Detection and Classification of Moving Objects Inside Closed Spaces Using Artificial Computer Vision Algorithms

Detección y Clasificación de Objetos en Movimiento en Espacios Cerrados Utilizando Algoritmos de Visión Artificial

Stalin Marcelo Hidrobo Proaño[1]

[1] Estudiante de doctorado Universidad Politécnica de Valencia
E-mail: stahid@doctor.upv.es

RESUMEN

El campo de visión asistida por computador se ha expandido a lo largo de los años, despertando un gran interés en su estudio y desarrollo investigativo en distintas disciplinas para conseguir el mayor rendimiento de las aplicaciones desarrolladas con estos métodos.

En este artículo se revisó de forma resumida los alcances de una investigación de mayor envergadura, presentada como Trabajo Final de Maestría llevada a cabo por el mismo autor en la Universitat Politècnica de València (2018) en la que se estudió algunas de las principales técnicas de videovigilancia por visión por computador y su comportamiento cuando el entorno cubierto vigilado está habitado permanentemente por una mascota. De tal modo, con la revisión de distintas alternativas, se diferencia entre una alerta generada por el movimiento de ésta y una generada por un elemento intruso, evitando que las alarmas se activen de forma no deseada.

En tal propósito, se usó dos recursos de video: 1) la librería de visión por computador OpenCV en un entorno de C# con el objetivo de poder repetir los experimentos en las mismas circunstancias para los distintos algoritmos a evaluar y, 2) distintos videos que registran animales y personas, simulando el entorno que se pretende vigilar.

El objetivo de este trabajo fue desarrollar un banco de pruebas que permita detectar y clasificar un objeto en movimiento dentro en un ambiente cerrado mediante el uso de algoritmos de procesamiento de imágenes y clasificación.

Al concluir los experimentos, los datos obtenidos fueron dispuestos en tablas que muestran los resultados al ejecutar diferentes secuencias de video en el banco de pruebas, utilizando todos los métodos de procesamiento y clasificación de imágenes desarrollados.

Palabras clave–– visión por computador; OpenCV; procesamiento de imágenes; mascotas; detección de objetos.

ABSTRACT

The field of computer-assisted vision has expanded over the years, awakening great interest in its study and research development in different disciplines to achieve the highest performance of the applications developed with these methods.

This article summarizes the scope of a larger investigation, presented as a Master's Final Project carried out by the same author at the Universitat Politècnica de València (2018) in which some of the main techniques of computer vision video surveillance and its behavior when the guarded covered environment is permanently inhabited by a pet.

In this way, with the review of different alternatives, a difference is made between an alert generated by its movement and one generated by an intruder, preventing the alarms from being triggered unwantedly.

For this purpose, two video resources were used: 1) the OpenCV computer vision library in a C # environment in order to be able to repeat the experiments in the same circumstances for the different algorithms to be evaluated and, 2) different videos that register animals and people, simulating the environment to be monitored.

The objective of this work was to develop a test bench that allows detecting and classifying a moving object inside a closed environment through the use of image processing and classification algorithms.

At the end of the experiments, the data obtained were arranged in tables that show the results when executing different video sequences on the test bench, using all the image processing and classification methods developed.

***Index terms*––** computer vision; OpenCV; image processing; pets; object detection.

1. INTRODUCCIÓN.

El campo de la visión por computador es interdisciplinario y experimenta un acelerado crecimiento debido a sus múltiples áreas de posible aplicación, tanto existentes como en desarrollo.

En sistemas sofisticados de seguridad facilitan, por ejemplo, la detección, rastreo, clasificación e identificación de elementos en movimiento a través de algoritmos que, finalmente, discriminan los eventos cotidianos de los no deseados en un ambiente vigilado según categorías previamente definidas (González Jiménez, 2000).

Estos algoritmos pueden incorporarse en sistemas de seguridad para diferenciar a ocupantes cotidianos de un ambiente —personas o mascotas— de posibles intrusos, para reducir falsas alarmas y minimizar errores humanos en el sistema de vigilancia con personal (Wong & Ong, 2009).

En este documento se busca desarrollar un programa que puede ser usado como banco de pruebas que permita la reproducción de videos pregrabados, aplicar algoritmos de detección y clasificación de objetos en movimiento —personas o animales— y su consecuente discriminación en una secuencia de imágenes para, en un futuro, evitar generar alarmas no deseadas de intrusión.

1.1. ESTRUCTURA DE UN SISTEMA DE VISIÓN ARTIFICIAL.

Para desarrollar una aplicación que requiera de análisis por computador, se debe tomar en cuenta elementos como las fuentes de luz necesarias para iluminar los objetos y los algoritmos a usarse para clasificarlos. Una vez obtenida la imagen digital e introducida en la memoria del computador, se aplica distintas técnicas de procesamiento de imagen y transformación morfológica para segmentar y extraer características que llevan al reconocimiento de patrones. Estos sistemas de visión pueden ser parte de un sistema global al integrarse a otros dispositivos como: robots, autómatas programables, cintas transportadoras o actuadores (De la Escalera Hueso, 2001).

1.2. ELEMENTOS DE UN SISTEMA DE VISIÓN ARTIFICIAL.

Iluminación: Coadyuva en la simplificación del análisis y posterior interpretación de la escena captada (Graham D, 2018).

