IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 27, 2019, accepted October 14, 2019, date of publication October 18, 2019, date of current version October 30, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2948321

Background-Subtraction Algorithm Optimization for Home Camera-Based Night-Vision Fall Detectors

MERCEDES ALONSO, ALBERTO BRUNETE, (Member, IEEE), MIGUEL HERNANDO, (Member, IEEE), AND ERNESTO GAMBAO, (Member, IEEE)

Centre for Automation and Robotics (CAR UPM-CSIC), Universidad Politécnica de Madrid, 28012 Madrid, Spain

Corresponding author: Alberto Brunete (alberto.brunete@upm.es)

This work was supported in part by the Robocity2030 III and IV Programs supported by the Comunidad de Madrid and European Commission.

ABSTRACT Background subtraction is one of the key pre-processing steps necessary for obtaining relevant information from a video sequence. The selection of a background subtraction algorithm and its parameters is also important for achieving optimal detection performance, especially in night environments. The research contribution presented in this paper is the identification of the optimal background subtractor algorithm in indoor night-time environments, with a focus on the detection of human falls. 30 background subtraction algorithms are analyzed to determine which has the best performance in indoor night-time environments. Genetic algorithms have been applied to identify the best background subtraction algorithm, to optimize the background subtractor parameters and to calculate the optimal number of pre- and post-processing operations. The results show that the best algorithm for fall-detection in indoor, night-time environments is the LBAdaptativeSOM, optimal parameters and processing operations for this algorithm are reported.

INDEX TERMS Fall detection, camera-based, background-subtraction.

I. INTRODUCTION

The risk of falling is one of the most prevalent problems faced by elderly individuals. A study published by the World Health Organization [1] estimates that between 28% and 35% of people over the age of 65 suffer at least one fall each year, and this figure increases to 42% for people over 70. According to the World Health Organization, falls represent greater than 50% of elderly hospitalizations and approximately 40% of non-natural mortalities for this segment of the population. Falls are a significant source of mortality for elderly individuals in developed countries. Falls are particularly dangerous for people that live alone because of the amount of time that can pass before they receive assistance. Approximately one-third of the elderly (those over the age of 65) in Europe live alone [2], and the elderly population is expected to increase significantly over the next twenty years.

The fall detection system proposed by Fallert [3] is based on a low-cost device comprising an embedded computer and a camera. Installed into walls or ceilings, this device monitors a room without human intervention. Thus, people monitored at home are not required to wear devices, and the system is capable of 24 h monitoring. Fallert's fall detection system works relatively well (over 96% accuracy) during daylight, but performs poorly at night because of the lack of light. To solve this problem, the inclusion of an infra-red emitter and a camera without an IR filter were required. Improvements to the background subtractor algorithm used previously [3] were required because of poor performance under night-time conditions.

Background subtraction is important for many image processing problems and has been extensively studied [4]. Several different background subtraction approaches are compared in [5]; however, the methods studied in this comprehensive review fall short when applied to infra-red video images. This paper analyses the performance of the background subtraction algorithms presented in [5] on night images taken with infra-red light, with the aim of selecting algorithms able to work with infra-red video. Pre- and post-processing improvements to background subtraction

The associate editor coordinating the review of this manuscript and approving it for publication was Jagdish Chand Bansal.

accuracy were considered, and a genetic algorithm was implemented for optimal parameter selection. The authors of [5] applied hand-fitted parameters for each of the algorithms, a technique that is relevant for many applications, from traffic cameras to fall-detection of elderly people. In contrast, in this paper, background subtraction parameters of several algorithms are optimized using a genetic algorithm, in order to compare them and select the best background subtraction algorithm for detecting falls in night-time video images.

This article is structured as follows: previous work and state of the art in background subtractor algorithms are discussed in sections II and III. The description of the Fallert system is presented in section IV. The methodology is described in section V, and the results and discussion in section VI. Finally the conclusions are summarized in section VII.

II. PREVIOUS WORK

Fall detection technologies can be divided into three categories, as explained by [6]: wearable sensors, ambient sensors, and vision-based technologies.

Most wearable sensors are based on accelerometers and gyroscopes that a person must wear or carry [7]–[11]. Some sensors use a mobile phone as a primary [12] or accessory device [13] to detect falls. These systems have a fundamental problem: they are based entirely on the person carrying the device. On the contrary, ambient sensors and vision-based technologies work independently from any user action. Most currently available commercial devices for fall detection are portable. In fact, the top 10 fall detectors listed on the website ''toptenreviews'' [1] are based on portable devices.

Ambient sensors monitor the environment continuously to detect falls. There are several approaches: technologies based on the interruption of some type of beam or set of beams of infra-red light (laser or not) [14], [15], 3D cameras [16], WiFi signal strength analysis [17], and sound [18], [19] and vibration [20], [21] detection.

Because of their applicability, vision-based systems are one of the most interesting approaches. They rely on artificial intelligence algorithms to analyze images or video taken from cameras. Following the discussion presented in [6], vision-based approaches are focused on real-time execution of a detection algorithm using standard computing platforms and low-cost cameras. There are several methods used to obtain semantic information through video analysis. Many such methods make use of a 2D or 3D model, while others are based on the extraction of some features after video image segmentation of the body. A more detailed explanation of those approaches can be found in [6] where they are classified into the following categories: body and shape change, posture detection, inactivity, spatiotemporal, and 3D head change.

In addition, two types of cameras are mainly used for fall detection: 2D cameras (like the one used in this paper or in [22], and 3D time of flight (ToF) cameras as discussed in [23] and [24]. Although ToF cameras provide more information, they have worse resolution and are more expensive; as a result, traditional cameras remain particularly attractive.

A common feature of most vision-based systems is the use of a background-subtractor algorithm. The segmentation of relevant scene information is a common first step for many computer vision algorithms. The most basic techniques involve subtraction of a background image or registration of scene changes between frames. More advanced algorithms, such as the background subtraction algorithm developed by [25], register the most common colors of each pixel and update learned data over time, consequently exhibiting adaptive capabilities responsive to scene changes over time.

The recognition of specific features can also be used to extract relevant information from a scene. Feature descriptor algorithms, such as histograms of oriented gradients, can be trained to identify certain features of the human body. One example of this application can be found in [26], [27], where a subject's head and body are independently followed, and accurate readings of their relative trajectories over time are found.

Due to their importance, many background-subtractor algorithms (included in the background-subtractor library `BGSLIbrary`) were studied in [5] under daytime conditions. The current article presents an analysis of background subtractors on indoor, night-time images. Furthermore, the use of background subtractors is focused on a specific application (the detection of falls), while in [5], analysis is performed for more general applications. A more detailed review of background-subtractor algorithms can be found in section III.

III. REVIEW OF THE BACKGROUND-SUBTRACTION ALGORITHMS UNDER ANALYSIS

In the previous section, the importance of the background subtractor algorithms was explained. The background subtractor library (`BGSLIbrary`) includes many algorithms, some of which were analyzed and compared in [5]. The purpose of this paper is to reanalyze these algorithms (as well as some more recent algorithms), to find the most suitable background-subtractor algorithm to detect human falls under night-time conditions. All background-subtractor algorithms can be found in https://github.com/andrewssobral/bgslibrary. The complete list of the algorithms is shown in Table 1.

Most of the algorithms listed in Table 1 require one or more input parameters. Parameter choice is critical to algorithm performance; thus, we used a genetic algorithm to find optimal parameters in order to obtain the best results with each of the algorithms.

