Vol. 20, n.° 2, pp. 65-74, 2021
Revista UIS Ingenierías
Página de la revista: revistas.uis.edu.co/index.php/revistauisingenierias

Comparativa entre la técnica de umbralización binaria y el método de Otsu para la detección de personas

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

Carlos Vicente Niño-Rondón [1a], Sergio Alexander Castro-Casadiego [1b], Byron Medina-Delgado [1c], Dinael Guevara-Ibarra[1d], Luis Leonardo Camargo-Ariza [2]

[1] Grupo de investigación y Desarrollo en Electrónica y Telecomunicaciones, Facultad de Ingeniería, Universidad Francisco de Paula Santander, Colombia. Orcid: [a] 0000-0002-3781-4564, [b] 0000-0003-0962-9916, [c] 0000-0003-0754-8629, [d] 0000-0003-3007-8354.
Correos electrónicos: [a] carlosvicentenr@ufps.edu.co, [b] sergio.castroc@ufps.edu.co, [c] byronmedina@ufps.edu.co, [d] dinaelgi@ufps.edu.co
[2] Grupo de Investigación en Desarrollo Electrónico y Aplicaciones Móviles, Facultad de Ingeniería, Universidad del Magdalena, Colombia. Orcid: 0000-0002-7956-441X. Correo electrónico: lcamargoa@unimagdalena.edu.co

Recibido: 16 julio, 2020. Aceptado: 31 agosto, 2020. Versión final: 2 enero, 2021.

Resumen

En procesos de detección por imágenes en las que existe variación de luminosidad entre pixeles, se requieren técnicas que permitan obtener valores óptimos y adaptables de umbral ante dichas variaciones. Por ello, se realiza una comparativa entre la técnica de umbralización binaria y el método adaptativo de Otsu, en videos con fondo dinámico y estático, ponderando el tiempo de respuesta del algoritmo, memoria utilizada, requerimiento de la unidad central de procesos y aciertos en las detecciones, en los lenguajes de Python y M (Matlab). Las técnicas en Python presentan mejores resultados en cuanto a tiempo de respuesta y espacio de memoria; mientras que, al utilizar Matlab, se presenta el menor porcentaje de requerimiento de máquina. Asimismo, el método de Otsu mejora el porcentaje de aciertos en 12.89 % y 11.3 % para videos con fondo dinámico y estático, respecto a la técnica de umbralización binaria.

Palabras clave: detección de personas; comparativa; umbralización binaria; método de Otsu; Python; Matlab; tiempo; memoria; requerimiento de máquina; aciertos.

Abstract

In image detection processes where there is a variation in brightness between pixels, techniques are required to obtain optimal and adaptable threshold values for these variations. Therefore, a comparison between the binary thresholding technique 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 and memory space; while, when using Matlab, the lowest percentage of machine requirement is presented. Also, the Otsu method 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: people detection; comparative; binary thresholding; Otsu method; Python; Matlab; time; memory; machine requirement; hits.

ISSN impreso: 1657 - 4583. ISSN en línea: 2145 - 8456, **CC BY-ND 4.0**
Como citar: C. V. Niño-Rondón, S. A. Castro-Casadiego, B. Medina-Delgado, D. Guevara-Ibarra, L. L.Camargo-Ariza, "Comparativa entre la técnica de umbralización binaria y el método de Otsu para la detección de personas," *Rev. UIS Ing.*, vol. 20, no. 2, pp. 65-74, 2021, doi: 10.18273/revuin.v20n2-2021006

1. Introducción

La visión artificial refiere al apartado de técnicas y procesos que permiten inferir información a través de imágenes y videos previamente analizados [1]. Una de las aplicaciones de mayor repercusión en los procesos de visión por computadora es la de detección, rastreo y seguimiento de personas en espacios controlados y no controlados [2][3]. En procesos de video vigilancia y conteo automático de personas, se requiere de tasas de acierto altas, por lo que resulta necesario aplicar técnicas que mejoren el rendimiento del sistema respecto a las detecciones realizadas y al tiempo de procesamiento del algoritmo [4].

Los procesos de detección de personas inician con transformaciones a la imagen de entrada, en las que se logra la conversión a escala de grises, así como la segmentación del fondo de la imagen [5], complementado con filtros de suavizado y filtros morfológicos [6]. La umbralización es una de las etapas más importantes en procesos de sustracción de fondo, en los que se separa al objeto en movimiento del fondo de la imagen [7]. Existen diversas técnicas de umbralización, entre las que destacan la umbralización binaria y el umbral adaptativo por el método de Otsu. La umbralización binaria es un proceso computacionalmente eficiente en donde los pixeles de la imagen con valores menores al umbral establecido se determinan como pertenecientes al fondo de la imagen y los que lo superen, son clasificados como pertenecientes al cuadro principal de la imagen [8];[9]. Tanto en videos con fondo dinámico y fondo estático, se presentan variaciones en los niveles de luminosidad y ruido, generando que los procesos se tornen complejos y disminuya el porcentaje de fiabilidad en las detecciones [10]. Es por esto, que se requiere de un método de umbralización que ante dichas variaciones, asuma un valor mínimo de umbral. El método de Otsu propone una maximización de varianza entre clases, determinando un valor que genere la mejor separación entre las mismas, siendo este el valor del umbral óptimo [11].

