ISSN 1870-4069

Embedded System for Human Detection Applied to Domotics

Oscar Arturo González González[1], Alina Mariana Pérez Soberanes[1], Víctor Hugo García Ortega[1], Julio César Sosa Savedra[2]

[1] Instituto Politécnico Nacional, ESCOM, Ciudad de México, México
[2] Instituto Politécnico Nacional, CICATA, Querétaro, México
oscar.ar-56@hotmail.com, psoberanes_mariana@hotmail.com
vgarciao@ipn.mx, jcsosa@ipn.mx

Abstract. This paper presents the development of an embedded system prototype that performs the home security monitoring, through image processing and classification algorithms to detect a human presence. If a human presence is detected, the system will send an alert message to the user. The embedded system is implemented on a Raspberry Pi 3 B, supported by a Pyroelectric Infrared Radial (PIR) motion sensor and a Raspberry Pi Camera V2. The algorithms are implemented in C language and were designed to take advantage of the hardware resources of the platform, through High Performance Computing (HPC) techniques. The selected classifier is a multilayer perceptron. This classifier obtained an accuracy of 96 %, approximately.

Keywords: image processing, domotics, embedded system, multilayer perceptron, HPC, Raspberry Pi.

1 Introduction

Statistics from the nonprofit organization *Observatorio Nacional Ciudadano* in Mexico indicates that only in the period March - April 2019, there was an increase of burglary investigation processes from 6,734 to 7,017 [1], that is an increase of 4.20%, in the number of investigation processes from March to April 2019. The situation of burglary in Mexico generates the need for people to keep their home safe. A possible solution to this problem is to develop new technologies that allow home automation. These technologies focus on the automation of processes within a building. This improves the quality of life and comfort, inside the home. It also allows constant monitoring of the place, through the Internet and the use of alarms, sensors or surveillance cameras. This domotics development has driven the creation of security systems such as those that are installed and monitored by private companies, which usually have a monthly cost per service, or commercial devices that can be installed and configured by the user. Some of these alternatives tend to be inefficient, and some others are often expensive or cannot be obtained by the majority of the population.

There are some papers proposed related to the home security monitoring and human detection. All these applications are developed in a Raspberry Pi platform and are supported by a camera and by sensors. The first paper *A New IoT Combined Body Detection of People by Using Computer Vision for Security Application* [2] presents a system developed on the Raspberry Pi platform with sensor elements: Raspberry Pi camera and a Pyroelectric Infrared Radial (PIR) sensor, which is used for motion detection. Once the image is obtained, the algorithm of the Histogram of Oriented Gradients (HOG) extracts the object's features and a Support Vector Machine (SVM) is used for the classification and learning of the human form. If a human presence is detected, the image is sent to a smartphone over the Internet, using the Telegram application. The artificial vision tool used for the development of this system is OpenCV. Regarding the results of the proposed method, it is presented that it was obtained a detection rate of 93.89 $\pm$5.3%.

The paper *Human Detector and Counter Using Raspberry Pi Microcontroller* [3] presents a system that identifies humans entering a room through a door. The system performs digital image processing using HOG as a feature extraction algorithm and a SVM for the classification, based on the OpenCV tool and developed in the Python programming language. It is specified that it is supported on a Raspberry Pi and uses a Raspberry Pi Camera, other sensors are a pair of PIR sensors, to indicate to the system the capture of input or output images; and a Bluetooth module (HC-05) to send the required data to another system through serial communication. It is has an efficiency of 83 %. In addition, it is mentioned that the use of Raspberry Pi platform avoided the use of desktop CPU and as a result it was reduced the system power consumption and the cost.

In this paper it is proposed an embedded system for home security monitoring, which is implemented in a Raspberry Pi 3 B platform using HPC techniques, thereby taking advantage of the hardware and software resources of the platform. The system detects the human presence and sends an alert message to the user. This allows users to keep their home safe through the remote monitoring. The system uses a motion sensor to perform the motion detection. The images are obtained from Raspberry Pi Camera V2. Also the system implements image processing and classification algorithms. This feature allows the system to improve its robustness and to perform a reliable detection of human presence inside the enclosure. Additionally, the system is considered to be scalable as a security module of a robust home automation system. This system can include the control of doors, windows, temperature, among other possible modules. The general diagram of the proposed system is presented in Fig. 1.

