International Journal of Computer Technology and Science
Volume. 2, Number. 1, Year 2025
e-ISSN : 3048-1961; dan p-ISSN : 3048-1899; Page. 98-111
DOI: https://doi.org/10.62951/ijcts.v2i1.134
Available online at: https://international.arteii.or.id/index.php/IJCTS

Comparative between the binary thresholding technique and the Otsu method for the people detection

Serri Ismael Hamad
University of Thi-Qar, College of Education for Pure Sciences, Iraq

Address: Nasiriyah, Dhi Qar Governorate, Iraq
Correspondence E-mail: serriismael@utq.edu.iq

Abstract. *In image detection processes where there is a variation in brightness between pixels, techniques are required to obtainoptimal and adaptable threshold values for these variations. Therefore, a comparison between the binary thresholdingtechnique and the adaptive method of Otsu is made, in videos with dynamic and static background, weighing the response time of the algorithm, memory used, requirement of the central processing unit and hits in the detections, in the languages of Python and M (Matlab). The techniques in Python present better results in terms of response time andmemory space; while, when using Matlab, the lowest percentage of machine requirement is presented. Also, the Otsumethod improves the percentage of hits in 12.89 % and 11.3 % for videos with dynamic and static background, with respect to the binary thresholding technique.*

Keywords: *Thresholding, Detection, Performance, Accuracy*

1. INTRODUCTION

Machine vision refers to the section of techniques and processes that allow information to be inferred through previously analyzed images and videos [1]. One of the most influential applications in computer vision processes is the detection, tracking and monitoring of people in controlled and uncontrolled spaces [2][3]. In video surveillance and automatic people counting processes, high success rates are required, so it is necessary to apply techniques that improve the performance of the system with respect to the detections made and the processing time of the algorithm [4].People detection processes begin with transformations to the input image, in which the conversion to grayscale is achieved, as well as the segmentation of the image background [5], complemented with smoothing filters and morphological filters [6]. Thresholding is one of the most important stages in background subtraction processes, in which the moving object is separated from the image background [7]. There are several thresholding techniques, among which the most notable are binary thresholding and adaptive thresholding by the Otsu method. Binary thresholding is a computationally efficient process where the pixels of the image with values lower than the established threshold are determined as belonging to the image background and those that exceed it are classified as belonging to the main frame of the image [8];[9]. In both videos with a dynamic background and a static background, there are variations in the levels of luminosity and noise, causing the processes to become complex and decreasing the percentage of reliability in the detections [10]. For this reason, a thresholding

Received: Desember 06, 2024; Revised: Desember 21, 2024; Accepted: Januari 07, 2025;
Published: Januari 08, 2025;

method is required that, in the face of such variations, assumes a minimum threshold value. The Otsu method proposes a maximization of variance between classes, determining a value that generates the best separation between them, this being the value of the optimal threshold [11].In this paper, a comparison is presented in the programming languages Python 3.7.5 and M (Matlab R2018b), between the binary thresholding method and the Otsu adaptive threshold method, considering the response time of the background subtraction algorithm, the memory space used, the requirements of the central processing unit and the successes in the detections, in videos with dynamic background and static background taken in the downtown area of the city of Cúcuta, Colombia, using a video camera with a recording frequency of 30 fps and a focal length of 35. nm, located at a height of 4.5 m, treated on a computer with a Core i7 processor, a working frequency of 2.4 GHz and 4 GB of installed RAM.

2. STATE OF THE ART

In this section, research related to the binary thresholding technique is analyzed, as well as the adaptive Otsu method; in addition to works in which multiplatform comparisons are made to image treatment techniques.Huamaní in his study proposes a practical approach for the implementation of the Otsu method in Matlab, analyzing images captured in open spaces, also complementing with filtering by morphological operations of dilation and erosion, with an average error of 5 % [12]. Likewise, Ieno et al. [13] make a comparison between the threshold values obtained by simple mathematical calculations with a single image iteration, and the Otsu method, achieving differences in terms of resource consumption and maximum operating frequency. For their part, Ramos et al. propose an evaluation in Matlab of spectral similarity indices in unsupervised environments, obtaining images with variable intensities that are compared with a reference image originated after classification [14].Triana et al. conduct a study of thresholding techniques for digital image processing in Matlab, considering, among other techniques, the Otsu method and the local minimum method, in which some deformations and a longer response time are presented as image subdivisions are added to the analysis [15]. Likewise, Gómez et al. perform tests on images in which they define high and low threshold values, and compare them with the one obtained by the Otsu method, with which the problems regarding non-uniform illumination and non-constant color are reduced, since with low threshold values all the points of the image are maintained but large amounts of light, and for

high values, an initial pattern is maintained but the low intensity points are lost [16]. Huang et al. propose an algorithm based on the Otsu method, weighting the distribution characteristics of the object present in the main frame of the image and the background of the image, improving the speed of calculation of the threshold and the precision of the process [17]. In addition to the above, Cortés et al. qualitatively and quantitatively compare the basic techniques of local thresholding such as Sauvola, Otsu, Entropy, etc., weighting factors such as the response time of the algorithm and the noise present during the processing of the images [18]. In the same line, Cavanzo et al. perform a multiplatform efficiency measurement of various artificial vision algorithms, where they propose the analysis according to the execution time as a classic method of comparison between presented services [19].