En este documento, se presenta una comparación en los lenguajes de programación Python 3.7.5 y M (Matlab R2018b), entre el método de umbralización binaria y el método de umbral adaptativo de Otsu, ponderando el tiempo de respuesta del algoritmo de sustracción de fondo, el espacio de memoria utilizado, los requerimientos de la unidad central de procesos y los aciertos en las detecciones, en videos con fondo dinámico y fondo estático tomados en la zona céntrica de la ciudad de Cúcuta, Colombia, mediante una cámara de video con frecuencia de grabación de 30 fps y distancia focal de 35 nm, ubicada a una altura de 4.5 m, tratados en una computadora con procesador Core i7, frecuencia de trabajo de 2.4 GHz y 4 GB de memoria RAM instalada.

2. Estado del arte

En esta sección, se analizan investigaciones relacionadas con la técnica de umbralización binaria, así como el método adaptativo de Otsu; además de trabajos en los que se realicen comparaciones en multiplataforma a técnicas de tratamiento de imagen.

Huamaní en su estudio propone un enfoque práctico para la implementación del método de Otsu en Matlab, analizando imágenes capturadas en espacios abiertos, complementando además con filtrado por operaciones morfológicas de dilatación y erosión, con un error medio de 5 % [12]. Así mismo, Ieno et al. [13] realizan una comparativa entre los valores de umbral obtenidos por cálculos matemáticos simples con única iteración de imagen, y el método de Otsu, logrando diferencias en términos de consumo de recursos y frecuencia máxima de operación. Por su parte, Ramos et al. proponen una evaluación en Matlab de índices de similitud espectral en ambientes no supervisados, obteniendo imágenes con intensidades variables que se comparan con una imagen de referencia originada posterior a la clasificación [14].

Triana et al. realizan un estudio a las técnicas de umbralización para el procesamiento digital de imágenes en Matlab, ponderando entre otras técnicas, al método de Otsu y al método del mínimo local, en los que se presentan algunas deformaciones y un tiempo de respuesta mayor a medida que se añadan al análisis subdivisiones de imágenes [15]. Así mismo, Gómez et al. realizan pruebas a imágenes en las que definen valores de umbral altos y bajos, y los comparan respecto al obtenido por el método de Otsu, con el cual disminuyen los problemas en cuanto a iluminación no uniforme y color no constante, ya que con valores bajos de umbral se mantienen todos los puntos de la imagen pero grandes cantidades de luz, y para valores altos, se mantiene un patrón inicial pero se pierden los puntos de baja intensidad [16]. Huang et al. proponen un algoritmo basado en el método de Otsu ponderando las caracteristicas de distribución del objeto presente en el cuadro principal de la imagen y el fondo de la misma, mejorando la velocidad de cálculo del umbral y la precisión del proceso [17]. En adición a lo anterior, Cortés et al. comparan cualitativa y cuantitativamente las técnicas básicas de umbralización local como Sauvola, Otsu, Entropía, etc., ponderando factores como el tiempo de respuesta del algoritmo y el ruido presente durante el procesamiento de las imágenes [18]. En esa misma línea,

Cavanzo et al. realizan una medición de eficiencia en multiplataforma a diversos algoritmos de visión artificial, donde proponen el análisis según el tiempo de ejecución como método clásico de comparación entre servicios presentados [19].

3. Metodología

Se propone una metodología basada en la aplicación y comparación de dos de las técnicas de umbralización básicas en procesamiento de imágenes: umbralización binaria, y umbral adaptativo de Otsu; una vez la imagen de entrada sea convertida a escala de grises y posteriormente se segmente el fondo de la misma y se filtre tanto por desenfoque gaussiano, como por operaciones morfológicas, ponderando factores como el requerimiento de Unidad Central de Procesos (CPU), espacio de memoria utilizado, tiempo de respuesta y aciertos en las detecciones [20], como se ilustra en la Figura 1.

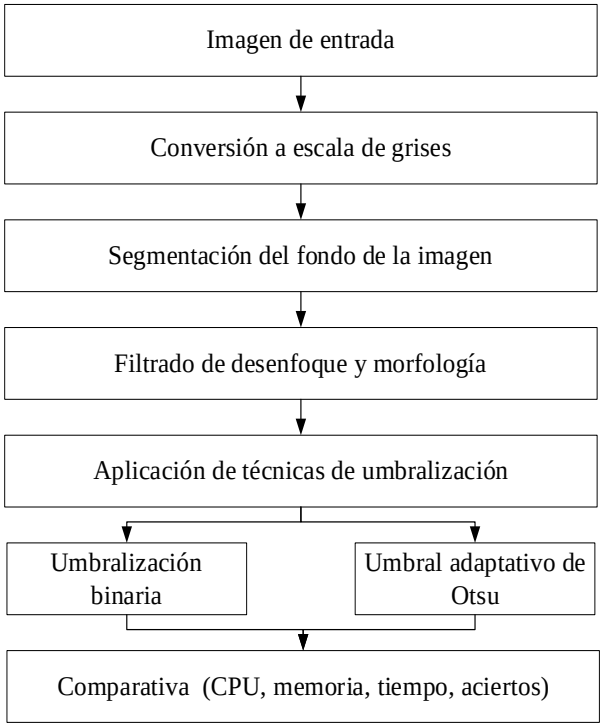

Figura 1. Metodología propuesta.
Fuente: elaboración propia.