2 Methodology

Raspberry Pi is a suitable platform for home automation systems. This is the selected platform for the implementation of the system. In [4] is mentioned that Raspberry Pi is a platform that supports a large number of input and output peripherals. Raspberry Pi is also suitable for interacting with different devices,

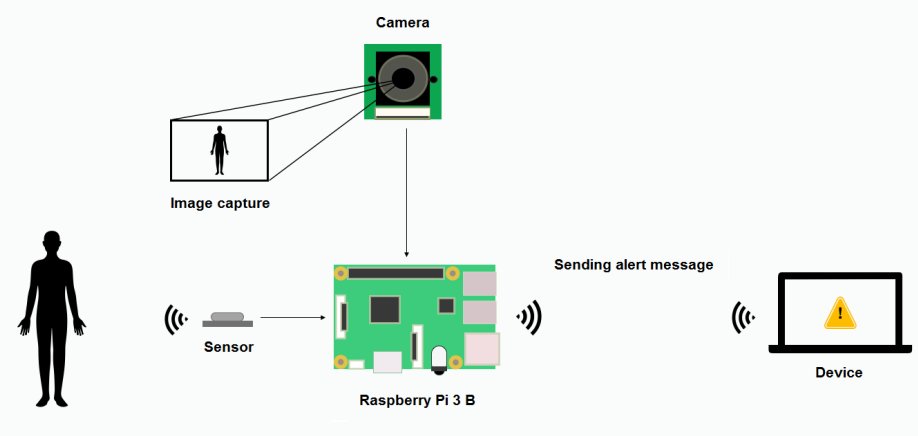

Fig. 1. Proposed system.

allowing its use in a wide range of applications. On the other hand, the ability to access the Internet makes the Raspberry Pi suitable for home automation related applications.

The system is composed of four main modules:

1. **Motion detection module**: In this module the embedded system through the AS312 PIR sensor detects movement and the Raspberry Pi Camera Module V2 perform the acquisition of images.
2. **Image processing module**: In this module the image subtraction for motion detection is performed. Then the algorithms for the image segmentation and the image feature extraction are performed.
3. **Classification module**: In this module the descriptor vectors obtained from the image feature extraction are analysed by the classification algorithm.
4. **Communication module**: This module sends a notification to the user's device to notify the possible human presence.

Each of the previous modules will be described in the following sections.

2.1 Motion Detection Module

For the motion detection it was implemented a circuit for the AS312 sensor, which is powered with 3.3 V. The circuit output depends on the activation of the AS312 sensor, if the sensor detects movement the output will be high (3.3 V), otherwise if the sensor does not detect movement, the output will be low (0 V). The sensor has a detection distance of 4 meters and a viewing angle of 60° approximately. The reading of the circuit output is performed through the General Purpose Terminals (GPIO) of the Raspberry Pi. This reading allows to send the signal to the system indicating if movement was detected or not.

This signal is read through the WiringPi library, which is a GPIO access library written in C language for the BCM2835, BCM2836 and BCM2837 System on Chip (SoC) devices used in all Raspberry Pi platforms [6]. If movement is detected the system will capture an image that will be processed by the next system module. This image capture is performed through the Raspberry Pi Camera Module V2, which is connected to the Raspberry Pi platform through the Camera Serial Interface (CSI) and is controlled by the *raspistill* commands of the operating system of the Raspberry Pi (Raspbian Stretch).

