evaluation.

All methods are implemented on a PC with an Intel Core i3-2350M CPU running at 2.30 GHz with

4 GB of RAM running at a minimum of 30 frames per second. Estimated profiling using Matlab tools show a processing rate of 30 frames per second (average) with a system memory consumption of maximum 2.07 Mbits for background model storing.

Input is received as a sequence of 62 images comprising the video clips used in scenes 1, 6 and 7. The script has been separated in three main blocks: training, background model building and classification. The training block computes and stores the cumulative sum, minimum and maximum values of an incoming grayscale pixel for 32 frames. The background model building block computes the standard deviation and mean of all processed pixels. Finally, the classification block receives a grayscale pixel and outputs its classification label during 30 frames.

The final output of the full automatic surveillance system is shown in Figure 5. The system receives an input frame from the fixed-position camera. The background model constructs the Foreground Mask, where objects that do not belong to the original background are rendered in white. Shadows have been filtered out by the pixel classifier; this mask is then processed by a blob detection stage. Tracked blobs of interested are finally framed with a rectangle.

#### 4.1 Discussion of the Results

Although our approach is developed mainly for indoor surveillance, we also tested performance under out-door conditions with reasonable reability. In the first scene we can observe the presence of dynamic multi-textured pixels (i.e., moving trees) among other environmental noises that hinder optimal segmentation. MoG achieves the best segmentation in the fifth scene, classifying correctly the dynamic pixels around the trees while



optimally preserving the shape of the detected foreground objects.

Scene two and four present the noisiest environments in the database. Both sequences feature reflective surfaces and irregular light distorted by moving objects.

The reflection of the target objects (the persons walking through the hall) is identified in all three methods as entirely new foreground objects. In the second scene, MoG manages to filter out most of the noise produced by the composite surfaces (such as the building reflective walls and the blue moving fabric), while the floor reflections are better segmented using both the Horprasert model and our model.

While evaluating the third scene, the three techniques achieve a good segmentation with low background noise and good shape preservation. Enough shadow information is filtered out by Horprasert's model and our approach, with no significant differences in both results. The sixth scene produces a lot of highlight and shadow noise with Horprasert's model and our method. As mentioned before, this sequence has heavy light variations throughout the span of the video clip.

The seventh and final scene presents an interesting scenario, with uneven light and numerous shadows sources. MoG presents a lot of environmental noise; meanwhile most of the shadows are correctly identified by the Horprasert model. Our model also accurately identifies shadows and highlights present in the frame.

## 5 Conclusions

Background modeling is a typical computer vision task; however, algorithm implementations aimed for real-time embedded applications are limited. In this paper we proposed a background modeling algorithm with emphasis on real-time processing and memory savings. The background modeling algorithm relies on statistical information and is aimed for a fast implementation. It is developed as a supervised pixel classifier, where a background reference model is built during a training stage using a video sequence of a fixed background acquired by a stationary camera. Once trained, the system can actively classify the pixels of an input

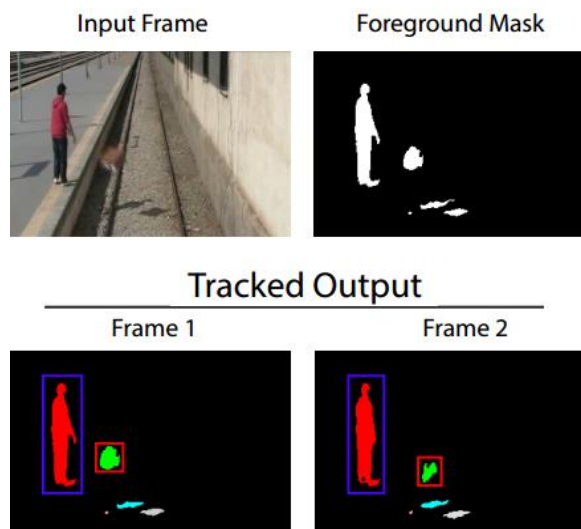


Fig. 5. Final output of the surveillance system

image in four different classes: background, shadows, highlights and foreground.

We compared the results obtained with two techniques that are often implemented on embedded platforms: the Gaussian Mixture Model and the Horprasert model. According to the tests conducted, the overall performance of our method is in-line with results obtained directly from software implementations. The impact of fixed-point operations in the proposed architecture is practically negligible while comparing with traditional statistical-based techniques.

Best classification results are obtained when the system monitors in-door areas with non-reflective surfaces, as reflective surfaces will introduce noise that can lead to miss-classified pixels. Under these circumstances noise can be minimized with the implementation of post-processing filters such as Gaussian, median, erosion and dilation. System accuracy can be generally boosted up by decimating the input stream of images. This will cause each  $N(\mu, \sigma)$  tuple value to be extended to accept broader intensity values, reducing the possible noise obtained during certain scenarios.