3.1. Transformación a las imágenes de entrada

En esta sección, se aplican procesos de tratamiento de imagen, como conversión a escala de grises, segmentación del fondo, filtrados de suavizado gaussiano, y por operaciones morfológicas.

La transformación a escala de grises se realiza según las recomendaciones del sistema de televisión en color (*NTSC*) [21], basándose en los colores rojo, azul y verde, tal y como se muestra en (1).

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

Para la segmentación del fondo de la imagen, se propone el método de inspección de picos, basado en las crestas del histograma de la imagen [22]. La primera cresta se obtiene por inspección, analizando el primer pico del histograma. Por su parte, la segunda cresta se obtiene de la forma mostrada en (2), donde k simboliza el nivel de gris, f el nivel de gris en el pico más alto y $h(k)$ el valor del histograma en el nivel de gris en cuestión.

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

Una vez segmentada la imagen, se filtra mediante suavizado o desenfoque gaussiano aplicando una media ponderada a los pixeles [23], como se muestra en (3). Donde x e y, hacen referencia a las distancias desde el origen en los ejes horizontal y vertical respectivamente, y σ, es la desviación estándar de la distribución gaussiana.

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

La transformación de la imagen por filtrado morfológico de cierre permite relacionar las dos operaciones morfológicas más utilizadas: dilatación y erosión; en la que una imagen A se dilata y posteriormente se erosiona respecto a un elemento estructural B [6], mediante operaciones lógicas, como se muestra en (4).

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

3.2. Umbralización

Las técnicas propuestas en esta etapa corresponden a la umbralización binaria y al método adaptativo de Otsu.

En los procesos de definición de umbral o umbralización binaria, se establece un valor T, que permita separar el objeto del fondo de la imagen como se muestra en (5).

$$\begin{matrix} f(x,y) > T; hace\ parte\ del\ objeto \\ f(x,y) < T; hace\ parte\ del\ fondo \end{matrix} \quad (5)$$

En la técnica de umbralización binaria el valor de umbral establecido no se modifica y las distinciones son dependientes de dicho valor [15]. Asimismo, se define 50

pixeles como el valor de umbral según el estado del arte en este tipo de procesamiento de imágenes [24].

Para la umbralización adaptativa de Otsu, mediante un análisis de varianzas se encuentra el valor especifico de umbral que se adapta al procesamiento según los niveles de luminosidad y tamaño de objetos presentes en el cuadro principal de la imagen [12], y que, además, minimiza la varianza entre las clases de los pixeles blancos y negros [5]. Dicho proceso se realiza de la forma mostrada en (6).

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

El método de Otsu realiza un barrido por la imagen en búsqueda del valor ideal para la umbralización, y en caso de no lograr adaptarse por la robustez de la imagen, toma un valor definido previamente, como se muestra en la Figura 2.

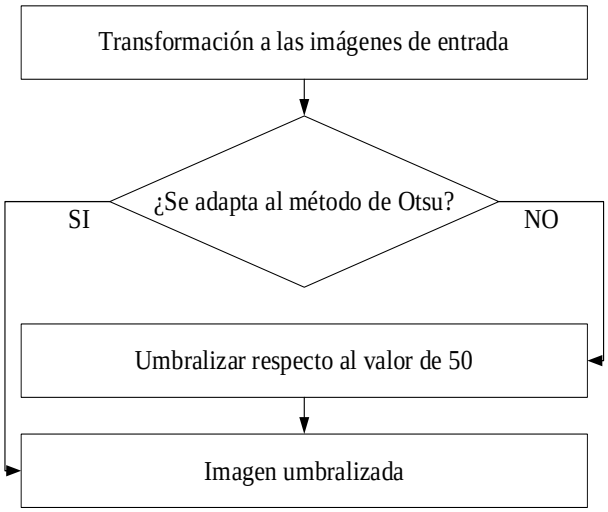

Figura 2. Método de umbralización propuesto.
Fuente: elaboración propia

3.3. Comparativa

Las técnicas de umbralización se comparan respecto al tiempo de respuesta del algoritmo de sustracción de fondo, espacio de memoria utilizado, requerimiento de unidad central de procesos (*CPU)* y porcentaje de acierto en las detecciones [25]. Los parámetros son obtenidos tanto para video 1 video con fondo dinámico en formato MP4 con duración de 20 segundos capturado en la Calle 9ª con Avenidas 6ª y 7ª de la ciudad de Cúcuta, y un video con fondo estático en formato MP4 con duración de 23 segundos capturado en la Avenida 6ª con Calles 6ª y 7ª de la ciudad de Cúcuta. desde una altura promedio de 4.2 metros con un dispositivo de resolución máxima de video de 2.07 MP y frecuencia de grabación de video de 30 fps, procesadas tanto en Python 3.7.5 como en Matlab R2018b, utilizando la misma herramienta de hardware para el tratamiento de las imágenes, correspondiente a una computadora personal con procesador Core i7, frecuencia de trabajo de 2.4GHz y memoria RAM instalada de 4 GB. Los datos empleados para la comparativa entre softwares de las técnicas de umbralización según el espacio de memoria y el requerimiento de la unidad central de procesos, se obtienen mediante el administrador de tareas de Windows en la sección denominada “rendimiento”. Por su parte, para la medición del tiempo de respuesta en Matlab se utilizan los comandos “Tic” y “Toc”, mientras que en Python se utiliza la librería “Time”.