In this section, we present an overview and brief analysis of the algorithms.

A. BASIC METHODS, MEAN AND VARIANCE WITH TIME

Methods based on basic functions usually rely on a single parameter. Briefly, in these methods each frame is compared with a frame selected as the true background, and the

[1] https://www.toptenreviews.com/health/senior-care/best-fall-detection-sensors/

TABLE 1. Algorithms compared.

Algorithm Name	Reference
Basic methods, mean and variance with time	
Frame Difference	
Static Frame Difference	
Weighted Moving mean	
Weighted Moving Variance	
Adaptative Background Learning	
Adaptative Selective Background Learning	
DPMean	
DPAdaptative Median	[28]
DPPrati Mediod	[29]
Statistical methods using a single Gaussian	
DPWren GA	[30]
LBSimple Gaussian	[4]
Statistical methods using multiple Gaussians	
DPGrimson GMM	[31]
DPZivkovic GMM	[32]
Mixture of Gaussian V2	OpenCV
Methods based on eigenvalues and eigenvectors	
DPEigenbackground	[33]
Local Binary Pattern (LBP)	
LbpMrf	[34]
SJN Multiclue	[35]
Local Binary Similarity Pattern (LBSP)	
SuBSENSEBGS	[36]
LOBSTERBGS	[37]
Methods based in fuzzy logic	
Fuzzy Sueno integral	[38]
Fuzzy Choque Integral	[39]
LBFuzzy gaussian	[40]
Methods based in type-2 fuzzy logic	
T2-FGMM-UM	[41]
T2-FGMM-UV	[41]
T2-FMRF-UM	[42]
T2-FMRF-UV	[42]
Neural and fuzzy-neural methods	
LBAdaptative SOM	[43]
LBFuzzy Adaptative SOM	[44]
Others	
Independent Multimodal	[45]
VuMeter	[46]
KDE	[47]

difference computed. The true frame is usually selected as the first frame, or the mean of the frames to that point.

B. STATISTICAL METHODS USING A SINGLE GAUSSIAN

In algorithms based on a single Gaussian model, each pixel is modelled with a probability function defined by its mean and standard deviation. With this Gaussian model, the probability of a pixel being background or foreground can be obtained. This method is more robust when illumination changes occur.

The DPWren GA algorithm, which models people in the image and the background, and the LB Simple Gaussian algorithm, which updates the mean and standard deviation with time belong to this category.

C. STATISTICAL METHODS USING MULTIPLE GAUSSIANS

One of the most popular background detection techniques is based on a parametric adaptive mixture of models. This model was first presented in [31], and improved in [48]. In this algorithm, the values of each pixel are modeled by a mixture of Gaussians. These distributions are generally updated using an algorithm of mean minimization that improves the use of the Gaussian distribution (see [30] an [4]). Every time a new frame is processed, the mixtures of Gaussians for each frame are updated.

D. METHODS BASED ON EIGENVALUES AND EIGENVECTORS

The only algorithm considered here that is based on eigenvalues and eigenvectors is the DPEigenbackgroundBGS method ([33]). This method builds an eigenspace that models the background. The method includes characteristics of surrounding pixels to obtain a more precise description of each pixel.

E. LOCAL BINARY PATTERN (LBP)

The Local Binary Pattern method is a gray-scale invariant presented in the early 1990s. In the original case (see [49]), the original 3 x 3 pixel neighborhood is thresholded by the value of the central pixel. The values of the pixels in the thresholded neighborhood are multiplied by weights given to the corresponding pixels. Finally, the values of the surrounding pixels are summed to obtain the value of this texture unit.

F. LOCAL BINARY SIMILARITY PATTERN (LBSP)

The LBSP is based on the definition of a new characteristic index defined by the following equations:

$$LBSP_R(x_c, y_c) = \sum_{p=0}^{P-1} d(i_p - i_c) \cdot 2^p \tag{1}$$

with

$$d(x) = \begin{cases} 1 & \text{if } |x| \leq T_d, \\ 0 & \text{if } |x| > T_d, \end{cases} \tag{2}$$

where T_d is a similarity threshold and i_c corresponds to the intensity of the central pixel (x_c, y_c), and i_p, the intensity of the pth pixel in the set of neighboring pixels P (see [50]).

G. METHODS BASED ON FUZZY LOGIC

The methods in this category use fuzzy logic in three different ways as described below.

First, the Fuzzy Sugeno Integral uses a fuzzy integral to fuse texture and color features for background subtraction (see [38]).

Second, in the Fuzzy Choquet Integral method (see [39]), background initialization is made by using the average of the first N video frames containing objects. An update rule applied to the background image is necessary for the algorithm to adapt well to the system over time. For this, a selective maintenance scheme is adopted.

Finally, the LBFuzzy Gaussian method (see [40]) uses a saturating linear function instead of a hard limiter (in the fuzzy background subtraction) to determine if the pixel belongs to the background or to the foreground.

H. METHODS BASED IN TYPE-2 FUZZY LOGIC

The first detector in this category is presented in [51] (see also [41], [52]) in two modes, T2-FGMM-UM and T2-FGMM-UV. Although both modes can be used to model the background, the T2-FGMM-UM is expected to be more robust than the T2-FGMM-UV.

In [42], the authors introduced spatial-temporal constraints into the T2-FGMM using the MRF (Markov Random Field), a framework to achieve superior modeling performance for dynamic backgrounds.

I. NEURAL AND FUZZY-NEURAL METHODS

Neural and fuzzy-neural methods use SOM (Self Organizing Maps) to detect the background and foreground (see [43], [44]. Each node of the SOM has an associated weight vector of the same dimension as the input data vector and a position in the map space. The nodes are usually organized in a hexagonal or quadrangular regular 2-dimensional grid. The SOM describes a mapping between the higher dimensional input state and the lower dimensional map. The process of placing an input vector in the map requires finding the node with the most similar weight (the closest one).

J. OTHERS

Algorithms that could not be placed in any of the previous categories are described in this section.

The Independent Multimodal Background Subtraction (IMBS) method discussed by [45] seeks fast and efficient background subtraction. The background model is computed through per-pixel on-line statistical analysis of a set L of N frames in order to achieve high computational speed. According to a sampling period P, the current frame I is added to L, thus becoming a background sample S_n, where $1 \leq n \leq N$.

The non-parametric model called *Vu Meter* [46] is based on a discrete estimation of the probability distribution. The key aspects of this method are the probabilistic model and the temporal update.

In [47] (see also [4]), a parzen-window estimator for each background pixel is proposed:

$$P(I_{s,t}) = \frac{1}{N} \sum_{i=t-N}^{t-1} K(I^j_{s,t} - I^j_{s,i}), \tag{3}$$

where K is the kernel (usually a Gaussian kernel), and N is the number of previous frames used to estimate P. A pixel is classified as an object when $P(I_t)$ is higher than a predefined threshold, i.e. when the probability it belongs to the background is low.

K. FINAL CONSIDERATIONS ABOUT THE BACKGROUND-SUBTRACTORS

As pointed out at the beginning of the section, most of the algorithms discussed above were studied in [5] comparing real and computer-generated videos. In that extensive work, a broad comparison including detector accuracy, execution time, and CPU and memory requirements showed that there is not a perfect algorithm, and performance is highly dependant on the application.

The algorithms studied by [5] were evaluated based on car detection in outdoor, daytime scenes, with different weather conditions. In contrast, our algorithm comparison is based on the detection and tracking of people inside a house. In addition, we are interested in night vision, which is not considered in [5].