## References

1. Toyama, K., Krumm, J., Brumitt, B., & Meyers, B. (1999). Wallflower: Principles and practice of

- background maintenance. *International Conference on Computer Vision (ICCV)*, pp. 255–261. DOI: 10.1109/ICCV.1999.791228.
2. **Wu, M. & Peng, X. (2010).** Spatio-temporal context for codebook-based dynamic background subtraction. *AEU-International Journal of Electronics and Communications*, Vol. 64, No. 8, pp. 739–747. DOI: 10.1016/j.aeue.2009.05.004.
  3. **Gangodkar, P., Kumar, D., & Mittal, A. (2013).** Robust segmentation of moving vehicles under complex outdoor conditions. *IEEE Transactions on Intelligent Transportation Systems*, Vol. 13, No. 4, pp. 1738–1752. DOI: 10.1109/TITS.2012.2206076.
  4. **Cheung, S. & Kamath, C. (2004).** Robust techniques for background subtraction in urban traffic video. *Proceedings of Video Communications and Image Processing, SPIE Electronic Imaging*, Vol. 5308, pp. 881–892. DOI: 10.1117/12.526886.
  5. **Acevedo-Avila, R., Gonzalez-Mendoza, M., & Garcia-Garcia, A. (2016).** A Linked List-Based Algorithm for Blob Detection on Embedded Vision-Based. Vol. 16, No. 6, pp. 782. DOI:10.3390/s16060782.
  6. **Mao, D., Mould, J., & Subramanian, S. (2007).** Background subtraction for realtime tracking of a tennis ball. (*VISAPP'07*), *Proceedings of the Second International Conference on Computer Vision Theory and Applications*, pp. 427–434.
  7. **El-Baf, F., Bouwmans, T., & Vachon, B. (2009).** Fuzzy statistical modeling of dynamic backgrounds for moving object detection in infrared videos. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 60–65. DOI: 10.1109/CVPRW.2009.5204109.
  8. **El-Baf, F., Bouwmans, T., & Vachon, B. (2008).** Type-2 fuzzy mixture of gaussians model: Application to background modeling. *International Symposium on Visual Computing, (ISVC'08)*, pp. 772–781. DOI: 10.1007/978-3-540-89639-5\_74.
  9. **Karaman, M., Goldmann, L., Yu, D., & Sikora, T. (2005).** Comparison of static background segmentation methods. *Visual Communications and Image Processing*, Vol. 5960. DOI:10.1117/12.633437.
  10. **Bouwmans, T. (2009).** Subspace Learning for Background Modeling: A Survey. *Recent Patents on Computer Science*, Vol. 2, No. 3, pp. 223–234.
  11. **Piccardi, M. (2004).** Background subtraction techniques: a review. *IEEE International Conference on Systems, Man and Cybernetics*. DOI: 10.1109/ICSMC.2004.1400815.
  12. **Hassanpour, M., Sedighi, H., & Manashty, A. (2011).** Video frame's background modeling: Reviewing the techniques. *Journal of Signal and Information Processing*, Vol. 2, No. 2, pp. 72–78. DOI: 10.4236/jsip.2011.22010.
  13. **Li, L., Huang, W., Yu-Hua, G. I. & Tian, Q. (2004).** Statistical Modeling of Complex Backgrounds for Foreground Object Detection. *IEEE Transactions on Image Processing*, Vol. 13, No. 11, pp. 1459–1472. DOI: 10.1109/TIP.2004.836169.
  14. **Juvonen, M. P. T., Coutinho, J. G. F., & Luk, W. (2007).** Hardware architectures for adaptive background modelling. *3rd Southern Conference on Programmable Logic, (SPL '07)*, pp. 149–154. DOI: 10.1109/SPL.2007.371739.
  15. **Appiah, K. & Hunter, A. (2005).** A single-chip fpga implementation of real-time adaptive background model. *FPT*, pp. 5–102, DOI:10.1109/FPT.2005.1568531.
  16. **Stauffer, C. & Grimson, W. (1999).** Adaptive Background Mixture Models for Real-Time Tracking. *Computer Vision and Pattern Recognition*, Vol. 2, pp. 2246–2252.
  17. **Wei, Z., Li, P., & Yue, H. (2015).** A Foreground-Background Segmentation Algorithm for Video Sequences. *International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, pp. 340–343. DOI: 10.1109/DCABES.2015.92.
  18. **Makarov, A. (1996).** Comparison of background extraction based intrusion detection algorithms. *International Conference on Image Processing*, pp. 521–524. DOI: 10.1109/ICIP.1996.559548.
  19. **Horprasert, T., Harwood, D., & Davis, L. S. (1999).** A statistical approach for real-time robust background subtraction and shadow detection. *Proceedings of the IEEE Frame-Rate Applications Workshop*, pp. 1–19.
  20. **Montgomery, D. & Runger, G. (2006).** *Applied Statistics and Probability for Engineers*. John Wiley & Sons.
  21. **Moore, D. & McCabe, G. (2002).** *The Introduction to the Practice of Statistics*. Freeman.
  22. **Power, P. & Schoonees, J. (2002).** *Understanding background mixture models for foreground segmentation*. Vol. 267.
  23. **Wegner, T. (2012).** *Applied Business Statistics: Methods and Excel-based Applications*.
  24. **Triola, M. (2012).** *Statistics for describing, exploring, and comparing data. Elementary Statistics Technology Update*. Pearson, Addison-Wesley.
  25. **De Jesus, M. (2012).** Robust background subtraction and shadow detection. [http://www.markyd13.com/code/background\\_subtraction/](http://www.markyd13.com/code/background_subtraction/)

26. **Acevedo, R. (2013).** Smart Surveillance Video Database. <https://github.com/gone-still/imgDB/tree/master/resources>
27. **Performance Evaluation of Tracking and Surveillance (PETS) (2001).** <ftp.pets.rdg.ac.uk>

*Article received on 13/01/2017; accepted on 17/02/2017.  
Corresponding author is Ricardo Acevedo-Avila.*