2.2 Image Processing Module

This module is divided into 4 submodules:

A. Image acquisition, grayscale conversion and noise filter application. The image obtained by the Raspberry Pi camera V2 is in BMP format -Bitmap-, which is a bitmap file, these kind of files are uncompressed and have 24 bits per pixel, also this kind of image do not have any type of processing of data, or alteration, so the original data is preserved [5]. Once the BMP image is obtained a grayscale conversion, and then a Gaussian filter is applied to the image. This filter allows to attenuate the noise and has a better frequency response. For its implementation, a convolution operation is performed with a mask, with a window size of 3x3 (2). Equation (1) is used to obtain the mask and the value of σ is 1.0:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad , \tag{1}$$

$$G = \begin{pmatrix} 0.0751 & 0.1238 & 0.0751 \\ 0.1238 & 0.2042 & 0.1238 \\ 0.0751 & 0.1238 & 0.0751 \end{pmatrix} \quad . \tag{2}$$

To avoid using the floating point unit by the Raspberry Pi processor, an approximation of (2) is obtained in (3):

$$G = \frac{1}{16} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} \quad . \tag{3}$$

This mask or kernel is multiplied by each pixel and its neighbors, in order to homogenize the colors of nearby neighbors. This achieve the elimination of white dots or white noise in an image, because the noise usually takes different pixel values from those in a neighborhood of pixels.

The Broadcom BCM2837 processor of the Raspberry Pi 3 B card has four cores, so the processing can be divided into four processing blocks. The Gaussian filter is implemented by dividing the processing in parallel into four processes

working on the same image (data parallelism). This allows to take advantage of the resources provided by the platform. The implementation of this Gaussian filter and all the algorithms that will be presented in the next sections are built in C language, which is a programming language that allows the compiler to generate the assembly code directly from the processor.

B. Motion detection in an image. Once the processing of the previous modules was applied to a base image - which is captured at the first process of the execution of the system -, this image is set as a background. Then if motion is detected by the sensor movement a subsequent image will be captured. The motion detection in the image is performed through implementing the background subtraction method, which is performed with the subtraction operation between the two images, with this operation is obtained the pixel differences between two images and thus be able to identify if there are differences between one scenario and another. Mathematically, the subtraction can be represented as $f = x - y$, where x and y are images respectively.

The simplest case to perform the subtraction and not to obtain negative values is to apply the absolute value function (4):

$$f_a = |x–y| \ . \tag{4}$$

Considering that the subtraction has the property of being commutative. The result of the subtraction between the two images must be a positive value. For this the value obtained is verified, if it is negative, the result of the subtraction will be replaced by the value of 0, otherwise the value obtained from the subtraction is assigned.

C. Image segmentation. This submodule is dedicated to the subdivision and analysis of image regions for segmentation. The region corresponding to the object of interest is represented with a pixel value of 255, while the region belonging to the background of the image adopts a pixel value of 0. First, the Otsu segmentation algorithm is performed to obtain an automatic global threshold depending on the analysis of the image pixels, it requires three parameters: the image obtained from the motion detection module, its width and height. With this operation the threshold is obtained and then it will be used in the image thresholding. The Otsu method requires the values of the image histogram, and are used to perform the consequent operations to get the maximum value. The Otsu method allows segmentation of an image through discriminant analysis, separating the image into two classes C_0 and C_1 this classes correspond to object and background. This method automatically finds the threshold of an image using its gray segmentation levels [7]. Class values take values $C_0 = \{0, 1, ..., u\}$ where u is the gray level and $C_1 = \{u+1, u+2, ..., L-1\}$ where L is the number of gray levels. To obtain the probability of occurrence of a gray level P_i in a digital image the following expression is used, where N is the number of pixels in gray level of the image and f_i is the number of frequency of pixels with the

same gray level (5):

$$P_i = \frac{f_i}{N} \ . \tag{5}$$

The classes C_0 and C_1 are given by (6) and (7):

$$C_0 : \frac{P_1}{\omega_0(u)}, ..., \frac{P_u}{\omega_0(u)} \ , \tag{6}$$

$$C_1 : \frac{P_{u+1}}{\omega_1(u)}, ..., \frac{P_L}{\omega_1(u)} \ . \tag{7}$$

Equations (8) and (9) define ω_0 and ω_1:

$$\omega_0(u) = \sum_{i=u+1}^{L} P_i \ , \tag{8}$$

$$\omega_1(u) = \sum_{i=u+1}^{L} P_i \ . \tag{9}$$

Subsequently, the average for each class is obtained, given by μ_0 and μ_1 (10), (11).