3. METHODOLOGY

A methodology is proposed based on the application and comparison of two of the basic thresholding techniques in image processing: binary thresholding, and Otsu adaptive thresholding; once the input image is converted to grayscale and subsequently the background is segmented and filtered both by Gaussian blur and by morphological operations, weighing factors such as the Central Processing Unit (CPU) requirement, memory space used, response time and successes in detections [20], as illustrated in Figure 1.

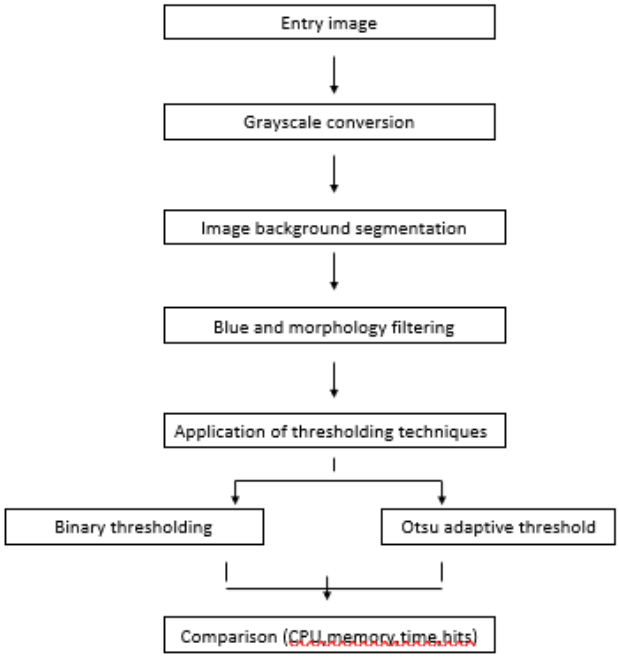


Figure 1. Proposed methodology

Transformation to input images

In this section, image processing processes such as grayscale conversion, background segmentation, Gaussian smoothing filters, and morphological operations are applied.The grayscale transformation is performed according to the recommendations of the national color television system (NCTS) [21], based on the red, blue, and green colors, as shown in (1).

$$Y = (r * 0.3) + (g * 0.59) + (b * 0.11) \quad (1)$$

For the segmentation of the image background, the peak inspection method is proposed, based on the crests of the image histogram [22]. The first crest is obtained by inspection, analyzing the first peak of the histogram. The second crest is obtained as shown in (2), where k represents the gray level, f the gray level at the highest peak, and $h(k)$ the value of the histogram at the gray level in question.

$$\text{cresta } 2 = \max[(k - f)^2 * h(k)] \quad (2)$$

Once the image is segmented, it is filtered using Gaussian smoothing or blurring by applying a weighted average to the pixels [23], as shown in (3). Where x and y refer to the distances from the origin on the horizontal and vertical axes respectively, and σ is the standard deviation of the Gaussian distribution.

$$G(x,y) = \frac{1}{2\pi\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (3)$$

Image transformation by closure morphological filtering allows to relate the two most commonly used morphological operations: dilation and erosion; in which an image A is dilated and subsequently eroded with respect to a structural element B [6], by means of logical operations, as shown in (4).

$$A \bullet B = (A \oplus B) \ominus B \quad (4)$$

Thresholding

The techniques proposed in this stage correspond to binary thresholding and the adaptive Otsu method.

In the binary thresholding or threshold definition processes, a value T is established, which allows the object to be separated from the image background as shown in (5).

$f(x, y) > T$; is part of the object

$f(x, y) < T$; is part of the background (5)

In the binary thresholding technique the threshold value set is not modified and the distinctions are dependent on said value [15]. Also, 50 is defined pixels as the threshold value according to the state of the art in this type of image processing [24]. For Otsu adaptive thresholding, a variance analysis is used to find the specific threshold value that adapts to the processing according to the luminosity levels and size of objects present in the main frame of the image [12], and which also minimizes the variance between the classes of black and white pixels [5]. This process is carried out as shown in (6).

$$\sigma_B^2(t^*) = Max_{0 \le t \le L-1}[Log_2 \sigma_B^2(t)] \qquad (6)$$

The Otsu method scans the image in search of the ideal value for thresholding, and if it fails to adapt due to the robustness of the image, it takes a previously defined value, as shown in Figure 2.