In order to focus tests of background subtractors (available in the library) on night-time videos taken inside a house, a metric was designed by removing the D-score index used by [5]. This D-score index weighs the importance of a pixel depending on its location in relation to the contour of the object. However, in our case the shape of the object is not critical to indoor, night-time conditions, and the D-score index could lead to misleading results.

In [5], the parameters of the algorithms were adjusted by manually searching for the best results. In our work, we systematically compare algorithms using a set of test cases and optimize parameters using a genetic algorithm.

FIGURE 1. Fall detection system prototype.

IV. DESCRIPTION OF THE FALLERT SYSTEM

The detection system used in this paper (called ''Fallert'' and shown in Fig. 1) was originally developed to be executed on a low-cost embedded computer. Several options were taken into account, and the Raspberry Pi board was chosen due to its sound technical characteristics, widespread adoption, and low price. In addition to the board (version 3B+ currently), the camera module designed for Raspberry Pi was used, which connects to the Raspberry Pi board via the CSI (Camera Serial Interface) port, thereby requiring significantly fewer CPU (Central Processing Unit) resources than a regular USB camera. An IR-LED ring is placed around the camera to provide IR illumination at night. The Fallert system also includes a case (designed specifically and printed with a 3D printer), an SD card, and a power supply. The electronic diagram can be found in Fig. 2.

This prototype is a fully capable, independent fall detection system with an estimated cost of less than 80€. The system connects to the internet using the built-in WiFi adapter of the

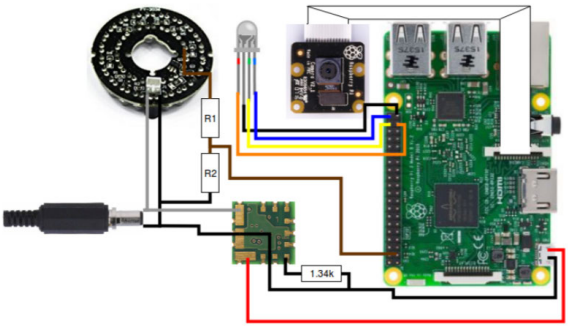


FIGURE 2. Electronic sketch of the fall detector.

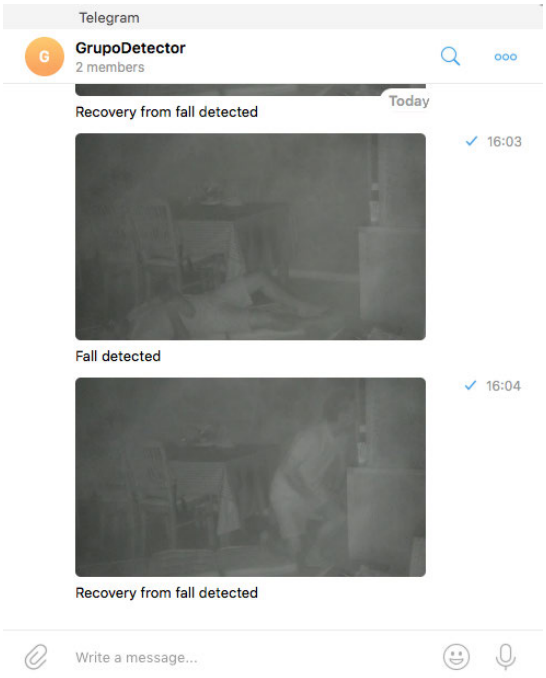


FIGURE 3. Example of a telegram alarm message.

Raspberry Pi and sends a message (email or Telegram) when a fall has been detected (as in Fig. 3). The message includes an image of the fall. If the person recovers, another message is sent.

Most currently available commercial devices for fall detection are portable. Regarding vision-based systems, similar products to Fallert exist, for example, "Carecams"[2], an online system based on IP cameras. However, these products employ fall detection algorithms run in a server outside the camera, where powerful computers can be used. In contrast, all image processing and fall detection in the Fallert system are performed in the Raspberry Pi.

V. METHODOLOGY

After reviewing the background subtraction algorithms in section III, the methodology used to evaluate each algorithm and to tune its parameters is presented in this section.

[2]https://www.carecams.co.uk/peace-of-mind-cameras

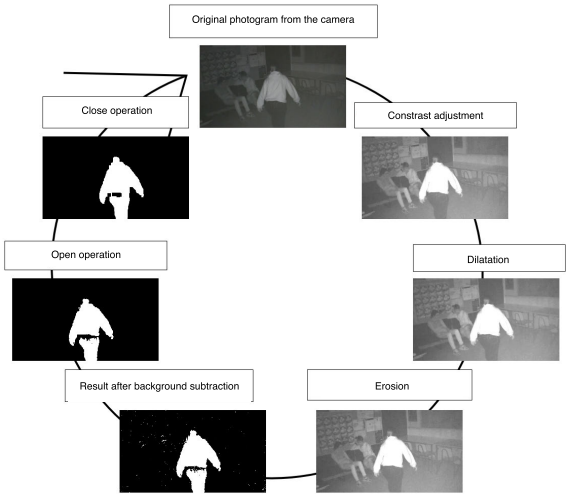


FIGURE 4. Steps of the detection process including the pre and post-operations applied to each frame.

The effectiveness of a background subtraction algorithm depends on the tuning of its parameters. Each algorithm has different parameters and to test all combinations can be a time-consuming task. To simplify this task we have developed a genetic algorithm to select the best combination of parameters for each algorithm and to compare the performance of the different algorithms for night conditions in home environments.

The genetic algorithm uses a fitness function that objectively measures algorithm performance for our task. This function is also used to choose the best parameter combinations for each algorithm. In addition, to get the best performance of each algorithm, we have also included a series of pre- and post-processing operations to the input frames and the algorithm output in order to improve the background subtraction operation. All these computer operations require significant amounts of time, therefore, the hardware used for computation is an important factor in this process. It is important to take into account that not all the algorithms take the same amount of time to process the video, and this can be a limitation in some applications.

Summarising, the methodology includes the following steps: (1) for each video, a set of pre-processing operations are performed; (2) the background subtraction algorithm is executed with the chosen parameters; and (3) some post-processing operations are performed. The genetic algorithm optimizes the number of pre- and post-processing operations and the values of the parameters of the background subtractors.

A. IMAGE PRE- AND POST-PROCESSING

In order to maximize the results of the background subtraction, a set of common operations were performed in each frame. The image processing cycle can be seen in Fig. 4. These operations, which are simple pixel transformations that

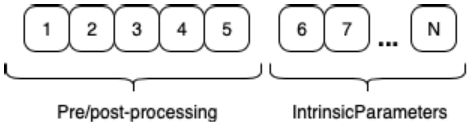


FIGURE 5. Gene codification for each background detection algorithm.

help to remove possible noise or increase the efficiency of the method, are as follows:

- Contrast: Used to adjust the image contrast.
- Dilation: This operation consists of convoluting an image A with some kernel (B). The shape of the kernel is set to be a circle. This operation makes the white object bigger and is used to make the foreground (usually clearer than the background) bigger and thus easier to analyze.
- Erosion: This operation computes a local minimum over the area of the kernel in order to reduce noise.
- Open: This operation consists of erosion followed by dilation and is used to remove noise from the image after background detection.
- Close: This operation consists of a dilation followed by an erosion and is used to fill holes in the foreground figures that may have appeared after background detection.

The experiments and results presented in the next section clarify the role that these basic image operations play in increasing the effectiveness of the background subtractors. Each operation is done with the default parameters, but the number of times each operation is performed is optimized using a genetic algorithm designed for speed and simplicity.