$$\mu_0 = \sum_{i=1}^{u} \frac{i * P_i}{\omega_0(u)} \ , \tag{10}$$

$$\mu_1 = \sum_{i=u+1}^{L} \frac{i * P_i}{\omega_1(u)} \ . \tag{11}$$

Once the means of the classes are obtained, the variance is obtained (12):

$$\sigma_B^2 = \omega_0 * (\mu_1 - \mu_T)^2 + \omega_1 * (\mu_2 - \mu_T)^2 \ , \tag{12}$$

where μ_T is the average obtained from the total pixels of the image in gray levels. The threshold t^* is selected by maximising the separation of classes from the image and the coefficient between the variance between classes is maximised. The greater the variance between classes and the smaller the variance within them, the discrimination will be better between the groups (13):

$$t^* = MAX(\sigma_B^2(u)) \quad 1 \leq u \leq L \ . \tag{13}$$

Subsequently, the thresholding of the image is performed using the threshold value obtained with the Otsu method. To do this, we proceed to verify if the value of the pixel of the image to be thresholded is greater or less than the desired threshold and depending on this, modify its value in 0 or 255 (14):

$$f(i,j) = \begin{cases} 255 & \text{si } l > u, \\ 0 & \text{si } l < u. \end{cases} \tag{14}$$

To perform the image threshold, it is required to define a threshold value u. If the gray level value l of an image $f(i,j)$ is greater than the threshold u.

After the Otsu threshold, the image is partitioned into subimages of specific size. In this subimages the pixel values are ordered and if the neighborhood is adequate, the area is dilated. The pixel values of the subimage are saved, considering the binarized image obtained with the Otsu method and once the values are sorted, a value of 0 or 255 is assigned to the analyzed pixel depending on the relationship that it has with the neighbor pixels, related with the pixel average. This processing of the white pixels allows to obtain a better image of the section that belongs to the object. This operations require further processing, thus operations are divided by four processes to take advantage of the hardware and software resources.

Finally, the Sauvola method is used to complete thicken regions in the image in addition to correcting shadows that are not related to the object. This method is used to reduce light intensities in an image. This method uses the average intensity of the values of a given region of pixels and their standard deviation. This region is defined within a bxb neighborhood or matrix, with a k parameter which varies between 0.2 and 0.5. This is implemented in the equation (15) obtained from [8]:

$$T(x,y) = m(x,y)\left[1 + k\left(\frac{\sigma(x,y)}{R} - 1\right)\right] \quad , \tag{15}$$

where:

- $\sigma(x,y)$: Is the standard deviation of the neighborhood.
- R: Is maximum value of the standard deviation, $R = 128$ with the maximum intensity being 255 in a neighborhood with $b = 15$.
- k: A value between 0.2 and 0.5.
- $m(x,y)$: Arithmetic mean of the neighborhood.
- $T(x,y)$: Value of the threshold in the neighborhood.

The processing result images of image subtraction and the image segmentation submodule are presented in the following Fig. 2.