4. Resultados

Siguiendo la metodología propuesta, se presenta en primer lugar el tratamiento a las imágenes de video, seguido de la aplicación de las técnicas de umbralización y su posterior comparativa en multiplataforma.

4.1. Procesamiento de imagen y umbralización

En la Figura 3, se presenta el procesamiento a las imágenes de video capturadas en la zona céntrica de la ciudad de Cúcuta. En la sección a, de izquierda a derecha se muestran imágenes originales capturadas de videos con fondo dinámico, la conversión a escala de grises y la segmentación de la misma por el método de inspección de picos. Seguidamente, se muestra la umbralización de la imagen por la técnica de umbralización binaria con un valor de 50 píxeles, y el histograma obtenido en dicho proceso. Finalmente, se presenta la umbralización de la imagen por el método adaptativo de Otsu y el histograma resultante por el mismo.

Los histogramas obtenidos por ambas técnicas muestran una tendencia de tonalidad oscura de la imagen; Además, en el histograma resultante del método de Otsu se muestra la supresión de algunos picos redundantes tanto en las tonalidades blancas como en las tonalidades negras de la imagen, mejorando así el número de aciertos en el proceso de detección. Igualmente, en la sección b de la Figura 3, se presenta el procesamiento a las imágenes de video capturadas mediante videos con fondo estático, mostrando de izquierda a derecha la imagen original, la conversión a escala de grises, la segmentación por inspección de picos, la umbralización binaria definiendo 50 píxeles como el valor del umbral y la aplicación del método adaptativo de Otsu como técnica de umbralización con el histograma resultante del mismo.

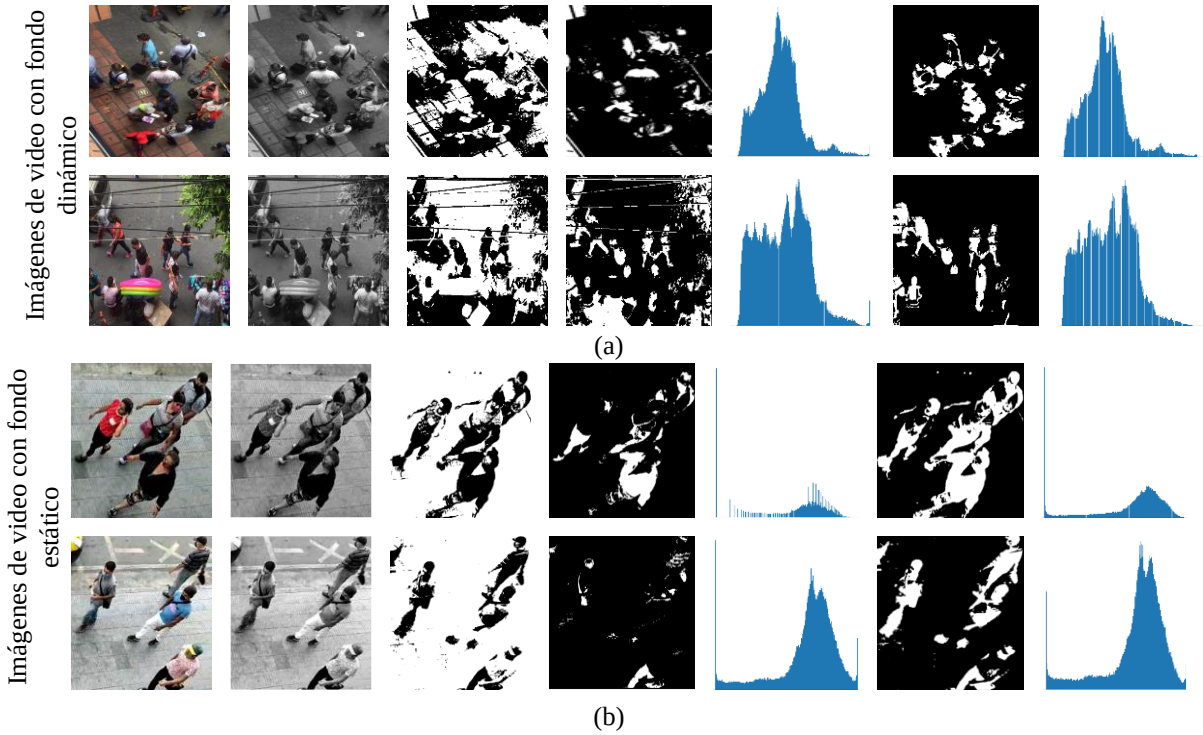

Figura 3. Procesamiento para umbralización a imágenes de video con fondo dinámico y estático. De izquierda a derecha: imagen original, conversión a grises, segmentación, umbralización binaria, histograma de umbralización binaria, umbralización por método de Otsu, histograma de umbralización por método de Otsu.
Fuente: elaboración propia.