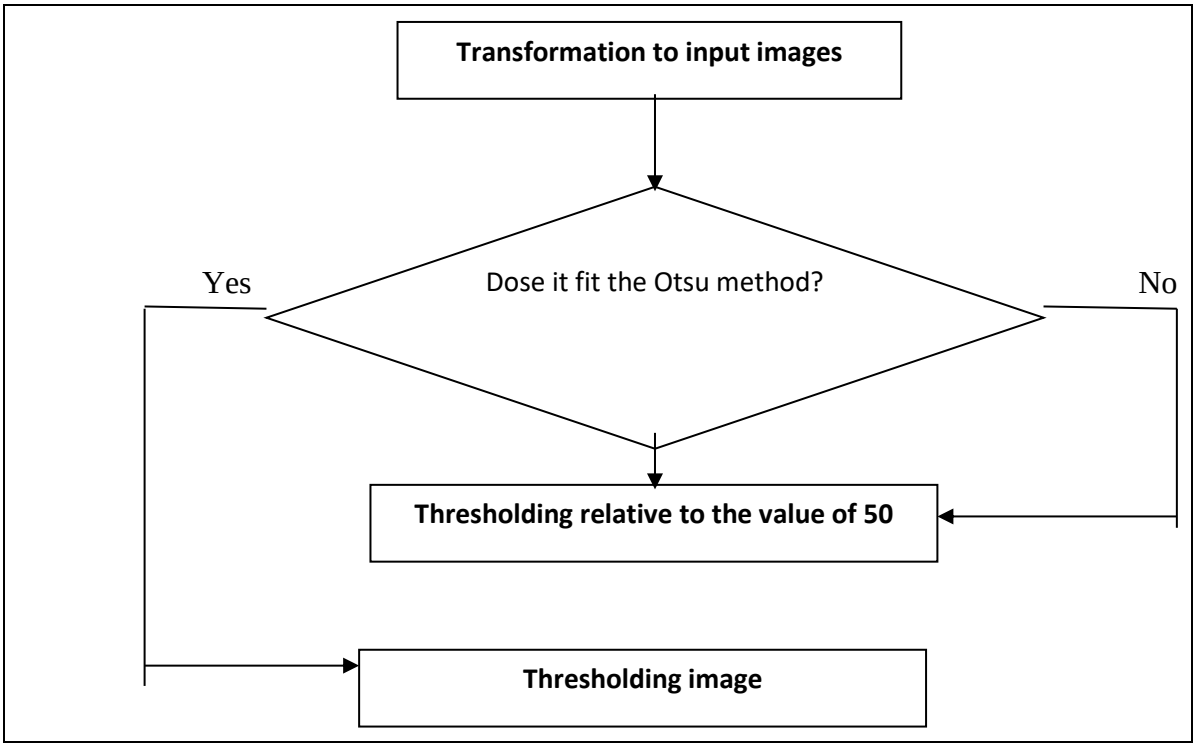


Figure 2. Proposed thresholding method

Comparison

The thresholding techniques were compared with respect to the response time of the background subtraction algorithm, the memory space used, the CPU requirements and the percentage of successful detections [25]. The parameters were obtained for both videos 1, a 20-second dynamic background video in MP4 format and 23-second static background video in

MP4 format from an average height of 4.2 meters with a device with a maximum video resolution of 2.07 megapixels and a video recording frequency of 30 frames per second, processed in both Python 3.7.5 and Matlab R2018b, using the same hardware tool for image processing, corresponding to a personal computer with a Core i7 processor, a working frequency of 2.4 GHz and 4 GB of installed RAM. The data used for the comparison of the thresholding techniques between the programs according to the memory space and CPU requirements are obtained through the Windows Task Manager in the section called "Performance". On the other hand, to measure the response time in Matlab the commands "Tic" and "Toc" are used, while in Python the "Time" library is used.