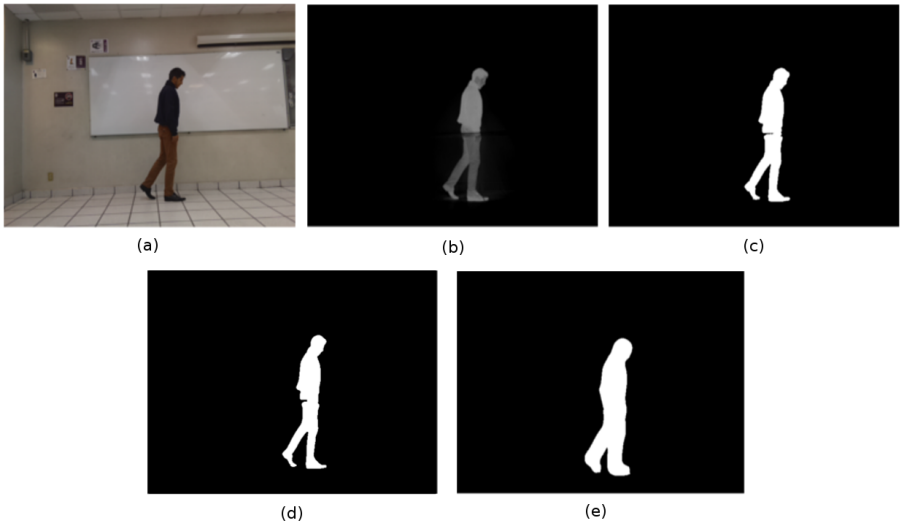

Fig. 2. Original image (a), substraction image (b), Otsu image (c), local threshold image (d), Sauvola image (e).

D. Feature extraction. This submodule obtains the descriptors of the segmented object. The invariant descriptors selected for this submodule are the Hu moments which are descriptors that extract invariant features to translations, rotations, changes of scale and illumination. These invariant moments are obtained from the geometric moments [7].

For the discrete case of an image $f(i, j)$ the standard geometric moments are obtained in (16):

$$M_{pq} = \sum_i \sum_j i^p j^q f(i, j) \quad \forall \quad p, q = 0, 1, 2, \ldots \tag{16}$$

To obtain the central moments of the object (17):

$$\mu_{pq} = \sum_i \sum_j (i - \bar{x})^p (j - \bar{y})^q f(i, j) \quad \forall\, p, q = 0, 1, \ldots \tag{17}$$

where $\bar{x}$ and $\bar{y}$ correspond to the coordinates of the center of gravity of the analysed object, which are obtained from the geometric moments of order 10, 01 and 00 (18):

$$\bar{x} = \frac{m_{10}}{m_{00}} \quad \bar{y} = \frac{m_{01}}{m_{00}} \ . \tag{18}$$

To obtain the invariants to changes of scale, each moment must be divided by a normalization factor (19) that cancels the scaling effect [7]:

$$\gamma = \frac{p + q}{2} + 1 \quad \text{for } p + q = 2, 3, \ldots \tag{19}$$

Equation (20) is used to calculate the normalized central moments:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \quad . \tag{20}$$

By obtaining the normalized central moments with (20), which allows to obtain the seven invariant moments –Hu moments– (21):

$$\begin{aligned}
\phi_1 &= \eta_{20} + \eta_{02} \\
\phi_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
\phi_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
\phi_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
\phi_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12}) \left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] + \\
&\quad (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03}) \left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] \\
\phi_6 &= (\eta_{20} - \eta_{02}) \left[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
\phi_7 &= (3\eta_{21} - \eta_{30})(\eta_{30} + \eta_{12}) \left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] + \\
&\quad (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03}) \left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] \quad .
\end{aligned} \tag{21}$$

2.3 Classification Module

Once the moments of Hu have been obtained from (21), they must be processed by the classification module to determine if the shape obtained from the image corresponds to a human form. For this, a classifier is required, in this case the multilayer perceptron was chosen, because it allows the classification of linearly non-separable patterns and it is adequate to solve biclass classification problems [7]. The *backpropagation* algorithm is used for multilayer perceptron training. The training of the network requires training patterns. This patterns correspond to a group of feature vectors obtained from segmented objects of a set of training images, which were processed with the algorithms of the previous modules. The number of elements of the training set is 182, divided into 91 elements that correspond to human form and the remaining 91 are non-human forms.

The 91 human forms were obtained from 6 volunteers, each person was photographed in different positions resembling the walking position of a person, see Fig. 3. On the other hand, the 91 images of non-human form are images of ordinary objects that are commonly found in a house, such as pets, balls, lamps.

