UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

CENTRO UNIVERSITARIO UAEM ATLACOMULCO

"Detección de incendios mediante identificación de humo con visión artificial en condiciones de iluminación variable"

T E S I S

Que para obtener el Grado Académico de:

Maestro en Ciencias de la Computación

Presenta:

José Luis Nieto González

Director de Tesis:

Dr. Everardo Efrén Granda Gutiérrez

Asesores Adjuntos:

Dr. José Arturo Pérez Martínez

Dr. Allan Antonio Flores Fuentes

Octubre de 2018

Universidad Autónoma del Estado de México
Centro Universitario UAEM Atlacomulco

Atlacomulco, Estado de México a 24 de Septiembre del 2018.

C. JOSÉ LUIS NIETO GONZÁLEZ
CANDIDATO AL GRADO DE MAESTRA EN CIENCIAS DE LA COMPUTACIÓN

PRESENTE:

De acuerdo con el Reglamento de Estudios Avanzados de la Universidad Autónoma del Estado de México UAEMex, y habiendo cumplido con todas las indicaciones que la Comisión Revisora realizó con respecto a su trabajo de Tesis titulado: *"DETECCIÓN DE INCENDIOS MEDIANTE IDENTIFICACIÓN DE HUMO CON VISIÓN ARTIFICIAL EN CONDICIONES DE ILUMINACIÓN VARIABLE"*, la Coordinación de la Maestría en Ciencias de la Computación en este Centro Universitario UAEM Atlacomulco, concede Autorización para que proceda la impresión del mismo.

Sin más por el momento, quedo de usted.

ATENTAMENTE

PATRIA, CIENCIA Y TRABAJO
"2018, Año del 190 Aniversario de la Universidad Autónoma del Estado de México"

C.U. UAEM ATLACOMULCO

DR. EN C.I.E. ALLAN ANTONIO FLORES FUENTES
COORDINADOR DE LA MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN
CENTRO UNIVERSITARIO UAEM ATLACOMULCO

Recibí
24-09-18

Km. 60 Carretera Toluca – Atlacomulco
C.P. 50450
Atlacomulco, Estado de México
Tels. (712) 122 04 36, 122 04 46, 122 0535
e-mail: cuatla@uaemex.mx
Centro Universitario UAEM Atlacomulco

Universidad Autónoma del Estado de México
Centro Universitario UAEM Atlacomulco

Atlacomulco, Estado de México a 24 de Septiembre de 2018.

DR. EN C.I.E. ALLAN ANTONIO FLORES FUENTES
COORDINADOR DEL PROGRAMA DE MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN
CENTRO UNIVERSITARIO UAEM ATLACOMULCO

P R E S E N T E

Por este medio comunicamos a usted que la Comisión Revisora designada para analizar la tesis denominada *"DETECCIÓN DE INCENDIOS MEDIANTE IDENTIFICACIÓN DE HUMO CON VISIÓN ARTIFICIAL EN CONDICIONES DE ILUMINACIÓN VARIABLE"*, que como parte de los requisitos para obtener el grado académico de Maestría en Ciencias de la Computación presenta el **C. José Luis Nieto González** con el número de cuenta **0920032** para sustentar el acto de Recepción Profesional, ha dictaminado que dicho trabajo reúne las características de contenido y de calidad necesarios para proceder la impresión del mismo.

ATENTAMENTE

PATRIA, CIENCIA Y TRABAJO
"2018, Año del 190 Aniversario de la Universidad Autónoma del Estado de México"

Dr. Everardo Efrén Granda Gutiérrez
Tutor Académico

Dr. José Arturo Pérez Martínez
Tutor Adjunto

Dr. Allan Antonio Flores Fuentes
Tutor Adjunto

Recibí 26-09-18

Km. 60 Carretera Toluca – Atlacomulco
C.P. 50450
Atlacomulco, Estado de México
Tels. (712) 122 04 36, 122 04 46, 122 0535
e-mail: cuatla@uaemex.mx
Centro Universitario UAEM Atlacomulco

Universidad Autónoma del Estado de México
Centro Universitario UAEM Atlacomulco

Atlacomulco, Estado de México a 24 de Septiembre de 2018.

DICTAMEN PARA LA AUTORIZACIÓN DE GRADO DE MAESTRÍA

TÍTULO DEL PROYECTO: *"DETECCIÓN DE INCENDIOS MEDIANTE IDENTIFICACIÓN DE HUMO CON VISIÓN ARTIFICIAL EN CONDICIONES DE ILUMINACIÓN VARIABLE"*

TESISTA: José Luis Nieto González

DICTAMEN:

NO. DE REVISIÓN: 1

- [ ] **RECHAZADO**
- [ ] **SUJETO A MODIFICACIONES**
- [ ] **ACEPTADO, CONDICIONADO**
- [x] **ACEPTADO**

OBSERVACIONES GENERALES: Aceptado para su Impresión.
Aceptado para su defensa de grado.