B. GENETIC ALGORITHM

There are several alternatives for dealing with the optimization of the proposed algorithms. Other optimization algorithms such as Differential Evolution or Particle Swarm Optimization, could have also worked. However, considering the similarity between the parameters to be optimised and the genetic algorithm genes, the genetic algorithm was considered as the most appropriate.

The genetic algorithm was implemented using `MatLab`, and its `ga` (genetic algorithm) function. To codify the information to optimize, the following genetic sequence (Fig. 5) was used. The number of genes depend on the algorithm to optimize, as the gene codification includes the intrinsic parameters of the algorithm. Here, N represents the number of specific parameters of the algorithm minus 5 (the number of parameters used for the pre- and post-processing steps.)

The range for the pre- and post-processing parameters for all algorithms was set as shown in Table 2.

Each algorithm has a specific set of parameters (or, in some cases no parameters). Explanations of each parameter is beyond the scope of this paper. Optimization was initialized using either the default parameter values (set by the algorithm authors), or those selected in the review paper [5].

For the selection of the internal genetic algorithm parameters, we followed the method used in [53]. The objective is to obtain acceptable results with a reasonable simulation time. A set of experiments were carried out in order to determine these parameters. The experiments were done simultaneously using different computers to reduce the impact of the randomness present in the genetic algorithm. The simplest algorithm, Frame Difference, was used in order to save time in this process. An example of these experiments can be seen in Figure 6.

TABLE 2. Range for the pre- and post-processing operations.

Operations	minimum	maximum
Contrast	0	10
Dilation	0	15
Erosion	0	15
Open	0	25
Close	0	25

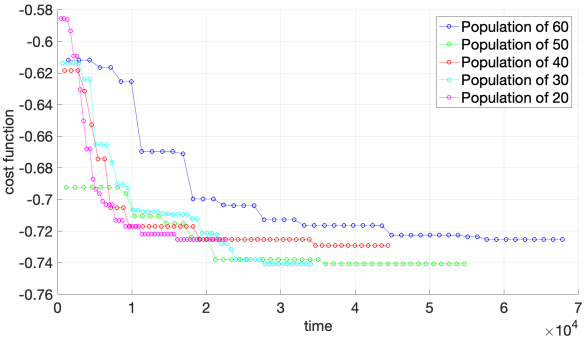


FIGURE 6. Comparison of the evolution of the genetic algorithm with a different number of generations for the Frame Difference algorithm with 0.7 crossover value.

Finally, the genetic algorithm parameters selected for the optimization were as follows:

- **FitnessLimit:** If the optimization function reaches this value (-1), the optimization will stop. It is the perfect value.
- **FunctionTolerance:** The algorithm will stop when the relative change in the fitness function of the last generations is equal or less than this value $(1e^{-1})$.
- **PopulationSize:** The number of individuals in each generation. After trials it was chosen 30.
- **CrossoverFraction:** The fraction of individuals in the next generation created with the crossover function. This value does not change during the execution of the optimization process. After trials it was chosen 0.6.

Apart from these parameters, the default genetic algorithm from MatLab was used, including the crossover and mutation functions.

C. FITNESS FUNCTION

The fitness function used is based on the one used in [5]. This metric is based on the comparison of the result obtained in a set of frames by the algorithm and the "ground truth". The ground truth is obtained manually for each frame we want to

TABLE 3. Comparison of pixels.

Category	Real value	Obtained value	Colour code
True Positive (TP)	1	1	white
False Positive (FP)	0	1	yellow
False Negative (FN)	1	0	red
True Negative (TN)	0	0	black

(a) Algorithm result (b) Ground truth

(c) Comparison with the color code explained in Table 3.

FIGURE 7. Comparison between the ground truth and the algorithm output for a certain frame. The color code of the result is explained in the Table 3.

compare. A first comparison is done pixel by pixel, obtaining 4 classes (see Table 3). An example of this is shown in Fig. 7.

With this pixel information, we use the following indexes:

- `precision (Pr)`: $TP/(TP + FP)$, results between [0,1].
- `Recall (Re)`: $TP/(TP + FN)$, results between[0,1].
- `F-Measure (Fm)`: $2 \cdot (Pr \cdot Re)/(Pr + Re)$, results between [0,1].

The fitness function also takes into account other indexes that treat the images as a whole, and not just the pixels alone (see [54]):

- `SSIM`: The Structural Similarity Index (SSIM) is used for measuring the similarity between two images, providing results in the range [0,1].
- `PSNR`: The Peak Signal-to-Noise Ratio is the ratio of the maximum possible power of a signal to the power of corrupting noise that affects the fidelity of its representation. An additional transformation is needed to change the dB results to the range [0,1].

Finally, the fitness function is the following:

$$F_{fitness} = -\frac{\sum_{i=0}^{N}\frac{Pr(i)}{N} + \sum_{i=0}^{N}\frac{Re(i)}{N} + \sum_{i=0}^{N}\frac{Fm(i)}{N}}{5} + \frac{\sum_{i=0}^{N}\frac{SSIM(i)}{N} + \sum_{i=0}^{N}\frac{PSNR(i)}{N}}{5} \quad (4)$$

where N is the number of frames to analyze in the video.

(a) Algorithm result

(b) Algorithm result

(c) Algorithm result

FIGURE 8. Example photograms from the different videos used in the genetic algorithm.

D. HARDWARE USED

To execute the experiments and simulations, several computers were used, and their main features are listed in Table 4.

The videos were taken with a Raspberry Pi model 3B and a camera Pi NoIR. This camera does not have an infra-red filter, allowing us to record videos without visible light (night videos). An infra-red LED ring was used to emit infra-red light as explained in Section IV. The videos were recorded in very different scenarios in order to have as general results as possible. Some examples of photograms used in the algorithm can be seen in Fig. 8.

VI. RESULTS AND DISCUSSION

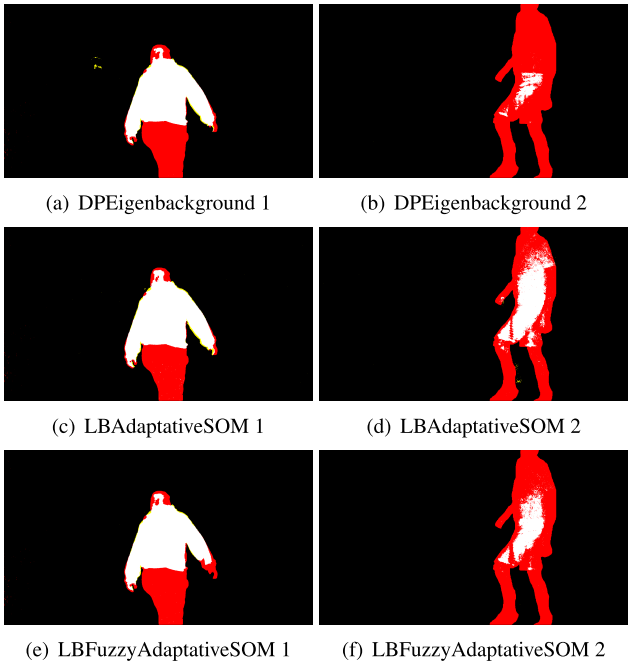
All results are given in Table 9. In this section, we explain and analyze the results.

TABLE 4. Computers used to execute the simulations.