Fuentes de luz: Estas pueden ser de tipo natural o artificial. Entre otras, tenemos: luces

incandescentes, diodos led, fluorescentes (Bader, Ma, & Oelmann, 2017), fibra óptica, laser, entre otros (De la Escalera Hueso, 2001).

Lente de la Cámara: Permite transmitir la luz hacia el sensor en forma controlada, obteniéndose como resultado una imagen enfocada en varios elementos **(Mushonnifah, Nurhadi, & Pramujati, 2018)**.

1.3. ALGORITMOS DE PROCESAMIENTO DE IMÁGENES.

1.3.1. Segmentación

Se refiere a la partición de una imagen en un conjunto de regiones no solapadas y homogéneas respecto a algún criterio cuya unión cubra la imagen completa. Con ella se busca separar objetos de interés del resto no relevante, considerado como fondo (Rodríguez Morales & Sossa Azuela, 2011).

1.3.2. Operaciones morfológicas

Simplifican las imágenes y preservan las formas principales de los objetos. En visión artificial es frecuente utilizar la morfología para el tratamiento de regiones en el sentido de determinar cómo se pueden cambiar, contar o evaluar. Se puede utilizar para suavizar los bordes de una región, separar determinadas regiones que en el proceso de segmentación las presenta unidas, unir regiones que han sido separadas durante el proceso de segmentación y facilitar el cómputo de regiones en una imagen (Pajares Martinsanz & de la Cruz García, 2007). Asimismo, las operaciones morfológicas se clasifican en:

Secuencia de imágenes en movimiento: Se obtiene con un conjunto de imágenes de la misma escena, capturadas en diferentes instantes y se puede detectar los cambios que haya podido ocurrir en el período comprendido entre ambas capturas (Lavanya & Lohan, 2019).

Reconocimiento de patrones: Es una disciplina científica cuyo objetivo es la clasificación de objetos en cierto número de categorías o clases. Se refiere a esos objetos de forma genérica usando el término patrones. Los enfoques en el reconocimiento de patrones han sido el estadístico y el sintáctico (Pajares Martinsanz & de la Cruz García, 2007).

Blob Detection: "Es un método matemático que detecta regiones o puntos en una imagen digital. Las regiones o puntos se diferencian notablemente con su alrededor. Además, un blob es una región o punto en que algunas propiedades son invariables dentro de un rango de valores establecidos" (Xing & Choi, 2013). A su vez, "puede proporcionar información complementaria sobre las regiones y, por lo tanto, puede ser utilizada para obtener regiones de interés para su posterior procesamiento" (Kay & Bunyarit, 2016).

2. MATERIALES Y MÉTODOS.

Al tratarse de desarrollo de un software para realizar las pruebas de los algoritmos de visión por computador, el material a utilizarse se compone, básicamente, de un entorno de desarrollo y el método utilizado de programación.

2.1. DESCRIPCIÓN DE TECNOLOGÍAS EMPLEADAS

2.1.1. Microsoft Visual C#

Se trata de un lenguaje de programación orientado a objetos (OOP). Se basa en una tecnología de componentes para el desarrollo de software (Mayo, 2002). Es amigable con el usuario y permite crear interfaces gráficas (GUI) con herramientas fáciles de usar. Al ser un software muy difundido, existen varias fuentes de consulta que facilitan el desarrollo de programas (Microsoft, 2020).

2.1.2. Open CV

Es una librería de visión artificial de código abierto, escrita en C++ y funciona con plataformas como: Linux, Windows, y Mac OSX. Existe un desarrollo activo en interfaces para Python, Ruby, Matlab y otros lenguajes. (Kaehler & Bradski, 2008).

2.2. ARQUITECTURA PROPUESTA.

La interfaz de programación que utilizamos para el desarrollo del banco de pruebas es C#.

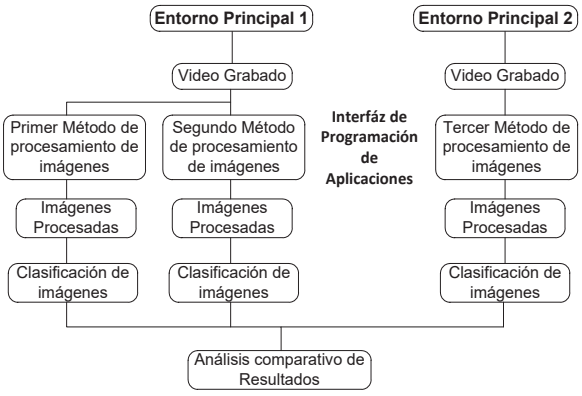

Figura 1: Entornos principales de prueba.
Fuente: Propia.

En la figura 1 se muestra dos entornos principales de prueba[1]. En el primero se procesa los videos previamente grabados e importados hasta el programa. Se desarrolló tres métodos de detección y clasificación y en el segundo entorno se replica el proceso del primero para, finalmente, desarrollar el cuarto método de detección y clasificación.