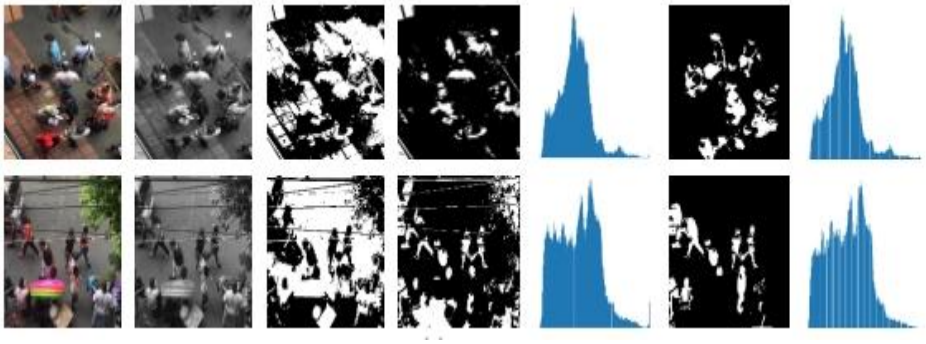
4. RESULTS

According to the proposed methodology, video image processing is first presented, followed by the application of thresholding techniques and their subsequent comparison on multiple platforms.

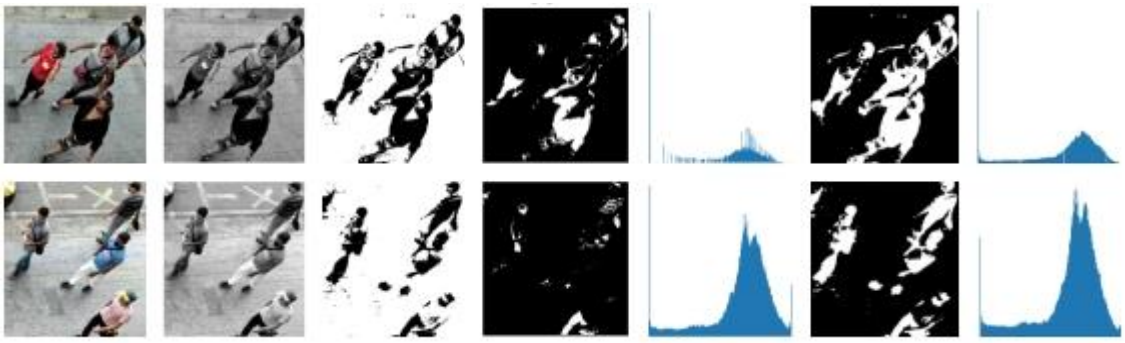
a. Image Processing and Thresholding

Figure 3 shows the processing of video images captured in a specific area. Section A shows, from left to right, the original images captured from videos with dynamic background, converted to grayscale and segmented using the peak scanning method. The image is then thresholded using a 50-pixel binary thresholding technique, along with the histogram obtained in this process. Finally, the thresholding of the image using the adaptive Otsu method and the resulting histogram are shown.

The histograms obtained by both techniques show a tendency towards dark tonality in the image; In addition, the histogram resulting from the Otsu method shows the suppression of some redundant peaks in both the white and black tones of the image, thus improving the number of hits in the detection process. Likewise, in section b of Figure 3, the processing of video images captured using videos with a static background is presented, showing from left to right the original image, the conversion to grayscale, the segmentation by peak inspection, the binary thresholding defining 50 pixels as the threshold value and the application of the Otsu adaptive method as a thresholding technique with the resulting histogram.



(a) Figure 2. Video images with dynamic background



(b) Figure 3. Video images with static background

Figure 3. Processing for thresholding video images with dynamic and static backgrounds. From left to right: original image, conversion to grayscale, segmentation, binary thresholding, binary thresholding histogram, thresholding by Otsu method, thresholding histogram by Otsu method.

The histograms obtained by both thresholding techniques show the tendency towards light tonality in the image. Likewise, the histogram resulting from the thresholding using the Otsu method shows the enhancement of sections of the image, thus allowing the distinction of the contours of the image, which facilitates the differentiation of the detected objects that are candidates for people.

Weighting of factors

Figure 4 shows the graph of the response time of the background subtraction algorithm with the embedded thresholding techniques. For the binary thresholding technique, average response times of 0.676 and 0.827 seconds are obtained in Python and Matlab respectively. Similarly, for the thresholding technique using the Otsu method, average times of 0.714 and 0.901 seconds are obtained when using Python and Matlab respectively. Likewise, Figure 5

shows the memory space required by the algorithm with the thresholding techniques in videos with a dynamic background.