Properties	Computer 1	Computer 2	Computers 3/4	Computer 5
Laptop/ PC	PC	PC	PC	Laptop
Operating System	Ubuntu 16.04	Ubuntu 14.04	Windows 7 Enterprise	Ubuntu 16.04
RAM	16 GB	8 GB	4 GB	8 GB
Type of Intel® Core™processor	i7-4790K 4.00 GHz x8	i5-4460 3.2GHz x 4	i5-3470 3.20 GHz x4	i7-3632QM 2.2 GHz x8
Graphic card	GeForce GTX 960/PCIe/SSE2	Intel® Haswell Desktop	Intel® HD Graphics	Intel® Ivybridge Mobile

TABLE 5. Top 5 algorithms with default parameters.

Algorithm	Fitness function result
LB Adaptative SOM	0.7321
LBFuzzy Adaptative SOM	0.6809
DPEigenbackground	0.6739
SJN-Multiclue	0.6229
SUBSENBGS	0.6059



(a) DPEigenbackground 1 (b) DPEigenbackground 2

(c) LBAdaptativeSOM 1 (d) LBAdaptativeSOM 2

(e) LBFuzzyAdaptativeSOM 1 (f) LBFuzzyAdaptativeSOM 2

FIGURE 9. Best 3 results without optimization. The color code is explained in the table 3.

To determine the limits of the detection cost value, we obtain results when the algorithm detects everything as either foreground or background:

- Everything detected as foreground: 0.3082
- Everything detected as background: 0.5683

Therefore, the value 0.5683 is considered as the lowest valid value for the algorithms.

A first test was made using the default parameters (the ones set in the paper [5] or in the code description), and results from the best-performing algorithms are shown in Table 5. An example of the detection in two particular frames is shown in the Figure 9.

TABLE 6. Top 2 algorithms after the optimization of the internal parameters.

Algorithm	Fitness function result
LB Adaptative SOM	0.822
LBFuzzy Adaptative SOM	0.8232

TABLE 7. Default parameters and optimized for LBAdaptativeSOM.

Parameter	Default value	Optimized value
Sensitivity	75	55
Training sensitivity	245	222
Learning rate	62	40
Training learning rate	255	250
Training steps	55	62

TABLE 8. Final parameters for LBAdaptativeSOM.

Parameter	value	range
Sensitivity	53	[50, 90]
Training sensitivity	247	[200, 300]
Learning rate	63	[40, 80]
Training learning rate	252	[250, 255]
Training steps	56	[25, 70]
Contrast	2.5967	[0, 10]
Dilation	1	[0, 15]
Erosion	3	[0, 15]
Open	21	[0, 25]
Close	5	[0, 25]

Consequently, the intrinsic algorithm parameters were optimized using the genetic algorithm (pre- and post-parameters were not optimised at this stage). The best results with this optimization are shown in Table 6. The results after the optimization are never worse than those obtained with the default parameters, as expected. We found that accurately tuning the intrinsic parameters of each algorithm is critical to performance, as clearly shown by the results presented in Fig. 10. The default and optimized intrinsic parameters are given in Table 7. Tests showed that after several executions the genetic optimization for this type of algorithms yielded similar results with small, non-significant variations in the adjustments. Therefore, the results were considered robust enough to draw conclusions.

Finally, pre- and post-parameters were optimised for the best algorithm of the previous phase (LBAdaptativeSOM), obtaining a fitness value of 0.8877. The result is shown in Fig. 11. From the initial result of the algorithm without optimization (0.7321) to the final result, the optimization has improved the result in a 21%.

The parameters obtained after adding the pre- and post-parameters in the genetic algorithm are shown in Table 8,

TABLE 9. Results of the optimization of the background-subtractors.

Method	Parameter	Default value	Cost function	Range	optimised value without processing	Cost function	optimised processed parameters $(\alpha, \beta, \ldots)$	optimised value with processing	Cost function
Basic methods, mean and variance with time									
Frame Difference	umbral	15	-	[0,35]	13	0.5722	-	-	-
Static Frame Difference	umbral	15	0.5682	[0,35]	-		-	-	-
Wighted moving mean	umbral	15	0.5140	[0,35]	-	-	-	-	-
Wighted moving variance	umbral	15	0.4592	[0,35]	-	-	-	-	-
Adaptative Backgroung learning	umbral α	15 0.05	0.4790	[0,25] [0.025, 0.075]	- -	-	-	- -	-
Adaptative Selective Background learning	umbral α_{Learn} $\alpha_{Detection}$ LearningFrames	15 0.05 0.05 -1	0.481	[0,25] [0.025, 0.075] [0.025, 0.075] [-1, 25]	- - - -	-	-	- - - -	-
DPMean	umbral α learningframes	2700 $1\exp^{-6}$ 30	0.4698	[2600, 2800] [0, 0.01] [15, 45]	- - -	-	-	- - -	-
DPAdaptative Median	umbral samplingRate learningRate	40 7 30	0.4761	[20, 60] [0, 15] [15, 45]	- - -	-	-	- - -	-
DPPratiMediod	umbral samplingRate historySize weight	30 5 16 5	0.5511	[15,45] [0, 15] [10, 22] [0,15]	- - - -	-	-	- - - -	-
Single Gaussian									
DPWrenGA	umbral α learningframes	12.25 0.005 30	0.5342	[5, 25] [0, 0.1] [15, 45]	- - -	-	-	- - -	-
LBSimple Gaussian	sensitivity noisevariance learningRate	66 162 18	0.4896	[30, 90] [100, 200] [5, 30]	- - -	-	-	- - -	-
Multiple Gaussians									
DPGrimsom GMM	threshold alpha gaussians	9 0.01 3	0.4803	[0, 20] [0, 0.01] [1, 5]	- - -	-	-	- - -	-
DPZivkovik	threshold alpha gaussians	25 0.001 3	0.5055	[10, 40] [0, 0.01] [1, 5]	- - -	-	-	- - -	-
Mixture of gaussians V2	alpha umbral	0.05 15	0.4790	[0.025, 0.075 [0, 25]	0.056 10	0.4791	(0.478, 5, 2, 13, 14)	0.025 14	0.5683
Methods based on eigenvalues and eigenvectors									
DPEigenbackgroundBGS	umbral historysize embeddedDim	225 20 10	0.674	[200, 225] [10, 30] [10, 20]	200 10 10	0.6823	-	- - -	-
LBP									
LBPMrf	-	-	0.6017	-	-	-	-	-	-
SJN Multiclue	-	-	0.6229	-	-	-	(3.158, 3, 2, 25, 0)	-	0.6878

TABLE 9. *(Continued.)* **Results of the optimization of the background-subtractors.**