Los videos utilizados e importados para el banco de trabajo tienen dimensiones demarcadas en píxeles. Así, 640 (ancho) x 360 (largo) a velocidad de 20 fps y formato mp4, grabados previamente utilizando una cámara fija en diferentes instantes, captando a una mascota y una persona por separado.

[1] Para acceder al software de los tres métodos propuestos, visite: https://gitlab.com/-/ide/project/xxxxxx/artificial-computer-algorithms-in-c-sharp/tree/master/-/Main/

2.3. MÉTODOS DE PROCESAMIENTO Y CLASIFICACIÓN DE IMÁGENES.

Se obtendrá diferentes imágenes segmentadas acorde a la necesidad de cada método de clasificación. Con el primer método se obtiene imágenes binarias en blanco y negro; con el segundo se obtiene imágenes de contornos y, con el tercero se obtiene imágenes binarias con un contorno con sombra blanca que muestra el movimiento del objeto.

Tras el procesado de las imágenes, se clasifica las características obtenidas. El objetivo principal de ésta es distinguir un objeto en movimiento y diferenciar entre una persona y una mascota.

2.4. PRIMER MÉTODO

Blob detect: Luego de obtener las imágenes grabadas y traerlas hacia el primer entorno principal, estas imágenes ahora son el punto de partida para continuar con el procesamiento.

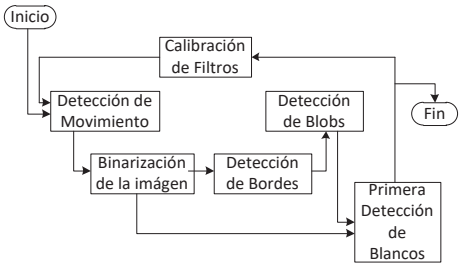

Figura 2: Proceso de tracking.
Fuente: Propia.

Para detectar movimiento, se aplica la resta de *frames* RMI (Recurrent Motion Image) en un intervalo determinado y se utiliza filtros de umbralización tal como se muestra en la figura 3a. La imagen resultante es binarizada utilizando el método truncado, con lo cual obtenemos una imagen de dos bits; nivel alto para blancos y nivel bajo para negros. De ese modo, obtenemos una imagen en blanco y negro que será de utilidad para procesos posteriores de clasificación, usando la cantidad de blanco que existe en la parte alta, media y baja de las imágenes como se observa en la figura 3b:

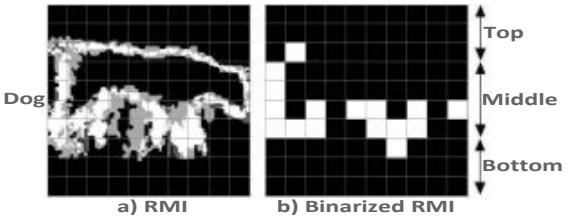

Figura 3: Imágenes RMI y Binarizada
Fuente: Wong & Ong, 2009

Con la imagen binarizada, se utiliza los bordes para definir los límites. Esto reduce significativamente la cantidad de datos y filtra la información inútil, conservando, a su vez, importantes propiedades estructurales de una imagen, previo al siguiente proceso (Detección de Blobs) *(Huamán, 2018)*. Luego de utilizar las funciones de procesamiento de imágenes de suavizado, dilatación y erosión, la imagen se transmite a través de un proceso llamado detector de bordes de canny y, a continuación, la detección de blobs, controlada por los siguientes parámetros: umbral, agrupamiento, fusión, cálculo de centro y radio, circularidad, convexidad, relación de inercia *(Kay & Bunyarit, 2016)*. Obtenidas las imágenes binarizadas y los parámetros de los blobs, se aplica un filtro de detección de blancos cuyo algoritmo permite detectar los primeros blancos que van apareciendo de derecha a izquierda y desde arriba hacia abajo, dibujando una línea blanca similar a la de la figura 4:

Figura 4: Detección de blanco
Fuente: Ruanpeng, Auephanwiriyakul, & Theera, 2017

Con el blob se consigue obtener un coeficiente de relación entre el ancho y el alto del objeto. Por lo general, si se detecta a una persona, el recuadro resulta más alto que ancho; lo contrario ocurre con un animal, que resulta más ancho que alto (Ruanpeng, Auephanwiriyakul, & Theera, 2017).

Con la primera detección de blancos se obtiene un valor de píxeles blancos que contendrá la parte alta, media y baja de cada imagen. Este método es útil cuando la relación de ancho y alto del coeficiente anterior sea de uno y no se pueda discriminar con el blob (Ruanpeng, Auephanwiriyakul, & Theera, 2017). Una persona genera más píxeles blancos que una mascota en las tres regiones del cuadro. Se calcula un coeficiente que resulta de la cantidad de píxeles blancos detectados en la zona alta del cuadro, divido para la sumatoria total de todos los píxeles del cuadro generado. Este es otro parámetro a tomar en cuenta para la clasificación, tal y como se muestra en la figura 5:

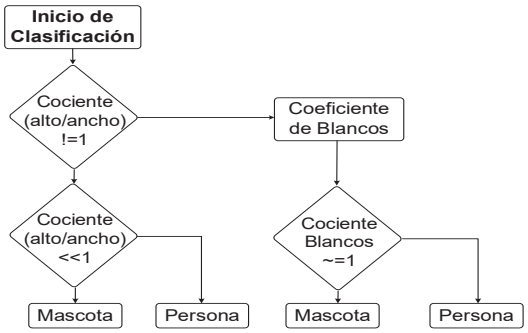

Figura 5: Diagrama de flujo primer método de clasificación
Fuente: Propia.