The final architecture of the neural network, after performing several tests, is presented in Fig. 4, which shows an architecture of 7 inputs, 5 neurons were required in the hidden layer and only one neuron in the output layer. The 7 inputs correspond to the 7 elements of the feature vector obtained from Hu moments (21), these are the image descriptors. The activation function of the hidden layer neurons is the *hyperbolic tangent*, which allowed the backpropagation algorithm to train faster. For the output layer it is assigned the *sigmoid function*, because

Fig. 3. Sample of human form images obtained from the set of 91 training elements.

of the output of the network is between 0 and 1 (1 correspond to human form and 0 correspond to non-human form).

Before training, the feature vectors of the set of 182 images were normalized with (22) obtained from [9]:

$$x'_j = \frac{1 - \exp^{-\left(\frac{x_j - \mu_j}{\sigma_j}\right)}}{1 + \exp^{-\left(\frac{x_j - \mu_j}{\sigma_j}\right)}} \quad , \tag{22}$$

where x'_j corresponds to the normalized value, x_j is the value to be normalized, μ_j is the average of the corresponding dimension and σ_j is the standard deviation of the corresponding dimension. This normalization allows to obtain a set of normalized data within the range of -1 to $+1$, which is the range of values that allow the activation function *hyperbolic tangent* to obtain appropriate activation values and to avoid values that can saturate the function.

Once the values of the normalized dataset are obtained these are labeled with the value of 0 and 1 (human form: 1 and non-human form: 0). This dataset is processed by the *backpropagation* training algorithm. The training parameters were an $\alpha = 0.5$ and a $\eta = 0.25$. The algorithm converges when the error between the target and the multilayer perceptron output is as small as possible, updating in each iteration the set of weights and bias of each layer connection.

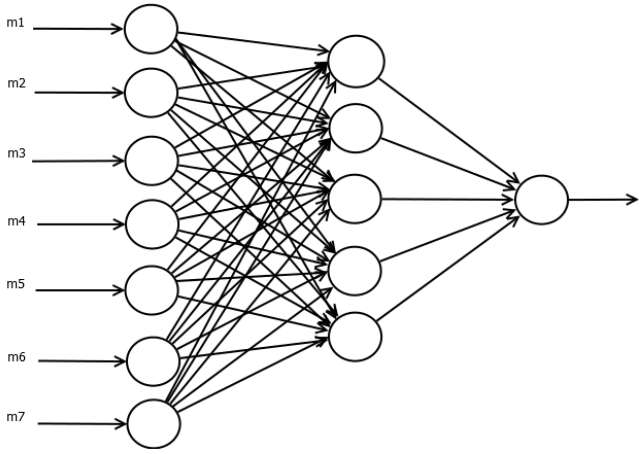

Fig. 4. Final architecture used for the classification module.

2.4 Communication Module

The communication is a simple client-server architecture, in which the image processing and classification server is implemented on the Raspberry Pi platform. The communication is implemented using a socket. The client is implemented on a computer with a Linux distribution, such as Ubuntu, which receives the status of the home monitoring performed by the server, through port 5000. The server and the client are developed under the TCP/IP protocol stack in C language.

3 Results

The performance of the classification model can be analyzed in a confusion matrix in Table 1. This matrix is a table of NxN –where N is the number of classes– that summarizes the level of success of the predictions of a classification model. In the case of the multilayer perceptron designed and implemented, the number of classes is equal to 2, because of the problem of recognizing a human form, in this case, is a binary classification problem.

The confusion matrix obtained is a matrix of $2x2$. And the results were obtained using a test set of 46 elements. The dataset of 228 images is divided in 182 training set images and 46 test set images.

Table 1. Confusion matrix of the classification model.

	Human form	**No human form**
Human form	22	1
No human form	1	22

The confusion matrix in Table 1 describes that the 23 images containing human form were correctly classified 22 and incorrectly classified 1 image. And the other 23 images that do not contain human form were correctly classified 22 and 1 incorrectly classified. Of the total of the 46 elements of the test set, 95.65% were classified by the model correctly, while 4.34% were incorrectly classified.

The execution time in which the Raspberry Pi platform performs the image processing, from the image acquisition module to the classifier module, is approximately of 21 seconds per image. This was measured considering the execution of the algorithms sequentially. Because of these results it was decided to apply HPC techniques, based in data parallelism. These techniques improved the performance, obtaining only 7 seconds of image processing, approximately.