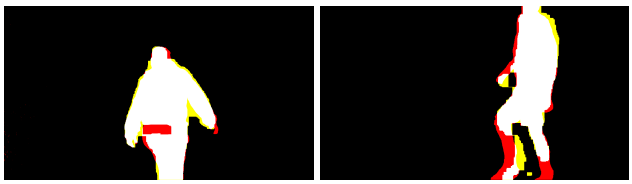
LBSP									
SubSENSEBGS	similarity threshold	0.333	0.6059	[0, 1]	0.9012	0.6192	-	-	-
	threshold offset	3		[1, 5]	4			-	
	min color threshold	30		[15, 45]	25			-	
	default bg samples	50		[25, 75]	53			-	
	required bg samples	2		[1, 5]	2			-	
	samples for AVGS	100		[80, 120]	98			-	
LOBSTERBGS	similarity threshold	0.365	0.4449	[0, 1]	-	-	-	-	-
	threshold offset	0		[0, 5]	-			-	
	default dist threshold	4		[0, 10]	-			-	
	default color threshold	30		[20, 40]	-			-	
	default bg samples	35		[25, 45]	-			-	
	required bg samples	2		[0, 5]	-			-	
Fuzzy									
Fuzzy Sugeno integral	framestolearn	10	0.5683	[0, 20]	-	-	-	-	-
	alphalearn	0.1		[0, 0.2]	-			-	
	alphaupdate	0.01		[0, 0.02]	-			-	
	colorspace	1		[1, 4]	-			-	
	option	2		[1, 2]	-			-	
	umbral	0.67		[0, 2]	-			-	
Fuzzy Choque integral	framestolearn	10	0.5683	[0, 20]	-	-	-	-	-
	alphalearn	0.1		[0, 0.2]	-			-	
	alphaupdate	0.01		[0, 0.02]	-			-	
	colorspace	1		[1, 4]	-			-	
	option	2		[1, 2]	-			-	
	umbral	0.67		[0, 2]	-			-	
LBFuzzy gaussian	sensitivity	72	0.4798	[50,90]	-	-	-	-	-
	bgthreshold	162		[100, 200]	-			-	
	learningrate	49		[20, 70]	-			-	
	noisevariance	195		[150,250]	-			-	
Type-2 Fuzzy based methods									
T2FGMM-UM	threshold	9	0.5683	[0, 20]	-	-	-	-	-
	alpha	0.01		[0, 0.02]	-			-	
	km	1.5		[0.5, 3]	-			-	
	kv	1.6		[0.5, 3]	-			-	
	gaussians	3		[1, 5]	-			-	
T2FGMM-UV	threshold	9	0.5283	[0, 20]	-	-	-	-	-
	alpha	0.01		[0, 0.02]	-			-	
	km	1.5		[0.5, 3]	-			-	
	kv	1.6		[0.5, 3]	-			-	
	gaussians	3		[1, 5]	-			-	
T2FMRF-UM	threshold	9	0.5126	[0, 20]	-	-	-	-	-
	alpha	0.01		[0, 0.02]	-			-	
	km	2		[0.5, 4]	-			-	
	kv	0.9		[0, 0.5]	-			-	
	gaussians	3		[1, 5]	-			-	
T2FMRF-UV	threshold	9	0.3082	[0, 20]	-	-	-	-	-
	alpha	0.01		[0, 0.02]	-			-	
	km	2		[0.5, 3]	-			-	
	kv	0.9		[0.5, 3]	-			-	
	gaussians	3		[1, 5]	-			-	
Neural and fuzzy-neural methods									

TABLE 9. *(Continued.)* **Results of the optimization of the background-subtractors.**

LBAdaptative SOM	sensitivity	75	-	0.4719	[50, 90]	55	-	-	0.8877
	trainingsensitivity	245			[200, 300]	222			
	learningrate	62	0.7321		[40, 80]	40	0.822	(2.5967, 1, 3, 21, 5)	
	trainiglearningrate	255			[250, 255]	250			
	trainingsteps	55			[25, 70]	62			
LBFuzzy Adaptative SOM	sensitivity	90			[7, 110]	59			0.8874
	trainingsensitivity	240			[200, 300]	207			
	learningrate	38	0.6809		[20, 50]	31	0.8232	(2.0927, 5, 9, 23, 8)	
	trainiglearningrate	255			[250, 255]	251			
	trainingsteps	81			[60, 90]	61			
others									
Independent Multimodal		-	-	0.4719	-	-	-	-	-
VuMeter	binsize	8	0.5617		[0, 15]	-	-	-	-
	alpha	0.995			[0.75, 1]	-	-	-	
	threshold	0.03			[0, 0.05]	-		-	
KDE	framestolearn	10	0.3082		[0, 20]	-	-	-	-
	sequencelearn	50			[25, 75]	-		-	-
	windowsize	100			[75, 125]	-	-	-	-
	th	$1\exp-8$			[0, 0.01]	-		-	-
	alpha	0.3			[0, 0.5]	-		-	-

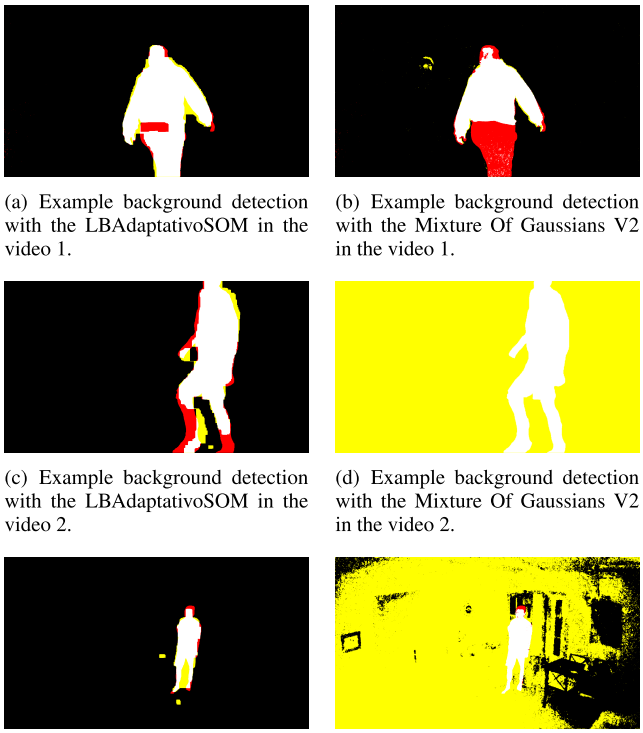
(a) (b)

FIGURE 10. Best result obtained with the parameters obtained with the genetic algorithm, using LBAdaptativeSOM algorithm and the parameters shown in Table 7. The color code is explained in the table 3.



(a) (b)

FIGURE 11. Best result obtained with the parameters optimized using the genetic algorithm (shown in Table 7) and additional processing, for the LBAdaptativeSOM algorithm. The color code is explained in the Table 3.



(a) Example background detection with the LBAdaptativoSOM in the video 1.

(b) Example background detection with the Mixture Of Gaussians V2 in the video 1.

(c) Example background detection with the LBAdaptativoSOM in the video 2.

(d) Example background detection with the Mixture Of Gaussians V2 in the video 2.

(e) Example background detection with the LBAdaptativoSOM in the video 3.

(f) Example background detection with the Mixture Of Gaussians V2 in the video 3.

FIGURE 12. Comparison between the results using the original background-subtractor algorithm Mixture Of Gaussians V2 and the completely optimized LBAdaptativoSOM algorithm. The color code is explained in the table 3.

along with the range used in the optimization process. Comparing the results in Tables 7 and 8 shows that the parameters of the algorithm change with and without the additional pre- and post-processing. We can conclude that

the parameters need to be re-optimized when additional pre- and post-processing is performed. The parameters optimized in the LBAdaptativeSOM algorithm are related to the learning process of the SOM and its sensibility.

To see how the evolution process helped the detection of background and foreground, the results obtained using the original background detector used in the Fallert System with those obtained using the optimized algorithm are compared in Figure 12. The original background subtractor selected, the `OpenCv` default background subtractor Mixture Of Gaussians V2, performs well in daylight conditions but fails in the night videos. Improvement to background/foreground detection with the optimized algorithm is clearly shown in Figure 12. From the initial result of the Mixture Of Gaussians V2 algorithm without optimization (0.4790) to the final result with the LBAdaptativeSOM algorithm (0.8877), the optimization and a different algorithm choice has improved the result in a 85.3%. Thus, the optimization process greatly improves results, even in night-time conditions, for a variety of scenarios.