El diagrama de flujo resume el primer método de clasificación, donde el coeficiente es mucho mayor para una persona gateando que para un animal, al igual que se observa en la figura 6:

Figura 6. Mascota y persona gateando detección de primeros blancos
Fuente: Ruanpeng, Auephanwiriyakul, & Theera, 2017

2.5. SEGUNDO MÉTODO

El segundo método, denominado *surf* —igual que el primero método— se compone de la parte de procesamiento. Se ejecuta los procedimientos para detectar el objeto. Consta principalmente de algoritmos de comparación de objetos con una imagen patrón y para su clasificación se utiliza un árbol de decisiones binarias (Dhivya, Sangeetha, & Sudhakar, 2020). Las imágenes obtenidas del video grabado previamente y traídas hacia el entorno son filtradas y segmentadas utilizando propiedades como el color. Las resultantes contienen regiones potenciales que se ponen a prueba. Se comparan con los parámetros de forma y color para determinar si existen o no coincidencias en las imágenes.

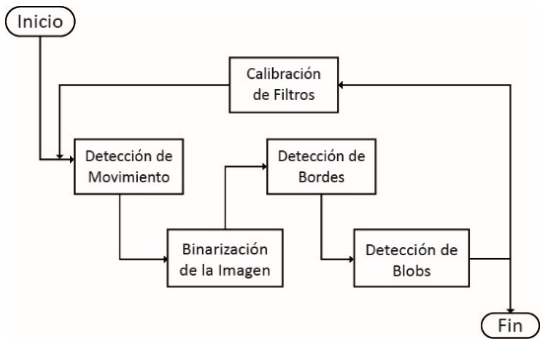

Figura 7. Proceso de detección y reconocimiento de coincidencias
Fuente: Propia.

Para la segmentación de color se utiliza una técnica de división simple, basada en modelo de color HSV de la librería OpenCV. Primero, se filtra la imagen a comparar mediante la técnica de suavizado o alisado gaussiano. La imagen alisada se convierte primero en el espacio de color HSV, luego se divide en una matriz de tono, saturación y valor, con imágenes grises llamadas canales. Tras utilizar las funciones de procesamiento de imágenes de suavizado, dilatación y erosión (igual que en el primer método de procesamiento), la imagen se transmite a través de un proceso de búsqueda de bordes de canny.

En el proceso de detección desarrollado, la forma de la imagen base es primero almacenada en la memoria como un contorno y luego se compara con los contornos de la imagen del objeto en una función que encuentra coincidencias. En la clasificación se realiza una comparación simple de estas entre la imagen patrón y las imágenes candidatas encontradas, en las cuales se determina la cantidad de píxeles semejantes. Si estos exceden un umbral determinado por el usuario, se da como válida la detección, caso contrario, se descarta las imágenes. El siguiente diagrama de flujo ilustra la segunda clasificación:

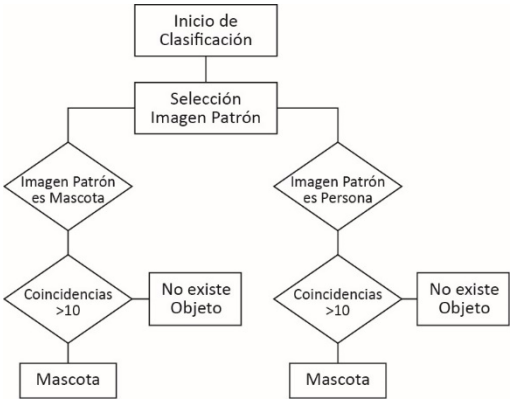

Figura 8. Diagrama de flujo segundo método de clasificación
Fuente: Propia.

2.6. TERCER MÉTODO

El tercer método, *Blob Tracker*, se compone de la parte de procesamiento, al igual que los métodos anteriores. Este ejecuta los procedimientos a seguir para detectar el objeto. Consta, principalmente, de un algoritmo de *tracking* de objetos en movimiento (Akgul, 2010). Para la clasificación se usa un coeficiente de relación alto y ancho. Aunque en inicio este método de procesamiento es similar al primero al basarse en *tracking,* difieren en el principio de procesamiento; mientras en el primer método la mayoría de funciones principales han sido desarrolladas apoyándonos en OpenCV, en el tercero nos valemos, principalmente, de una librería de OpenCV. El proceso de *Tracking Blob* es resumido en la figura 9*:*

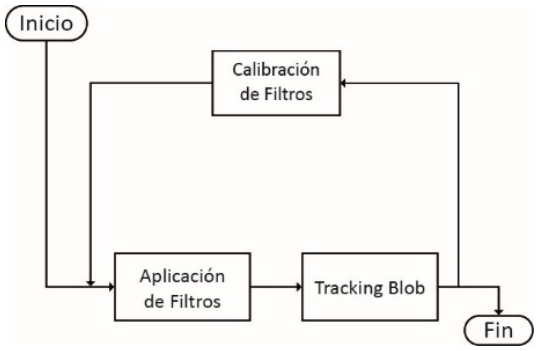

Figura 9. Proceso tracking blob
Fuente: Propia.