Los histogramas obtenidos por ambas técnicas de umbralización muestran la tendencia a la tonalidad clara en la imagen. Asimismo, el histograma resultante de la umbralización por método de Otsu, muestra el realce de secciones de la imagen permitiendo así la distinción de los contornos de la misma, lo que facilita la diferenciación de los objetos detectados candidatos a personas.

4.2. Ponderación de factores

En la Figura 4, se muestra la gráfica del tiempo de respuesta del algoritmo de sustracción de fondo con las técnicas de umbralización inmersas.

Para la técnica de umbralización binaria se obtienen tiempos de respuesta promedio de 0.676 y 0.827 segundos en Python y Matlab respectivamente. Del mismo modo, para la técnica de umbralización por método de Otsu se obtienen tiempos promedio de 0.714 y 0.901 segundos al utilizar Python y Matlab respectivamente.

Igualmente, en la Figura 5 se presenta el espacio de memoria requerido por el algoritmo con las técnicas de umbralización en videos con fondo dinámico.

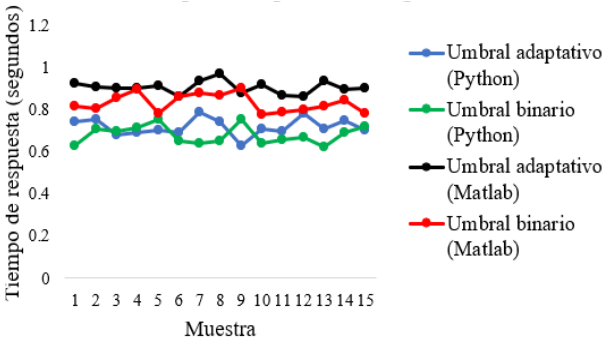

Figura. 4. Tiempo de respuesta al umbralizar un video con fondo dinámico. Fuente: elaboración propia.

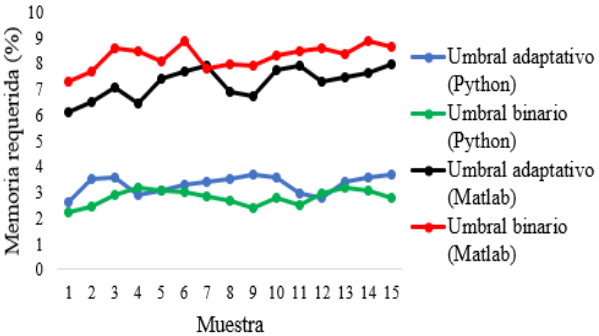

Figura. 5. Memoria requerida al umbralizar un video con fondo dinámico. Fuente: elaboración propia.

La técnica de umbralización binaria en Python requiere en promedio del 2.807 % del espacio de memoria disponible en el dispositivo, mientras que al utilizar Matlab se requiere en promedio del 8.276 % de la memoria. Por su parte, la técnica de umbral adaptativo por el método de Otsu requiere en promedio del 3.308 % y 7.255 % del espacio de memoria disponible, en Python y Matlab respectivamente.

Así mismo, en la Figura 6 se muestra el requerimiento de CPU durante la umbralización de las imagenes de vídeo con fondo dinámico.

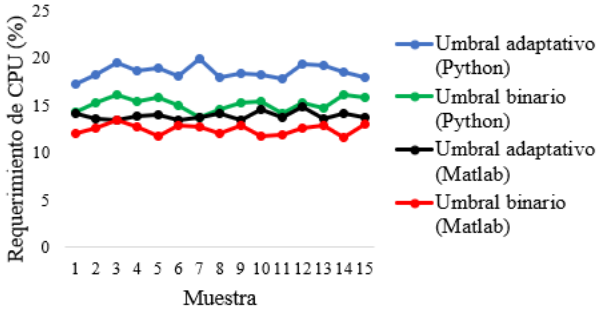

Figura 6. Requerimiento de CPU al umbralizar un video con fondo dinámico. Fuente: elaboración propia.

Se requiere del 15.15 % y 12.48 % de la unidad central de procesos durante la implementación de la técnica de umbralización binaria en Python y Matlab respectivamente. Así mismo, al umbralizar por el método adaptativo de Otsu en Python se requiere del 18.54 % de la unidad central de procesos, mientras que en Matlab se requiere del 13.906 % de la misma.

En la Figura 7, se presenta el comportamiento respecto al tiempo de respuesta del algoritmo de sustracción de fondo, en imágenes de video con fondo estático.

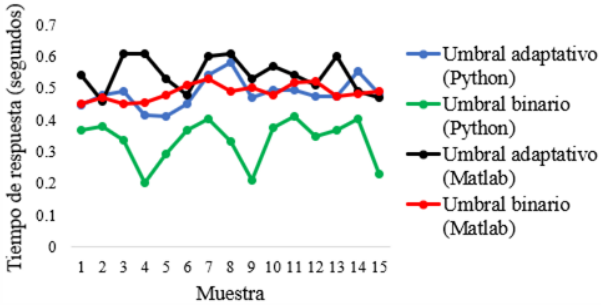

Figura 7. Tiempo de respuesta al umbralizar un video con fondo estático. Fuente: elaboración propia.