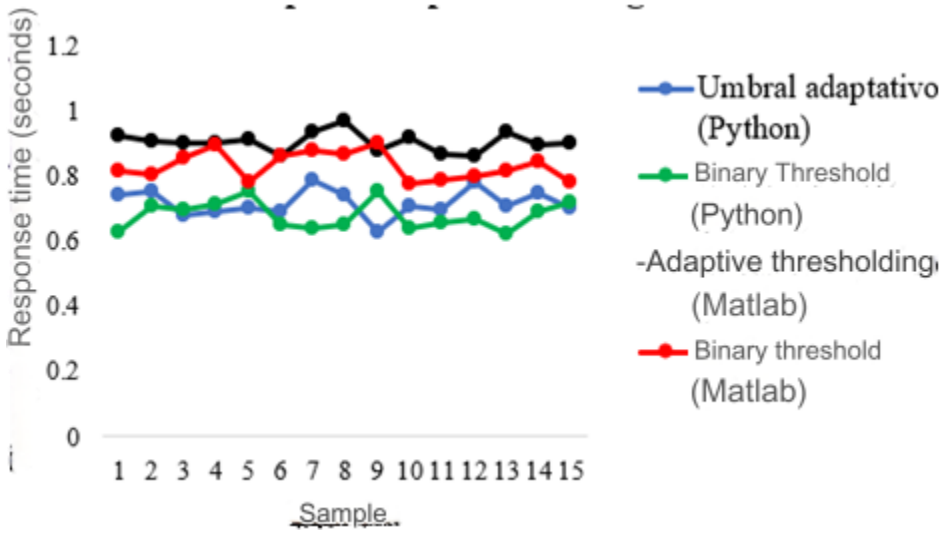


Figure. 4. Response time when thresholding a video with a dynamic background

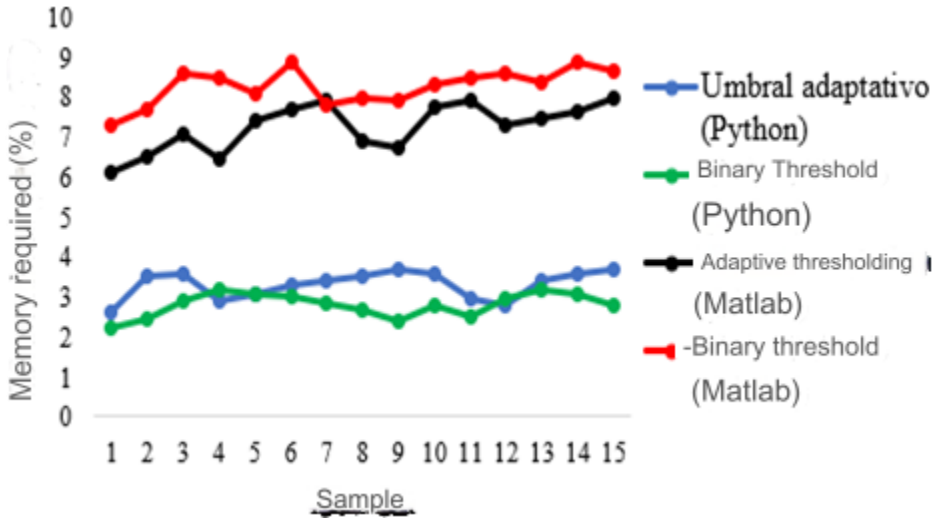


Figure 5. Memory required when thresholding a video with a dynamic background

The binary thresholding technique in Python requires on average 2.807% of the available memory space on the device, while using Matlab it requires on average 8.276% of the memory. On the other hand, the adaptive thresholding technique using the Otsu method requires on average 3.308% and 7.255% of the available memory space, in Python and Matlab respectively.

Likewise, Figure 6 shows the CPU requirement during thresholding of video images with dynamic background.

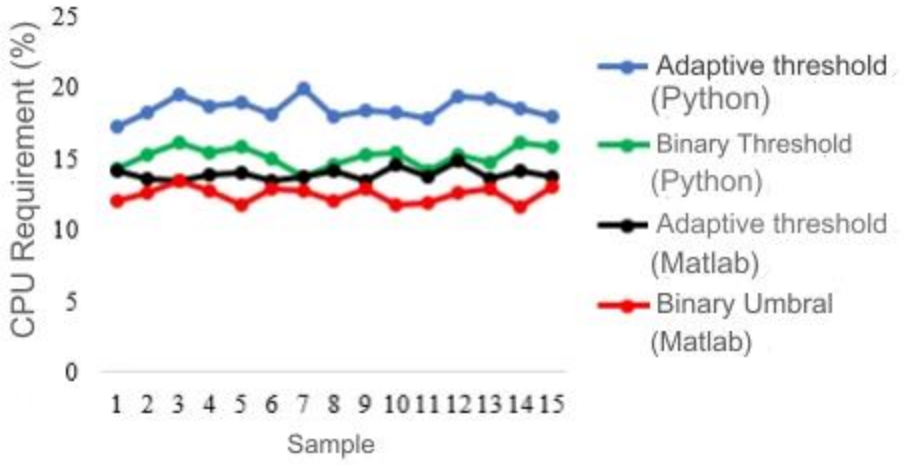


Figure 6. CPU requirement when thresholding a video with a dynamic background

15.15% and 12.48% of the CPU are required during the implementation of the binary thresholding technique in Python and Matlab respectively. Likewise, when thresholding using the adaptive Otsu method in Python, 18.54% of the CPU is required, while in Matlab, 13.906% of the CPU is required. Figure 7 shows the behavior regarding the response time of the background subtraction algorithm, in video images with a static background.