Tutor Académico	Tutor Adjunto	Tutor Adjunto
Dr. Everardo Efrén Granda Gutiérrez	Dr. José Arturo Pérez Martínez	Dr. Allan Antonio Flores Fuentes

Recibí 28-09-18

Km. 60 Carretera Toluca – Atlacomulco
C.P. 50450
Atlacomulco, Estado de México
Tels. (712) 122 04 36, 122 04 46, 122 0535
e-mail: cuatla@uaemex.mx
Centro Universitario UAEM Atlacomulco

DEDICATORIAS

A mi familia, por todo su apoyo en los momentos difíciles y su compañía en las incontables horas de esfuerzo requeridas para completar éste proyecto; por ser mi fuerza y el principal soporte de mis acciones y decisiones.

Sin ustedes, nada de esto habría sido posible.

AGRADECIMIENTOS

Agradezco a CONACYT por el apoyo económico brindado durante el tiempo empleado para la realización de este proyecto.

Al Centro Universitario UAEM Atlacomulco, por permitir mis estudios y la realización del presente trabajo.

Quiero agradecer también a mi director de tesis, el Dr. Everardo Efrén Granda Gutiérrez, su guía a lo largo de este proceso fue invaluable para mi desarrollo, tanto académico como profesional.

Agradezco también a mis asesores adjuntos, el Dr. José Arturo Pérez Martínez y el Dr. Allan Antonio Flores Fuentes, por sus aportaciones y consejos en la realización del proyecto.

A los integrantes del cuerpo académico de la Maestría en Ciencias de la Computación, en quienes encontré siempre valiosas enseñanzas, apoyo y guía.

A mi familia por su apoyo incondicional en todo momento, así como su guía en momentos difíciles y su ayuda para lidiar con las situaciones adversas.

Agradezco a mi esposa, por su amor, compañía, apoyo y paciencia siempre que fue requerida.

RESUMEN

La detección de humo en áreas abiertas representa una gran dificultad para los medios convencionales para detección de incendios. Mientras que la mayoría de los dispositivos utilizados para monitorear la presencia de fuego, están diseñados para trabajar en contacto con alguno producto de la combustión, como la temperatura o la concentración de humo en el aire, las herramientas basadas en Visión Artificial aprovechan las características ópticas del fuego o del humo, permitiendo realizar el monitoreo y la detección de incendios a mayor distancia. Sin embargo, las condiciones de captura de las imágenes complica el proceso. Diferentes niveles de iluminación, condiciones climáticas, así como la presencia de otros objetos móviles reducen el nivel de exactitud de los algoritmos existentes para la detección de humo.

El presente proyecto se enfoca en presentar una propuesta de algoritmo para detección de humo mediante Visión Artificial que afronta el problema de la variación en las detecciones debida a los cambios de iluminación ambiental. Con este propósito, se diseñó un algoritmo compuesto por distintas etapas que analizan las imágenes en busca de características estáticas o dinámicas del humo.

El algoritmo propuesto es descrito en el quinto capítulo de este trabajo escrito. Inicialmente, parte de una etapa de pre-procesamiento que permite ajustar la resolución de las imágenes extraídas desde un video de entrada, balancear la iluminación de las imágenes y etiquetarlas para evaluar la herramienta. Posteriormente, se emplea una etapa que realiza la detección de movimiento, una de análisis de la dirección del movimiento, otra más para el análisis de la información obtenida en espacio de Wavelets y un par de etapas complementarias que analizan el color en espacio RGB y YCbCr.

Finalmente, los resultados son evaluados por una etapa clasificadora basada en la herramienta AdaBoost, para realizar la toma de decisiones y notificar sobre una detección de incendio.

El algoritmo propuesto es evaluado a partir de los criterios de exactitud Sensibilidad (el porcentaje de detecciones correctas realizadas) y Especificidad (el porcentaje de no-detecciones correctamente realizadas). Los resultados de exactitud descritos en el sexto capítulo del presente trabajo escrito, se contrastan con los obtenidos por otros algoritmos replicados a partir del estado del arte.