La técnica de umbralización binaria presenta tiempos de respuesta promedio de 0.335 y 0.486 segundos en Python y Matlab respectivamente. Por su parte, la técnica de umbralización adaptativa por el método de Otsu presenta tiempos de respuesta promedio de 0.484 segundos en Python y 0.5433 segundos en Matlab.

De igual forma, en la Figura 8 se presenta el espacio de memoria utilizado por el algoritmo con las técnicas de umbralización para videos con fondo estático.

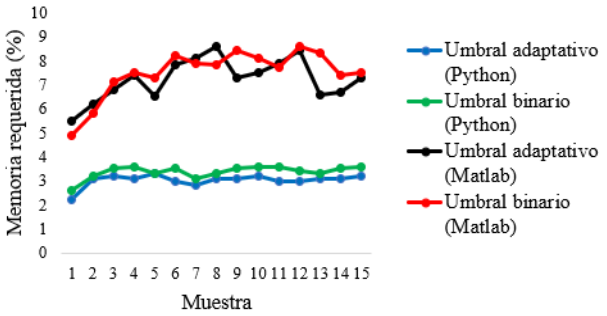

Figura 8. Memoria requerida al umbralizar un video con fondo estático. Fuente: elaboración propia.

El 3.37 % de la memoria disponible es requerida al aplicar la técnica de umbralización binaria en Python, mientras que en Matlab, dicho valor es del 7.5 %. De igual forma, se requiere del 3.03 % y del 7.3 % de la memoria disponible al aplicar la técnica de umbralización adaptativa de Otsu, en Python y Matlab respectivamente.

Así mismo, en la Figura 9 se ilustra el requerimiento de CPU durante el procesamiento de la imagen de vídeo con fondo estático.

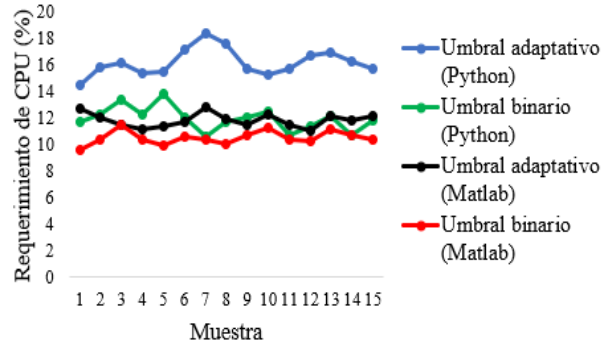

Figura 9. Requerimiento de CPU al umbralizar un video con fondo estático. Fuente: elaboración propia.

La técnica de umbralización binaria requiere del 11.8 % de la unidad central de procesos al aplicarse en Python, mientras que, en Matlab, dicho valor es del 10.3 %. Así mismo, la técnica de umbral adaptativo por el método de Otsu requiere del 15.7 % de la unidad central de procesos al aplicarse en Python, y del 12.1 % al hacerlo en Matlab.

Comparar las técnicas de umbralización binaria y umbralización adaptativa por el método de Otsu en los lenguajes Python y M (Matlab) según el tiempo de respuesta, espacio de memoria utilizada y requerimientos de unidad central de procesos, resulta necesario a la hora de definir las herramientas a utilizar en un sistema de procesamiento de imagen; ya que, si bien en los dos ambientes probados relativos a videos con fondos dinámico y estático se obtienen tendencias similares, también se presentan variaciones en algunos de los picos de los valores obtenidos, que potencialmente tendrían relevancia al trabajar con dispositivos de hardware que presenten limitaciones de memoria y procesamiento.

En la Tabla 1, se muestra la comparativa entre la técnica de umbralización binaria y el método adaptativo de Otsu, según el número de falsos positivos (FP), falsos negativos (FN) y el porcentaje de error respecto al número real de personas (NP) que intervienen en la imagen de video, y el número de detecciones (ND) al aplicar el algoritmo para la sustracción de fondo.

Tabla 1. Comparación de las técnicas de umbralización

	Vídeo con fondo dinámico				
	NP	**ND**	**FP**	**FN**	**Error**
Umbralización binaria	69	57	3	15	22.17 %
Método de Otsu	69	70	5	4	9.28 %
	Vídeo con fondo estático				
	NP	**ND**	**FP**	**FN**	**Error**
Umbralización binaria	62	55	5	12	20.97 %
Método de Otsu	62	68	6	0	9.67 %

Fuente: elaboración propia.

El tratamiento a las imágenes del video con fondo dinámico mediante la técnica de umbralización binaria arroja una tasa de aciertos de 77.83 %, mientras que al aplicar el método de Otsu, dicho valor asciende a 90.72 %. Asimismo, para el video con fondo estático se obtienen aciertos de 79.03 % y 90.33 % al aplicar la técnica de umbralización binaria y el método de Otsu respectivamente.

5. Conclusiones

En videos con fondo dinámico, se obtuvo en promedio un tiempo de respuesta de 0.751 segundos al aplicar la técnica de umbralización binaria, y 0.807 segundos con el método de Otsu. De igual forma, la técnica de umbralización binaria exigió en promedio el 5.541 % del espacio de memoria, mientras que el método de Otsu exigió el 5.281 % del mismo. De la unidad central de procesos, al aplicar la técnica de umbralización binaria se necesitó en promedio del 13.81 % de la misma, mientras que al aplicar el método de Otsu dicho valor aumentó a 16.22 %. Los datos mencionados hacen referencia al promedio de las medias en las pruebas realizadas.