El filtro a utilizar en las imágenes obtenidas al inicio es el gaussiano de desenfoque. Se transforma los frames a escala de grises.

El principio del algoritmo *Tracking Blob* se basa en el cambio del tamaño de un objeto en movimiento, además que las estructuras de la imagen pueden variar con el tiempo. Este algoritmo, implementado por la librería OpenCV, trata los problemas de detección de blobs. Adicionalmente, presenta un marco combinado para el seguimiento de características de la imagen que en cada momento se detecta (Bretzner & Lindeberg, 1998). Este método discrimina la detección de movimiento de acuerdo a un coeficiente de relación de alto y ancho de una detección de movimiento encontrada, según se explica en la figura 10:

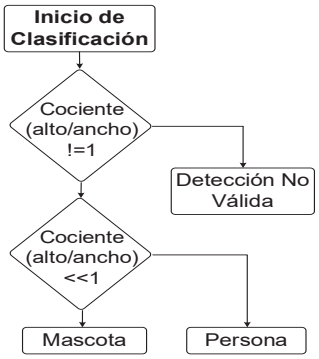

Figura 10. Diagrama de flujo tercer método de clasificación
Fuente: Propia.

Este método de clasificación no contempla el caso en el que el coeficiente de relación (alto/ancho) sea igual a uno, por lo que se tomará como una detección no válida si esto ocurre.

3. RESULTADOS

Se realizó pruebas con distintos archivos de video, usando una cámara fija dentro de un escenario fijo; variando el objeto a detectar. En unos casos se utilizó una mascota y en otros una persona. Con el fin de encontrar falsos positivos, también se utilizó un video en el que aparece una persona gateando.

El número total de videos importado para analizar es 13, todos en formato mp4, con tamaño de 640x360, 20fps y con duración no mayor a un minuto. El computador utilizado tiene un Procesador Intel Core i7 2,6 Ghz, Memoria Ram de 16GB, sistema operativo de 64 bits Windows 8.1.

Se realizó la detección aplicando los tres algoritmos desarrollados, a los *frames* de: una persona de pie y agachada, un niño de pie y un gato, como se muestra en la figura 11:

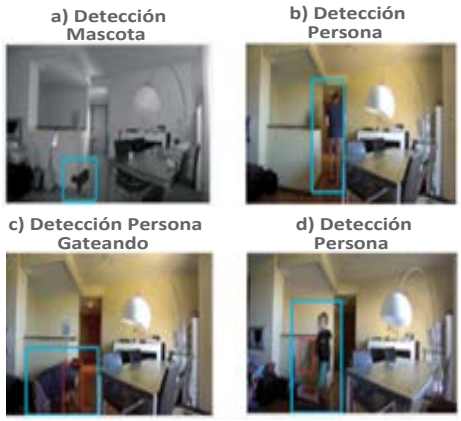

Figura 11. Resultados detección diferentes escenarios blob detect
Fuente: Propia.

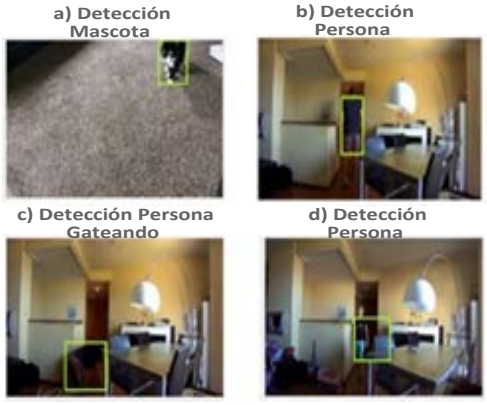

Figura 12. Resultados detección diferentes escenarios surf
Fuente: Propia.

Tras la ejecución de los videos en los respectivos programas, se realizaron tablas de cada proceso. En la tabla 1 se resume los valores obtenidos por el primer algoritmo de procesamiento.

Figura 13. Resultados detección diferentes escenarios blob tracking
Fuente: Propia.

En la columna "No Válida", se apreciará que todos los valores son cero, ya que este método no detecta un objeto que no esté en movimiento, por lo que no existen falsos positivos en la detección, pero sí en su clasificación, ya que el algoritmo puede confundir, en contadas ocasiones, a una persona con una mascota y viceversa.

Tabla 1
Resultados Programa Blob Detect

Objeto a detectar	Detección			Total de Frames Analizados
	Válida	No Válida	Con error de clasificación	
Mascota 1	62	0	12	960
Mascota 2	0	0	0	1008
Mascota 3	32	0	4	1104
Mascota 4	0	0	0	1200
Persona	65	0	7	768
Mascota 5	0	0	0	360
Mascota 6	40	0	7	576
Mascota 7	39	0	8	480
Mascota 8	32	0	9	1008
Mascota 9	1	0	1	864
Mascota 10	0	0	0	720
Niño	82	0	1	288
Gateador	84	0	50	1344

Fuente: Propia.