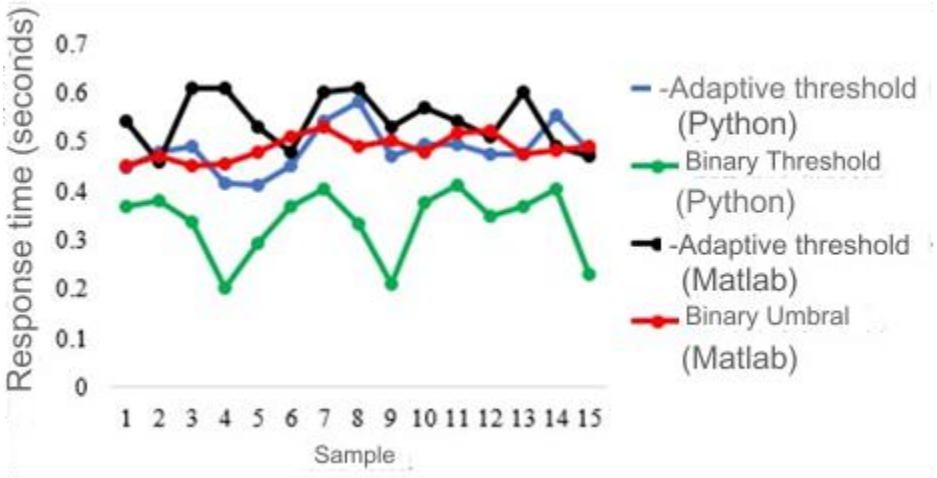


Figure 7. Response time when thresholding a video with a static background

The binary thresholding technique presents average response times of 0.335 and 0.486 seconds in Python and Matlab respectively. On the other hand, the binary thresholding technique Adaptive thresholding using the Otsu method presents average response times of 0.484 seconds in Python and 0.5433 seconds in Matlab.

Similarly, Figure 8 shows the memory space used by the algorithm with thresholding techniques for videos with a static background.

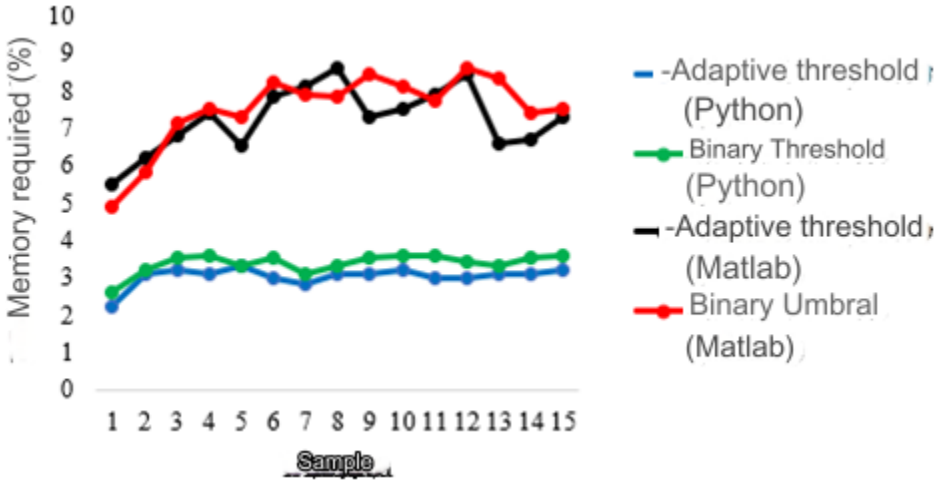


Figure 8. Memory required when thresholding a video with a static background

3.37% of the available memory is required when applying the binary thresholding technique in Python, while in Matlab, this value is 7.5%. Similarly, 3.03% and 7.3% of the available memory are required when applying the Otsu adaptive thresholding technique, in Python and Matlab respectively.

Likewise, Figure 9 illustrates the CPU requirement during the processing of the video image with a static background.