En videos con fondo estático, el tiempo de respuesta promedio al aplicar la técnica de umbralización binaria fue de 0.411 segundos, mientras que al aplicar el método de Otsu dicho valor ascendió a 0.5135 segundos. Igualmente, el espacio de memoria requerido al aplicar la técnica de umbralización binaria fue de 5.435 %, mientras que al utilizar el método de Otsu, este valor disminuyó a 5.165 %. El porcentaje de requerimiento promedio de la unidad central de procesos fue de 11.05 % al aplicarse la técnica de umbralización binaria, y de 13.9 % al utilizarse el método de Otsu. Los datos mencionados hacen referencia al promedio de las medias en las pruebas realizadas.

Python al ser un lenguaje de programación basado en software libre presenta una ventaja respecto a Matlab que es licenciado, al disminuir los costos de implementación. Además, el código en Python tiende a ser menos detallado, por lo que aproximadamente se reduce a la mitad de su extensión. En contraparte, para procesamiento de imágenes, Matlab presenta herramientas de alto nivel para la comunidad científica con amplio soporte matemático para la comprensión de las etapas de procesamiento, mientras que la información disponible en Python se enfoca en su mayoría, exclusivamente al desarrollo desde el punto de vista de la codificación.

En promedio, en 12.095 % mejoró la tasa de aciertos al aplicar el método de Otsu respecto a la técnica de umbralización binaria, debido a que se suprimen los errores por variación de luz y ruido al momento de la captura de la imagen, así como la dependencia de las tonalidades en las prendas de vestir utilizadas por las personas presentes en las zonas de espacio público analizadas, lo que hace relevante el uso del método de Otsu en procesos de videovigilancia, seguimiento y rastreo de personas en espacios abiertos, mediante dispositivos de hardware que no presenten limitaciones de procesamiento y memoria disponible.

Este documento presenta información a tener en cuenta al momento de implementar procesos de detección de personas mediante técnicas de visión por computador al comparar el tiempo de respuesta, memoria utilizada,

requerimientos de unidad central de procesos y aciertos en las detecciones, ya que el rendimiento y eficiencia de la mayoría de los procesos depende en gran medida del desempeño de la herramienta de hardware utilizada. Asimismo, las técnicas de umbralización binaria y umbralización adaptativa por el método de Otsu corresponden a técnicas de gran potencial en el tratamiento de imágenes, y generan tasas de acierto altas que permiten que dichas técnicas sean replicadas en procesos de detección de enfermedades y recuperación de características en imágenes médicas, así como en procesos de identificación y clasificación según la calidad de cultivos, entre otros. Como trabajo futuro, se propone acompañar los procesos de detección por técnicas de umbralización, de etapas adaptativas en la búsqueda de contornos, generando realces y predicción en los mismos, disminuyendo así el número de falsos negativos.

Referencias

[1] M. Leo, G. Medioni, M. Trivedi, T. Kanade, G. M. Farinella, "Computer vision for assistive technologies", *Comput. Vis. Image Underst.*, vol. 154, pp. 1-15, 2017, doi: 10.1016/j.cviu.2016.09.001

[2] D. L. Siqueira, A. Manso Correa MacHado, "People Detection and Tracking in Low Frame-rate Dynamic Scenes", *IEEE Lat. Am. Trans.*, vol. 14, no. 4, pp. 1966-1971, 2016, doi: 10.1109/TLA.2016.7483541

[3] C. V. Niño Rondón, S. A. Castro Casadiego, B. Medina Delgado, D. Guevara Ibarra, "Análisis de viabilidad y diseño de un sistema electrónico para el seguimiento de la dinámica poblacional en la ciudad de Cúcuta", *Ing. USBMed*, vol. 11, no. 1, pp. 56-64, 2020, doi: 10.21500/20275846.4489

[4] C. Raghavachari, V. Aparna, S. Chithira, V. Balasubramanian, "A Comparative Study of Vision Based Human Detection Techniques in People Counting Applications", *Procedia Comput. Sci.*, vol. 58, pp. 461-469, 2015, doi: 10.1016/j.procs.2015.08.064

[5] J. Maldonado Beltrán, C. Peña Cortés, O. Gualdrón González, "Identificación automática de cilindros de almacenamiento de gas utilizando redes neuronales tipo hopfield", *Rev. UIS Ing.*, vol. 11, no. 1, pp. 103-111, 2012.