A partir de los casos de prueba planteados para cada escenario de iluminación evaluado, se identificó una reducción en la variación de los resultados, es decir, el cambio en los porcentajes de sensibilidad y especificidad en diferentes condiciones de iluminación, es menor al obtenido por los algoritmos replicados.

ABSTRACT

Smoke detection in open areas represents a great difficulty for conventional detection means. While most of devices used for fire detection and monitoring are designed to have contact with a combustion product such as heat or smoke concentration on the air, Artificial Vision based tools exploit smoke or fire optical features, allowing to perform fire detection over longer distances. However, video capturing conditions complicate the process. Different illumination levels, climatic conditions as well as the presence of additional mobile objects reduce accuracy of existing smoke detection algorithms.

This project focuses in presenting an algorithm proposal for smoke detection through Artificial Vision which faces the variation on detection levels due to environmental illumination changes. With this purpose, an algorithm composed by different stages was designed, with every stage analyzing pictures looking either for static or dynamic features of smoke.

The proposed algorithm is described on chapter five. It starts with a pre-processing stage which allows adjusting the resolution on images extracted form an input video file, it also allows to balance the illumination on every frame and to assign a tag to it in order to evaluate the tool. A movement detection stage, movement direction analysis stage, another stage used for Wavelet domain information analysis and a couple more stages for color detection on RGB and YCbCr spaces are used later.

Finally, the results are evaluated for a classifying stage based on AdaBoost to perform decision making and notify about fire detection.

Proposed algorithm is evaluated through accuracy criteria of Sensitivity (rate of correctly performed detections) and Specificity (rate of correct non-detections). Accuracy results are described on chapter six, and are also compared with those obtained by other algorithms that were replicated form the state of the art.

From test cases set for each illumination scenario evaluated, a reduction was identified on results variation; it means that sensitivity and specificity rate change under different illumination conditions is lower compared to that of replicated algorithms.

ÍNDICE

DEDICATORIAS iv

AGRADECIMIENTOS v

RESUMEN vi

ABSTRACT vii

ÍNDICE viii

ÍNDICE DE TABLAS xi

ÍNDICE DE FIGURAS xii

1 INTRODUCCIÓN 1

2 PLANTEAMIENTO DEL PROBLEMA 3

2.1 Definición del problema 4

2.2 Objetivos de investigación 5

2.2.1 Objetivo General 5

2.2.2 Objetivos Específicos 5

2.3 Preguntas de investigación 6

2.4 Justificación 6

2.5 Impactos 8

3 HIPÓTESIS 9

4 ESTADO DEL ARTE 10

4.1 Detección de humo mediante características estáticas 11

4.2 Detección de humo mediante características dinámicas 14

4.3 Herramientas para toma de decisión 20

4.4 Métricas para evaluación de los algoritmos 21

4.5 Acondicionamiento de las imágenes 23

4.6 Resumen de la discusión del estado del arte 23

5 MÉTODO 33
5.1 Replica de algoritmos del estado del arte 33
5.2 Diseño e implementación del algoritmo propuesto 39
5.2.1 Funcionamiento general del algoritmo 39
5.2.2 Etapa de pre-procesamiento de secuencias de imágenes 40
5.2.3 Etapa de detección de movimiento. 44
5.2.4 Etapa de análisis en espacio Wavelets 48
5.2.5 Etapa de detección de la dirección del movimiento 48
5.2.6 Etapa de análisis de color 50
5.2.7 Etapa de clasificación mediante el algoritmo AdaBoost 53
5.3 Experimentación 56
5.3.1 Casos de prueba 57
5.3.2 Pruebas preliminares 60
5.3.3 Evaluación de exactitud 60
5.3.4 Comparativa de resultados con el Estado del Arte 61
5.4 Interfaz gráfica 62
6 RESULTADOS Y DISCUSIÓN 66
6.1 Réplicas de algoritmos extraídos del estado del arte 66
6.2 Pruebas Preliminares 72
6.3 Resultados de las pruebas de exactitud 73
6.4 Parámetros de calidad 84
6.5 Comparativa de resultados con el estado del arte 90
7. CONCLUSIONES 101
REFERENCIAS 103
ANEXOS 107

Anexo 1 107
Código de implementación de la réplica del algoritmo de (Toreyin et al. 2005) ... 107
Anexo 2 109
Código de implementación de la réplica del algoritmo de (Vieira et al. 2016) 109
Anexo 3 111
Código de implementación de la réplica del algoritmo de (Zhao et al. 2015) 111
Anexo 4 114
Código correspondiente a la etapa de pre-procesamiento 114
Anexo 5 115
Código de la etapa de detección de movimiento 115
Anexo 6 116
Código de la etapa de análisis en espacio de Wavelets 116
Anexo 7 117
Código de la etapa de detección de dirección del movimiento 117
Anexo 8 117
Código de la etapa de análisis de color 117
Anexo 9 118
Código de la etapa de toma de decisiones mediante el algoritmo AdaBoost 118
Anexo 10 119
Código de la interfaz gráfica de usuario. 119