En la tabla 2, se observa los valores obtenidos con el segundo algoritmo de procesamiento. En la columna de detección "Con error de clasificación" no existen valores, ya que el algoritmo de clasificación de este procesamiento no contempla este parámetro (figura 8), debido a que se selecciona manualmente la imagen patrón, pudiendo ser una persona o mascota, por lo que, si en el proceso aparece una detección errónea, esta se la etiqueta como "no válida" directamente.

Tabla 2
Resultados programa surf

Objeto a detectar	Detección			Total de Frames Analizados
	Válida	No Válida	Con error de clasificación	
Mascota 1	10	57	n/a	960
Mascota 2	10	180	n/a	1008
Mascota 3	0	0	n/a	1104
Mascota 4	0	0	n/a	1200
Persona	20	37	n/a	768
Mascota 5	0	2	n/a	360
Mascota 6	0	0	n/a	576
Mascota 7	8	11	n/a	480
Mascota 8	2	16	n/a	1008
Mascota 9	1	4	n/a	864
Mascota 10	0	0	n/a	720
Niño	2	4	n/a	288
Gateador	10	50	n/a	1344

Fuente: Propia.

En la tabla 3 se aprecia los valores obtenidos con el tercer algoritmo. En la columna "Detección" se evidencia escasas detecciones "Válidas", así como "No Válidas" y "Con error de clasificación", ya que, de los tres, este algoritmo tarda más; explicado con la ayuda de la columna "Total de Frames Analizados", pudiéndose comparar el total de imágenes analizadas con las detecciones válidas y no válidas, reconociendo que las detecciones son mínimas en comparación con la cantidad de imágenes en todos los videos.

4. DISCUSIÓN.

Los parámetros de análisis son: detecciones válidas, detecciones no válidas, detecciones con error de clasificación, tiempo de procesamiento y porcentaje de CPU utilizado.

En la figura 14 se aprecia la comparación de detecciones válidas generadas por los tres métodos de procesamiento.

Tabla 3
Resultados Programa Blob Tracking

Objeto a detectar	Detección			Total de Frames Analizados
	Válida	No Válida	Con error de clasificación	
Mascota 1	1	0	0	960
Mascota 2	2	4	0	1008
Mascota 3	0	0	0	1104
Mascota 4	4	4	1	1200
Persona	60	0	0	768
Mascota 5	0	0	0	360
Mascota 6	10	0	5	576
Mascota 7	1	0	1	480
Mascota 8	1	2	0	1008
Mascota 9	0	7	0	864
Mascota 10	0	0	0	720
Niño	2	0	0	288
Gateador	4	0	0	1344

Fuente: Propia.

Resulta evidente que el método Blob Detect arroja más detecciones válidas, seguido por el método Surf y, por último, se encuentra el Blob Tracking. Esto es entendible ya que, el primer método es el más robusto de los tres y tiene más parámetros de discriminación en comparación de los otros dos.

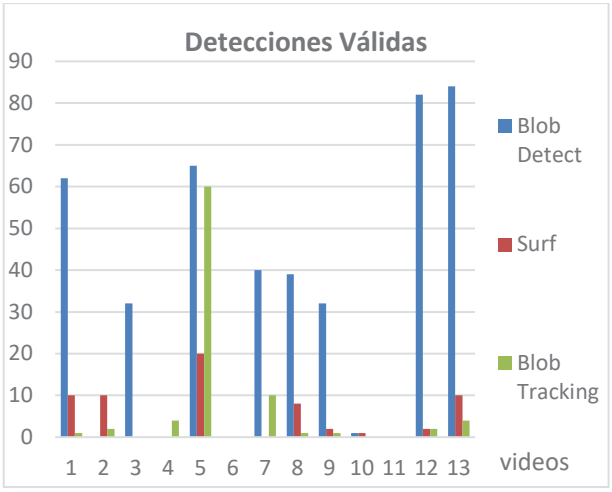

Figura 14. Detecciones Válidas
Fuente: Propia

Referente a las detecciones no válidas, en la figura 15 se observa que el método *Surf* registra un mayor número, seguido por el *Blob Tracking* y no existen detecciones no válidas para el método *Blob Detect*. Esto debido a que el segundo método contempla comparaciones con imágenes patrón y en estas se puede confundir la imagen de fondo luego de haberlas filtrado.

En cuanto al tiempo de procesamiento en relación al tiempo de reproducción del video, siempre este será menor que aquél, debido, justamente, al desfase propio provocado por los cálculos del programa durante su ejecución.

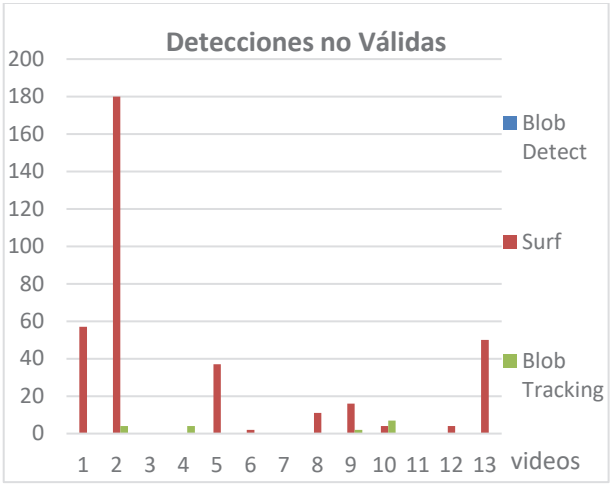

Figura 15. Detecciones no Válidas
Fuente: Propia

La figura 16 muestra que el tiempo de procesamiento del método Blob Tracking en la mayoría de los casos, excepto en los videos 1, 2 y 13, es el mayor de los tres. Esto se debe a que el algoritmo de tracking utilizado ralentiza el proceso.