Finally, to see the improvement of the GA in the algorithm, image 9 c), showing the initial algorithm without optimisation, can be compared with image 12 a), showing the result after the whole GA optimisation.

VII. CONCLUSION AND FUTURE WORK

In this paper, we analyze several background-subtractor algorithms to determine which has the best performance in detecting human falls in indoor, night-time environments.

The starting point of this research was the analysis of several background-subtractor algorithms (from the library `BGSLIbrary`) performed by [5]. These algorithms have been reanalyzed (adding recent algorithms), to find the most suitable background-subtractor algorithm to detect human falls in night-time conditions. In order to improve the detection process, a genetic algorithm was used to optimize background subtractor parameters and to select the optimal number of pre- and post-processing operations performed on images. Our results show that the use of a genetic algorithms can help to optimize artificial vision algorithms.

In conclusion, the best background subtractor for detecting falls in indoor, night-time environments is the LBAdaptativeSOM with the parameters shown in Table 8.

Future work will focus on testing the application in other home environments with a larger set of videos and falls.

ACKNOWLEDGMENT

The research leading to these results has received funding from RoboCity2030-DIH-CM, Madrid Robotics Digital Innovation Hub, S2018/NMT-4331, funded by "Programas de Actividades I+D en la Comunidad de Madrid" and co-financed by Structural Funds of the EU.

REFERENCES

[1] *WHO Global Report on Falls Prevention in Older Age*, World Health Organization, Geneva, Switzerland, 2007.
[2] R. Rodrigues, M. Huber, and G. Lamura, Eds., "Facts and figures on healthy ageing and long-term care," Eur. North Amer., Eur. Centre, Vienna, Austria, Occasional Rep. Ser. 8, 2012.
[3] K. de Miguel, A. Brunete, M. Hernando, and E. Gambao, "Home camera-based fall detection system for the elderly," *Sensors*, vol. 17, no. 12, p. 2864, 2017. [Online]. Available: http://www.mdpi.com/1424-8220/17/12/2864
[4] Y. Benezeth, P.-M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger, "Review and evaluation of commonly-implemented background subtraction algorithms," in *Proc. IEEE 19th Int. Conf. Pattern Recognit.*, Dec. 2008, pp. 1–4.
[5] A. Sobral and A. Vacavant, "A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos," *Comput. Vis. Image Understand.*, vol. 122, pp. 4–21, May 2014.
[6] M. Mubashir, L. Shao, and L. Seed, "A survey on fall detection: Principles and approaches," *Neurocomputing*, vol. 100, pp. 144–152, Jan. 2013.
[7] W. T. Kate, I. C. M. Flinsenberg, N. Chen, S. Kim, S. Schlumbohm, and R. B. M. Schonenberg, "Fall detection system," U.S. Patnet 8 408 041 B2, Apr. 2, 2013.
[8] W. Saadeh, M. A. B. Altaf, and M. S. B. Altaf, "A high accuracy and low latency patient-specific wearable fall detection system," in *Proc. IEEE EMBS Int. Conf. Biomed. Health Inform. (BHI)*, Feb. 2017, pp. 441–444.
[9] C. M. Chan, C. C. Cheung, K. L. Fan, and H. K. Tse, "Fall detection system," U.S. Patent 20 090 174 565 A1, Jul. 9, 2009.
[10] Y. Peng and S. Jin, "Fall detection system," U.S. Patent 8 381 603 B2, Feb. 26, 2013.
[11] I. C. M. Flinsenberg and W. R. T. T. Kate, "Fall detection system," U.S. Patent 8 749 391 B2, Jun. 10, 2014.
[12] J. Dai, X. Bai, Z. Yang, Z. Shen, and D. Xuan, "PerFallD: A pervasive fall detection system using mobile phones," in *Proc. 8th IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PERCOM Workshops)*, Mar./Apr. 2010, pp. 292–297.
[13] S. Abbate, M. Avvenuti, F. Bonatesta, G. Cola, P. Corsini, and A. Vecchio, "A smartphone-based fall detection system," *Pervasive Mobile Comput.*, vol. 8, no. 6, pp. 883–899, 2012.
[14] A. L. Cheng, C. Georgoulas, and T. Bock, "Fall detection and intervention based on wireless sensor network technologies," *Automat. Construct.*, vol. 71, pp. 116–136, Nov. 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0926580516300437
[15] S. Tao, M. Kudo, and H. Nonaka, "Privacy-preserved behavior analysis and fall detection by an infrared ceiling sensor network," *Sensors*, vol. 12, pp. 16920–16936, Dec. 2012.
[16] G. Mastorakis and D. Makris, "Fall detection system using Kinect's infrared sensor," *J. Real-Time Image Process.*, vol. 9, no. 4, pp. 635–646, 2014.
[17] H. Wang, D. Zhang, Y. Wang, J. Ma, Y. Wang, and S. Li, "RT-fall: A real-time and contactless fall detection system with commodity WiFi devices," *IEEE Trans. Mobile Comput.*, vol. 16, no. 2, pp. 511–526, Feb. 2017.
[18] X. Zhuang, J. Huang, G. Potamianos, and M. Hasegawa-Johnson, "Acoustic fall detection using Gaussian mixture models and GMM supervectors," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Apr. 2009, pp. 69–72.
[19] M. S. Khan, M. Yu, P. Feng, L. Wang, and J. Chambers, "An unsupervised acoustic fall detection system using source separation for sound interference suppression," *Signal Process.*, vol. 110, pp. 199–210, May 2014.
[20] M. Alwan, P. J. Rajendran, S. Kell, D. Mack, S. Dalal, M. Wolfe, and R. Felder, "A smart and passive floor-vibration based fall detector for elderly," in *Proc. 2nd Int. Conf. Inf. Commun. Technol.*, 2006, pp. 1003–1007.
[21] H. Rimminen, J. Lindström, M. Linnavuo, and R. Sepponen, "Detection of falls among the elderly by a floor sensor using the electric near field," *IEEE Trans. Inf. Technol. Biomed.*, vol. 14, no. 6, pp. 1475–1476, Jun. 2010.
[22] Y. W. Hsu, J. W. Perng, and H. L. Liu, "Development of a vision based pedestrian fall detection system with back propagation neural network," in *Proc. IEEE/SICE Int. Symp. Syst. Integr. (SII)*, Dec. 2015, pp. 433–437.
[23] G. Diraco, A. Leone, and P. Siciliano, "An active vision system for fall detection and posture recognition in elderly healthcare," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2010, pp. 1536–1541.
[24] M. Kepski and B. Kwolek, "Fall detection using ceiling-mounted 3D depth camera," in *Proc. Int. Conf. Comput. Vis. Theory Appl. (VISAPP)*, vol. 2, Jan. 2014, pp. 640–647.
[25] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction," in *Proc. 17th Int. Conf. Pattern Recognit. (ICPR)*, vol. 2, Aug. 2004, pp. 28–31.