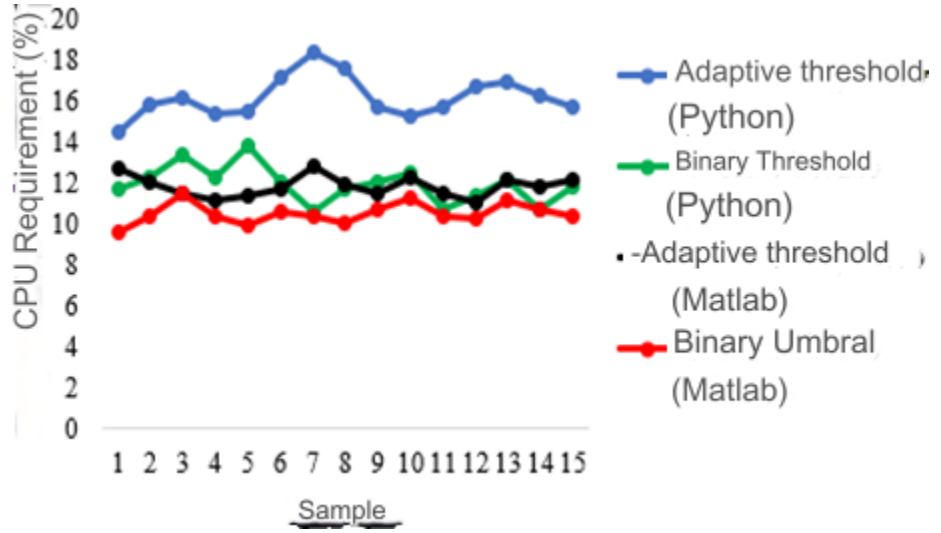


Figure 9. CPU requirement when thresholding a video with a static background

The binary thresholding technique requires 11.8% of the central processing unit when applied in Python, while in Matlab, this value is 10.3%. Likewise, the adaptive thresholding technique using the Otsu method requires 15.7% of the central processing unit when applied in Python, and 12.1% when applied in Matlab.

Comparing binary thresholding techniques and adaptive thresholding using the Otsu method in Python and M (Matlab) languages according to response time, memory space used and central processing unit requirements is necessary when defining the tools to be used in an image processing system; since, although similar trends are obtained in the two environments tested for videos with dynamic and static backgrounds, there are also variations in some of the peaks of the values obtained, which could potentially be relevant when working with hardware devices that have memory and processing limitations.Table 1 shows the comparison between the binary thresholding technique and the Otsu adaptive method, according to the number of false positives (FP), false negatives (FN) and the percentage of error with respect to the real number of people (NP) that appear in the video image, and the number of detections (ND) when applying the algorithm for background subtraction.

Table 1. Comparison of thresholding techniques

	Video with dynamic background				
	NP	**ND**	**FP**	**FN**	**Error**
Binary Thresholding	69	57	3	15	22.17 %
Otsu method	69	70	5	4	9.28 %
	Video with static background				
	NP	**ND**	**FP**	**FN**	**Error**
Binary Thresholding	62	55	5	12	20.97 %
Otsu method	62	68	6	0	9.67 %

The treatment of the images of the video with a dynamic background using the binary thresholding technique yields a success rate of 77.83%, while when applying the Otsu method, this value rises to 90.72%. Likewise, for the video with a static background, success rates of 79.03% and 90.33% are obtained when applying the binary thresholding technique and the Otsu method, respectively.

5. CONCLUSIONS

In videos with dynamic backgrounds, an average response time of 0.751 seconds was obtained when applying the binary thresholding technique, and 0.807 seconds with the Otsu method. Similarly, the binary thresholding technique required an average of 5.541% of the memory space, while the Otsu method required 5.281% of it. From the central processing unit,

when applying the binary thresholding technique, an average of 13.81% of it was required, while when applying the Otsu method, this value increased to 16.22%. The data mentioned refer to the average of the means in the tests performed. In videos with a static background, the average response time when applying the binary thresholding technique was 0.411 seconds, while when applying the Otsu method, this value rose to 0.5135 seconds. Likewise, the memory space required when applying the binary thresholding technique was 5.435%,while when using the Otsu method, this value decreased to 5.165%. The average requirement percentage of the central processing unit was 11.05% when applying the binary thresholding technique, and 13.9% when using the Otsu method. The data mentioned refers to the average of the means in the tests performed. Python, being a programming language based on free software, has an advantage over Matlab, which is licensed, by reducing implementation costs. In addition, Python code tends to be less detailed, so it is roughly reduced to half its length. In contrast, for image processing, Matlab presents high-level tools for the scientific community with extensive mathematical support for understanding the processing stages, while the information available in Python is mostly focused exclusively on development from the coding point of view. On average, the success rate improved by 12.095% when applying the Otsu method compared to the binary thresholding technique, because errors due to light variation and noise at the time of image capture are eliminated, as well as the dependence of the shades in the clothing used by the people present in the public space areas analyzed, which makes the use of the Otsu method relevant in video surveillance processes, tracking and tracing of people in open spaces, using hardware devices that do not have limitations in processing and available memory. This document presents information to be taken into account when implementing people detection processes using computer vision techniques by comparing response time, memory used, requirements of central processing unit and success rates in detections, since the performance and efficiency of most processes depends largely on the performance of the hardware tool used. Likewise, binary thresholding and adaptive thresholding techniques using the Otsu method correspond to techniques with great potential in image processing, and generate high success rates that allow these techniques to be replicated in disease detection processes and feature recovery in medical images, as well as in identification and classification processes according to crop quality, among others. As future work, it is proposed to accompany detection processes

using thresholding techniques with adaptive stages in the search for contours, generating highlights and predictions in them, thus reducing the number of false negatives.