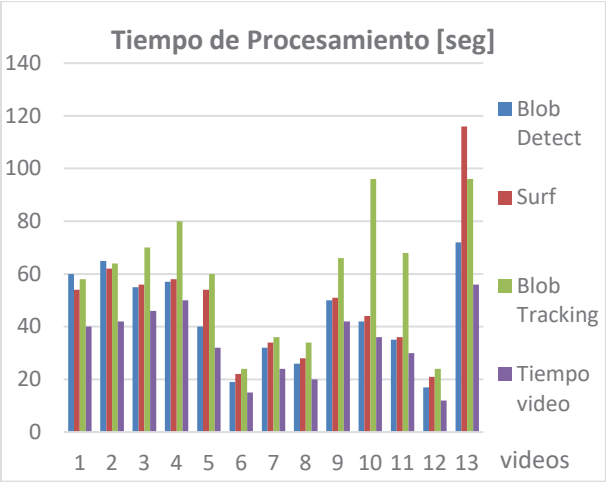

Figura 16. Tiempo de Procesamiento
Fuente: Propia

En la figura 17 se muestra la detección con error de clasificación, y se observa que el método *Surf* no presenta errores de clasificación; esto, debido a que el algoritmo no contempla tal categoría, ya que la imagen patrón es diferente para cada objeto a detectar (niño, mascota). Lo que sí contempla este método son las detecciones no válidas, las cuales son varias.

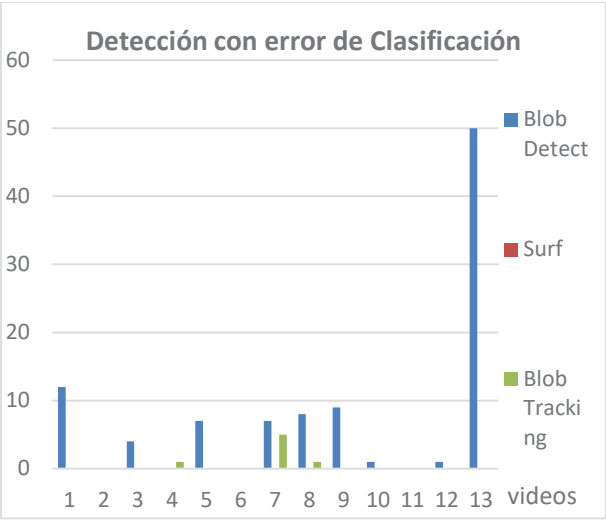

Figura 17. Detección Válida con error de Clasificación
Fuente: Propia

En la figura 18 se aprecia el porcentaje de CPU utilizado al ejecutar los tres programas.

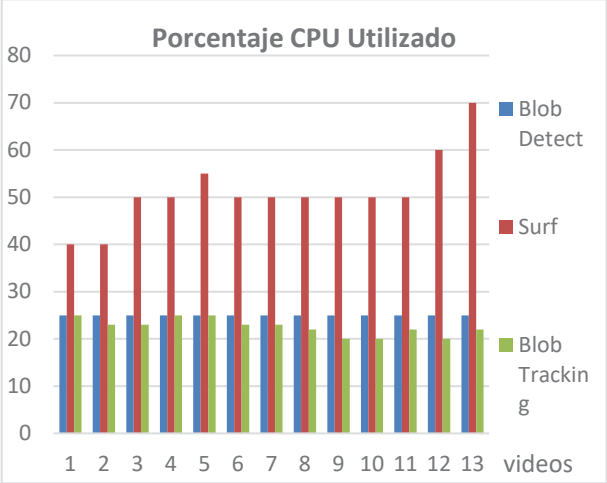

Figura 18. Porcentaje de CPU Utilizado
Fuente: Propia.

El segundo método de procesamiento (*Surf*) es el que más recursos del CPU consume, debido a que los algoritmos de las librerías de comparación de imágenes que contiene OpenCV requieren una gran cantidad de estos para procesar los datos rápidamente; por tal razón, el tiempo de procesamiento de los videos del primer método (*Blob Detect*) no difiere mucho del tercero (*Blob Tracking*).

5. CONCLUSIONES

En este documento se desarrolló un sistema de un banco de pruebas que permitió la ejecución de tres métodos de procesamiento y clasificación de imágenes.

Para ello se ha desarrollado un programa que permita la ejecución de algoritmos y librerías que contienen dichos métodos.

Además, se analizó los resultados presentados en tablas que contenían datos de los diferentes parámetros analizados utilizando gráficas comparativas para conocer el método de procesamiento y clasificación con mejor rendimiento.