[6] S. Shoba, R. Rajavel, "Image Processing Techniques for Segments Grouping in Monaural Speech Separation", *Circuits, Syst. Signal Process.*, vol. 37, no. 8, pp. 3651-3670, 2018, doi: 10.1007/s00034-017-0728-x

[7] M. Paul, S. M. E. Haque, S. Chakraborty, "Human detection in surveillance videos and its applications - a review", *EURASIP J. Adv. Signal Process.*, vol. 2013, no. 1, pp. 1-16, 2013, doi: 10.1186/1687-6180-2013-176

[8] E. N. Kajabad, S. V. Ivanov, "People Detection and Finding Attractive Areas by the use of Movement Detection Analysis and Deep Learning Approach", *Procedia Comput. Sci.*, vol. 156, pp. 327-337, 2019, doi: 10.1016/j.procs.2019.08.209

[9] F. T. Aleskerov, V. V. Chistyakov, "The threshold decision making", *Procedia Comput. Sci.*, vol. 17, pp. 1103-1106, 2013, doi: 10.1016/j.procs.2013.05.140

[10] C. V. Niño Rondón, S. A. Castro Casadiego, B. Medina Delgado, "Caracterización para la ubicación en la captura de video aplicado a técnicas de visión artificial en la detección de personas", *Rev. Colomb. Tecnol. Av. RCTA*, vol. 2, no. 36, pp. 83-88, 2020, doi: 10.24054/16927257.v36.n36.2020.3720

[11] B. López-Portilla Vigil, R. Menéndez Alonso, M. Iglesias Martínez, "Implementación del Algoritmo de Otsu sobre FPGA", *Rev. Cuba. Ciencias Informáticas*, vol. 10, no. 3, pp. 16-26, 2016.

[12] P. Huamaní Navarrete, "Umbralización múltiple utilizando el método de Otsu para reconocer la luz roja en semáforos", *Scientia*, vol. 17, no. 17, pp. 247-262, 2016, doi: 10.31381/scientia.v17i17.393

[13] E. Ieno, L. M. Garcés, A. J. Cabrera, T. C. Pimenta, "Simple generation of threshold for images binarization on FPGA", *Ing. e Investig.*, vol. 35, no. 3, pp. 69-75, 2015, doi: 10.15446/ing.investig.v35n3.51750

[14] J. F. Ramos, D. Renza, D. M. Ballesteros L., "Evaluation of spectral similarity indices in unsupervised change detection approaches", *DYNA*, vol. 85, no. 204, pp. 117-126, 2018, doi: 10.15446/dyna.v85n204.68355

[15] N. Triana, A. E. Jaramillo, R. M. Gutiérrez, C. A. Rodríguez, "Técnicas de umbralización para el procesamiento digital de imágenes de GEM-Foils", *Sci. Tech.*, vol. 21, no. 4, pp. 352-261, 2016.

[16] A. Gómez-Villa, G. Díez-Valencia, A. E. Salazar-Jimenez, "A Markov random field image segmentation model for lizard spots", *Rev. Fac. Ing.*, vol. 2016, no. 79, pp. 41-49, 2016, doi: 10.17533/udea.redin.n79a05

[17] M. Huang, W. Yu, D. Zhu, "An improved image segmentation algorithm based on the Otsu method", en *Proc. - 13th ACIS Int. Conf. Softw. Eng. Artif. Intell. Networking, Parallel/Distributed Comput. SNPD 2012*, pp. 135-139, doi: 10.1109/SNPD.2012.26

[18] J. Cortes Osorio, J. Chaves Osorio, J. Mendoza Vargas, "Comparación cualitativa y cuantitativa de las técnicas básicas de umbralización local para el procesamiento digital de imágenes", *Sci. Tech.*, vol. 2, no. 51, pp. 236-241, 2012, doi: 10.22517/23447214.1539

[19] G. Cavanzo, M. Pérez, F. Villavisan, "Medición de eficiencia de algoritmos de visión artificial implementados en raspberry pi y ordenador personal mediante Python", *Ingenium*, vol. 18, no. 35, pp. 105-119, 2017, doi: 10.21500/01247492.3218

[20] N. I. Maya Perfetti, A. M. Nuñez Bedoya, H. A. Romo Romero, "Análisis de rendimiento de algoritmos de reconocimiento de placas de números de vehículos desarrollado mediante la transformación de ondas discretas y la correlación de imagen digital", *Investig. e Innovación en Ing.*, vol. 7, pp. 133-141, 2019, doi: 10.17081/invinno.7.1.2990

[21] G. Sánchez-Torres, J. A. Taborda-Giraldo, "Estimación automática de la medida de ocupación de playas mediante procesamiento de imágenes digitales", *TecnoLógicas*, vol. 17, no. 33, pp. 21-30, 2014.

[22] G. Vargas, O. Neira, R. Arango, D. Fernando, "Métodos de segmentación de nubes en imágenes satelitales Cloud segmentation methods applied to satellite images", *Tecnura*, vol. 17, no. 36, pp. 96-110, 2013, doi: 10.14483/udistrital.jour.tecnura.2013.2.a08

[23] E. S. Gedraite, M. Hadad, "Investigation on the effect of a Gaussian Blur in image filtering and segmentation", en *Proc. Elmar - Int. Symp. Electron. Mar.*, 2011, pp. 393-396.

[24] L. Neumann, A. Vedaldi, "Tiny People Pose", *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11363 LNCS, pp. 558-574, 2019, doi: 10.1007/978-3-030-20893-6_35

[25] J. D. Guerrero Balaguera, "Algoritmos De Procesamiento De Imágenes Y Redes Neuronales Artificiales Para El Reconocimiento De La Lengua De Señas Colombiana (Lsc) Image Processing Algorithms and Artificial Neural Networks To Recognition of Colombian Sign Language (Lsc)", *Rev. Colomb. Tecnol. Av. RCTA*, vol. 2, no. 28, pp. 1-8, 2016.