REFERENCES

Aleskerov, F. T., & Chistyakov, V. V. (n.d.). The threshold decision making. *Procedia Computer Science*.

Cavanzo, G., Perez, M., & Villavisan, F. (n.d.). Efficiency measurement of computer vision algorithms implemented on Raspberry Pi and personal computer using Python.

Cortes Osorio, J., Chaves Osorio, J., & Mendoza Vargas, J. (n.d.). Qualitative and quantitative comparison of basic local thresholding techniques for digital image processing.

Gedraite, E. S., & Hadad, M. (n.d.). Investigation on the effect of a Gaussian blur in image filtering and segmentation.

Gomez-Villa, A., Diez-Valencia, G., & Salazar-Jimenez, A. E. (n.d.). A Markov random field image segmentation model for lizard spots.

Guerrero Balaguera, J. D. (n.d.). Image processing algorithms.

Huamani Navarrete, P. (n.d.). Multiple thresholding using the Otsu method for red light recognition in traffic lights. *Scientia*.

Huang, M., Yu, W., & Zhu, D. (n.d.). An improved image segmentation algorithm based on the Otsu method. In *Proceedings of the 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking*.

Ieno, E., Garces, L. M., Cabrera, A. J., & Pimenta, T. C. (n.d.). Simple generation of threshold for image binarization on FPGA.

Kajabad, E. N., & Ivanov, S. V. (n.d.). People detection and finding attractive areas by the use of movement detection analysis and deep learning approach. *Procedia Computer Science*.

Leo, M., Medioni, G., Trivedi, M., Kanade, T., & Farinella, G. M. (n.d.). Computer vision for assistive technologies. *Computer Vision and Image Understanding*.

Lopez-Portilla Vigil, B., Menendez Alonso, R., & Iglesias Martinez, M. (n.d.). Implementation of the Otsu algorithm on FPGA. *Revista Cubana de Computer Sciences*.

Maldonado Beltran, J., Pena Cortes, C., & Gualdron Gonzalez, O. (n.d.). Automatic identification of gas storage cylinders using Hopfield neural networks.

Maya Perfetti, N. I., Nunez Bedoya, A. M., & Romo Romero, H. A. (n.d.). Performance analysis of vehicle number plate recognition algorithms developed using discrete wavelet transformation and digital image correlation. *Investigación e Innovación en Ingeniería*.

Neumann, L., & Vedaldi, A. (n.d.). Tiny people pose. In *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*.

Nino Rondon, C. V., Castro Casadiego, S. A., & Medina Delgado, B. (n.d.). Characterization for the location in video capture applied to artificial vision techniques in the detection of people. *Revista Colombiana de Tecnología Avanzada*.

Nino Rondon, C. V., Castro Casadiego, S. A., Medina Delgado, B., & Guevara Ibarra, D. (n.d.). Feasibility analysis and design of an electronic system for monitoring population dynamics.

Paul, M., Haque, S. M. E., & Chakraborty, S. (n.d.). Human detection in surveillance videos and its applications - A review. *EURASIP Journal on Advances in Signal Processing*.

Raghavachari, C., Aparna, V., Chithira, S., & Balasubramanian, V. (n.d.). A comparative study of vision-based human detection techniques in people counting applications. *Procedia Computer Science*.

Ramos, J. F., Renza, D., & Ballesteros, D. M. (n.d.). Evaluation of spectral similarity indices in unsupervised change detection approaches.

Sanchez-Torres, G., & Taborda-Giraldo, J. A. (n.d.). Automatic estimation of beach occupancy by digital image processing.

Shoba, S., & Rajavel, R. (n.d.). Image processing techniques for segments grouping in monaural speech separation. *Circuits, Systems*.

Siqueira, D. L., & Correa Machado, A. M. (n.d.). People detection and tracking in low frame-rate dynamic scenes. *IEEE Latin America Transactions*.

Triana, N., Jaramillo, A. E., Gutierrez, R. M., & Rodriguez, C. A. (n.d.). Thresholding techniques for digital image processing of GEM-Foils.

Vargas, G., Neira, O., Arango, R., & Fernando, D. (n.d.). Cloud segmentation methods applied to satellite images.