Luego del análisis de las gráficas, se puede concluir que el método con mejor rendimiento es el *Blob Detect*, el cual no consume muchos recursos de CPU, su tiempo de procesamiento de imágenes es el esperado y los errores de detección y clasificación son aceptables. Por lo que se lo tomará en cuenta para trabajos futuros.

Asimismo, se plantea introducir el uso de redes neuronales recurrentes (RNN) (Guo, et al., 2016) en la parte de clasificación, ya que en este trabajo se utilizó un simple árbol de decisión binaria y se desea trabajar con redes neuronales para una clasificación más automática y que tenga la capacidad de aprender y alimentar el sistema con datos para mejorar el rendimiento de los métodos de clasificación. Finalmente, se plantea el uso de redes neuronales convolucionales (CNN) *(Wang, et al., 2017)* como método innovador para la detección de objetos.

REFERENCIAS BIBLIOGRÁFICAS.

Akgul, A. Y. (2010). A Fast Method for Tracking People. Trends and Topics in Computer Vision , 143-152.

Bader, S., Ma, X., & Oelmann, B. (2017). Characterization of Indoor Light Conditions. IEEE SENSORS JOURNAL, VOL. 17, NO. 12, 3884-3891.

Bretzner, L., & Lindeberg, T. (1998). Feature Tracking With Automatic Selection of Spatial Scales. Computer Vision and Imaging Understanding, 71, 385 392.

De la Escalera Hueso, A. (2001). En Visión por Computador Fundamentos y Métodos (págs. 12,13,14,15,16,17,18,19,20,21,22). Madrid: PEARSON EDUCACIÓN, S.A.

Dhivya, S., Sangeetha, J., & Sudhakar, B. (2020). Copy-move forgery detection using SURF feature extraction and SVM supervised learning technique. Soft Computing, 14429-14440.

González Jiménez, J. (2000). En Visión por Computador (págs. 1,11,49). Madrid: Paraninfo.

Graham D, F. (2018). Colour and illumination in computer vision. Interface Focus, 1-8.

Guo, T., Xu, Z., Yao, X., Chen, H., Aberer, K., & Funaya, K. (2016). Robust online time series prediction with recurrent neural networks. Proceedings - 3rd IEEE International Conference on Data Science and Advanced Analytics, DSAA, 816-825.

Hornberg, A. (2006). En Handbook of Machine Vision (págs. 88-101, 362-365, 373-379). Ladenburg: Steingraeber Satztechnik GmbH.

Huamán, A. (2018). The OpenCV Tutorials, Release 2.4.13.7. Recuperado el 17 de 07 de 2020, de https://docs.opencv.org/2.4/opencv_tutorials.pdf

Kaehler, A., & Bradski, G. (2008). En Learning openCV computer Vision with the openCV Library (págs. 1-4). United States of América: O'REILLY.

Kay, T. M., & Bunyarit, U. (2016). A Survey of Blob Detection Algorithms for Biomedical Images. 7th International Conference on Information Communication Technology for Embedded Systems 2016 (IC-ICTES 2016), 57-60. Obtenido de https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7467122

Lavanya, S., & Lohan, N. (2019). Performance analysis of moving object detection. Int. J. Spatio-Temporal Data Science, Vol. 1, No. 1, 22-53.

Mayo, J. (2002). En C# Al descubierto (págs. Introducción,4,8,9). Madrid: PEARSON EDUCATION, S.A.

Microsoft. (2020). Tutoriales de Visual Studio C#. Recuperado el 14 de 07 de 2020, de https://docs.microsoft.com/es-es/visualstudio/get-started/csharp/?view=vs-2019

Mushonnifah, S., Nurhadi, H., & Pramujati, B. (2018). Conceptual machine vision design for day and night based on experiment approach. Proceeding - ICAMIMIA 2017: International Conference on Advanced Mechatronics, Intelligent Manufacture, and Industrial Automation, 297-299.

Pajares Martinsanz, G., & de la Cruz García, J. (2007). En VISIÓN POR COMPUTADOR Imágenes digitales y aplicaciones (págs. 146,147,245-254, 338,358,359). Madrid: Ra-Ma.

Rodríguez Morales, R., & Sossa Azuela, J. H. (2011). En Procesamiento y Análisis Digital de IMÁGENES (págs. 155,156,195). Madrid: Ra-Ma.

Ruanpeng, C., Auephanwiriyakul, S., & Theera, N. (2017). Human and dog movement recognition using fuzzy inference system with automatically generated membership functions. IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 1-6.

Wang, D., Zeng, S., Xu, C., Qiu, W., Liang, Y., Joshi, T., & Xu, D. (2017). MusiteDeep: A deep-learning framework for general and kinase-specific phosphorylation site prediction. Bioinformatics, 3909-3916.

Wong, C. E., & Ong, T. J. (2009). A New RMI Framwork for Outdoor Objects Recognition. 2009 International Conference on Advanced Computer Control, 555.

Xing, X., & Choi, B.-J. (2013). Comparative Analysis of Detection Algorithms for Corner and Blob Features in Image Processing. International Journal of Fuzzy Logic and Intelligent Systems vol.13, no.4, 284-290. Obtenido de https://www.koreascience.or.kr/article/JAKO201304536732802.pdf