[26] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, "Monocular 3D head tracking to detect falls of elderly people," in *Proc. Int. Conf. IEEE Eng. Med. Biol. Soc.*, Aug. 2006, pp. 6384–6387.
[27] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, "3D head tracking for fall detection using a single calibrated camera," *Image Vis. Comput.*, vol. 31, no. 3, pp. 246–254, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0262885612002107
[28] N. J. B. McFarlane and C. P. Schofield, "Segmentation and tracking of piglets in images," *Brit. Mach. Vis. Appl.*, vol. 8, no. 3, pp. 187–193, May 1995.
[29] S. Calderara, R. Melli, A. Prati, and R. Cucchiara, "Reliable background suppression for complex scenes," in *Proc. 4th ACM Int. Workshop Video Surveill. Sensor Netw.*, New York, NY, USA, 2006, pp. 211–214.
[30] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 780–785, Jul. 1997.
[31] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2, Jun. 1999, pp. 246–252.
[32] Z. Zivkovic and F. van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognit. Lett.*, vol. 27, no. 7, pp. 773–780, 2006.
[33] N. M. Oliver, B. Rosario, and A. P. Pentland, "A Bayesian computer vision system for modeling human interactions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 831–843, Aug. 2000. doi: 10.1109/34.868684.
[34] C. Kertész, "Texture-based foreground detection," *Int. J. Signal Process. Image Process. Pattern Recognit.*, vol. 4, no. 4, pp. 51–62, 2011.
[35] S. Noh and M. Jeon, "A new framework for background subtraction using multiple cues," in *Computer Vision—ACCV* (Lecture Notes in Computer Science), vol. 7726. Heidelberg, Germany: Springer, 2013, pp. 493–506.
[36] P.-L. St-Charles, G.-A. Bilodeau, and R. Bergevin, "Flexible background subtraction with self-balanced local sensitivity," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2014, pp. 414–419.
[37] P.-L. St-Charles and G. A. Bilodeau, "Improving background subtraction using Local Binary Similarity Patterns," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2014, pp. 509–515.
[38] H. Zhang and D. Xu, "Fusing color and texture features for background model," in *Fuzzy Systems and Knowledge Discovery*. Berlin, Germany: Springer, 2006, pp. 887–893.
[39] F. El Baf, T. Bouwmans, and B. Vachon, "Fuzzy integral for moving object detection," in *Proc. IEEE Int. Conf. Fuzzy Syst. (IEEE World Congr. Comput. Intell.)*, vol. 1, no. 1, Jun. 2008, pp. 1729–1736.
[40] M. H. Sigari, N. Mozayani, and H. R. Pourreza, "Fuzzy running average and fuzzy background subtraction: Concepts and application," *IJCSNS Int. J. Comput. Sci. Netw. Secur.*, vol. 8, no. 2, pp. 138–143, 2008.
[41] F. El Baf, T. Bouwmans, and B. Vachon, "Fuzzy statistical modeling of dynamic backgrounds for moving object detection in infrared videos," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2009, pp. 60–65.
[42] Z. Zhao, T. Bouwmans, X. Zhang, and Y. Fang, "A fuzzy background modeling approach for motion detection in dynamic backgrounds," in *Proc. Int. Conf. Multimedia Signal Process.*, 2011, pp. 177–185.
[43] L. Maddalena and A. Petrosino, "A self-organizing approach to background subtraction for visual surveillance applications," *IEEE Trans. Image Process.*, vol. 17, no. 7, pp. 1168–1177, Jul. 2008.
[44] L. Maddalena and A. Petrosino, "A fuzzy spatial coherence-based approach to background/foreground separation for moving object detection," *Neural Comput. Appl.*, vol. 19, no. 2, pp. 179–186, 2010.
[45] D. Bloisi and L. Locchi, "Independent multimodal background subtraction," in *Computational Modelling of Objects Represented in Images: Fundamentals, Methods and Applications*. Sep. 2012, pp. 39–44. doi: 10.1201/b12753-8.
[46] Y. Goya, T. Chateau, L. Malaterre, and L. Trassoudaine, "Vehicle trajectories evaluation by static video sensors," in *Proc. IEEE Intell. Transp. Syst. Conf.*, Sep. 2006, pp. 864–869.
[47] A. Elgammal, R. Duraiswami, D. Harwood, and L. S. Davis, "Background and foreground modeling using nonparametric kernel density estimation for visual surveillance," *Proc. IEEE*, vol. 90, no. 7, pp. 1151–1163, Jul. 2002.
[48] E. Hayman and J. O. Eklundh, "Statistical background subtraction for a mobile observer," in *Proc. 9th IEEE Int. Conf. Comput. Vis.*, vol. 1, Oct. 2003, pp. 67–74.
[49] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern Recognit.*, vol. 29, no. 1, pp. 51–59, 1996.
[50] G.-A. Bilodeau, J.-P. Jodoin, and N. Saunier, "Change detection in feature space using local binary similarity patterns," in *Proc. Int. Conf. Comput. Robot Vis.*, May 2013, pp. 106–112.
[51] T. Bouwmans and F. El Baf, "Modeling of dynamic backgrounds by type-2 fuzzy Gaussians mixture models," *MASAUM J. Basic Appl. Sci.*, vol. 1, no. 2, pp. 265–276, 2009.
[52] F. El Baf, T. Bouwmans, and B. Vachon, "Type-2 fuzzy mixture of Gaussians model: Application to background modeling," in *Advances in Visual Computing*. Berlin, Germany: Springer, 2008, pp. 772–781.
[53] I. Kucukkoc, A. D. Karaoglan, and R. Yaman, "Using response surface design to determine the optimal parameters of genetic algorithm and a case study," *Int. J. Prod. Res.*, vol. 51, no. 17, pp. 5039–5054, Feb. 2015.
[54] A. Hore and D. Ziou, "Image quality metrics: PSNR vs. SSIM," in *Proc. Int. Conf. Pattern Recognit.*, Aug. 2010, pp. 2366–2369.

MERCEDES ALONSO received the bachelor's degree in industrial electronics and automation from the Universidad Politécnica de Madrid, in 2017, where she is currently pursuing the master's degree in automation and robotics. She is also collaborating with the Centre for Automation and Robotics. Her research interests include computer vision and machine learning.

ALBERTO BRUNETE received the M.S. degree in telecommunication engineering and the Ph.D. degree in robotics and automation from the Universidad Politécnica de Madrid (UPM), in 2000 and 2010, respectively. He was the Technical Manager of the Research Center for Smart Buildings and Energy Efficiency (CeDInt), UPM, and a Visiting Professor with Carlos III University. He is currently an Associate Professor with the UPM and a Researcher with the Centre for Automation and Robotics, UPM. His main research interests include service robots and smart environments. In 2016, he has received the Spanish prize ABC Solidario of a fall detector project for elderly people.

MIGUEL HERNANDO received the M.S. degree in electrical engineering degree and the Ph.D. degree in automatic control and robotics from the Universidad Politécnica de Madrid (UPM), in 1997 and 2003, respectively. He has participated in several international and national research and development projects on robotics. He is one of the founders of the startup Biicode whose aim is to give (C/C++) to a dependency manager (currently Conan.io). He is currently a Professor with the Department of Electronics and Automatic Control, UPM, where he is currently a Researcher with the Centre for Automation and Robotics (CAR), CSIC. His research interests include motion path planning, service robots, and micro-robotics.

ERNESTO GAMBAO received the M.S. degree in electrical engineering and the Ph.D. degree in automatic control and robotics from the Universidad Politécnica de Madrid (UPM), Madrid, Spain, in 1996. He is currently a Professor with the Department of Automatic Control (UPM), where he is also a Researcher with the Centre for Automation and Robotics (CAR), CSIC. He has participated and coordinated numerous national and international research and development projects on robotics and automation. His research interests include collaborative robotics, service robots, and micro-robotics. He serves as a Research Project Reviewer for the European Commission.