The Raspberry Pi Camera V2 obtains an image in an amount of time of approximately 1 second, this execution time was obtained due to the reduction of the configuration time of the camera to only 500ms, which is the minimum required for the camera. As a result the total execution time is approximately of 7 to 8 seconds per image. Also the PIR sensor reading is configured to perform a sampling time of 0.5 seconds.

4 Conclusions

The development of the system on a SoC-based platform and the use of HPC techniques allowed to obtain a better computational performance. This allowed to take advantage of the available software and hardware resources of the Raspberry Pi 3 B. Also the implementation in C language, contributed to obtain a better performance.

About the image processing algorithms, the lighting changes on the test scenario generated noise in the sample images. To avoid this it was necessary to implement global and local thresholds in the image regions, in addition to the Sauvola algorithm. This algorithms of the segmentation module obtained an image without white noise or lighting changes, this result improved the robustness of the system, because of the avoiding of inaccuracies. Also obtaining the characteristic features of an object in an image through the Hu moments allowed to avoid the problem of translations, rotations, changes of scale and illumination.

The classification module required the soft-max normalization algorithm to improve the separation of the descriptor vectors of each object detected in an image. This normalization also improved the execution of the backpropagation algorithm. The result of the classifier were satisfactory. It was obtained an accuracy of approximately 96%. The result allows the system to be placed as a reliable and robust application for home automation. In comparison with the development presented by Othman and Aydin in [2], it has an accuracy of 93%, approximately. And Mathur et al. in [3] presents a development with an accuracy of only 83%. There is a significant accuracy difference and the development presented in this paper is implemented taking advantage of the hardware and software resources of the Raspberry Pi platform, through HPC techniques.

Acknowledgements. The authors thank the support of Instituto Politécnico Nacional (IPN). SIP-IPN contributed to the development of this work through 20180341 project.

References

1. Observatorio Nacional Ciudadano: Reporte sobre delitos de alto impacto. Apr. 2019. [Online]. Available: `http://onc.org.mx/wp-content/uploads/2019/06/mensual-abril.pdf`. [Accessed: Jun. 11, 2019]
2. Othman, N.A., Aydin, I.: A new IoT combined body detection of people by using computer vision for security application. In: 9th International Conference on Computational Intelligence and Communication Networks (CICN), pp. 108–112 (2017)
3. Mathur, S., Subramanian, B., Jain, S., Choudhary, K., Prabha, D.R.: Human detector and counter using Raspberry Pi microcontroller. In: 2017 Innovations in Power and Advanced Computing Technologies (i-PACT), pp. 1–7 (2017)
4. Maksimovic, M., Vujovic, V., Davidovi, N., Milosevic, V., Perisic, B.: Raspberry Pi as Internet of Things hardware: Performances and Constraints. In: IcETRAN2014, Vrnjacka Banja, Serbia (2014)
5. Villagómez, C.: El formato BMP. CCM (2017). [Online]. Available: `https://es.ccm.net/contents/719-el-formato-bmp`. [Accessed: Jan. 21, 2019]
6. Wiring Pi: GPIO Interface library for the Raspberry Pi. Wiring Pi. [Online]. Available: `http://wiringpi.com/`. [Accessed: Mar. 20, 2019]
7. Rodríguez, R., Sossa, J.H.: Procesamiento y análisis digital de imágenes. 1ra ed. México: Alfaomega Grupo Editor, pp. 180–230, 250–290, 297–350 (2012)
8. Cortés, J., Chaves, J., Mendoza, J.: Comparación cualitativa y cuantitativa de las técnicas básicas de umbralización local para el procesamiento digital de imágenes. Scientia et Technica 2(51), p236–241. [Online]. Available: `https://revistas.utp.edu.co/index.php/revistaciencia/article/view/7561/4707`. [Accessed: Dec. 11, 2018]
9. Kevin, P., Priddy, L.: Artificial Neural Networks: An Introduction. Washington: SPIE, pp. 16–17 (2005)