ÍNDICE DE TABLAS

Tabla 4.1: Matriz de referencias para determinar el estado del arte del proyecto de investigación 24
Tabla 5.1: Especificaciones de los videos de prueba 34
Tabla 5.2: Tipos de videos de prueba para el algoritmo 58
Tabla 6.1: Porcentajes de sensibilidad y especificidad de las pruebas preliminares realizadas a las etapas para detección 72
Tabla 6.2: Pruebas con videos de humo bajo iluminación diurna. 74
Tabla 6.3: Pruebas con videos sin humo bajo iluminación diurna. 76
Tabla 6.4: Pruebas en videos con humo bajo iluminación de día nublado. 77
Tabla 6.5: Pruebas con videos sin humo bajo iluminación de día nublado. 77
Tabla 6.6: Pruebas en videos con humo bajo iluminación nocturna. 79
Tabla 6.7: Pruebas con videos sin humo bajo iluminación nocturna. 79
Tabla 6.8: Criterios de exactitud obtenidos para los distintos valores en el parámetro α 86
Tabla 6.9: Valores de exactitud obtenidos mediante los diferentes métodos para determinar el umbral de energía de frecuencia 89
Tabla 6.10: Resultados de sensibilidad obtenidos por los algoritmos replicados del estado del arte para los casos de prueba establecidos 93
Tabla 6.11: Resultados de especificidad obtenidos por los algoritmos replicados del estado del arte para los casos de prueba establecidos 94
Tabla 6.12: Tiempos de ejecución de los algoritmos analizados en pruebas con diferentes cantidades de cuadros de entrada. 98

ÍNDICE DE FIGURAS

Figura 4.1: Gráficos de evaluación del sistema de Luo et al. (Luo et al. 2015).....22
Figura 4.2: Diagrama del algoritmo propuesto para la detección del humo en secuencias de imágenes.....31
Figura 5.1: Muestras representativas de los videos de prueba. De arriba abajo: Video 1, 2, 3 y 4, como aparecen en la tabla 5.1 34
Figura 5.2: Diagrama de flujo del algoritmo propuesto en (Töreyin et al. 2005) 36
Figura 5.3: Diagrama de flujo del algoritmo propuesto en (Vieira et al. 2016).....37
Figura 5.4: Diagrama de flujo del algoritmo propuesto en (Zhao et al. 2015) 39
Figura 5.5: Diagrama de secuencias de la herramienta propuesta 41
Figura 5.6: Diagrama de flujo de la etapa desarrollada para la extracción de cuadros a partir de un video de entrada. 42
Figura 5.7: Muestra del proceso de redimensionamiento de imágenes 43
Figura 5.8: Diagrama de flujo de la etapa de análisis de energía Wavelets. 49
Figura 5.9: Diagrama de flujo correspondiente a la etapa de detección de la dirección del movimiento 51
Figura 5.10: Diagrama de flujo para la estimación del umbral de análisis RGB.....52
Figura 5.11: Diagrama de flujo del entrenamiento de AdaBoost.....55
Figura 5.12: Diagrama de flujo de la ejecución de AdaBoost 56
Figura 5.13: Diagrama de flujo del etiquetado previo a las pruebas del algoritmo 57
Figura 5.14: Diagrama de casos de uso para de la interfaz gráfica desarrollada 62
Figura 5.15: Estructura gráfica desarrollada para la interfaz de usuario.....63
Figura 6.1: Gráfica comparativa de sensibilidad de los algoritmos en pruebas con videos con humo. 66
Figura 6.2: Gráfica comparativa de especificidad de los algoritmos en videos sin humo. 67
Figura 6.3: Gráficas de los tiempos de ejecución de los algoritmos evaluados con el video de prueba 1. 70
Figura 6.4: Gráficas de los tiempos de ejecución de los algoritmos evaluados con el video de prueba 2. 70

Figura 6.5: Gráficas de los tiempos de ejecución de los algoritmos evaluados con el video de prueba 3. 71
Figura 6.6: Gráficas de los tiempos de ejecución de los algoritmos evaluados con el video de prueba 4. 72
Figura 6.7: Porcentajes de Sensibilidad obtenidos por cada video de prueba con humo en condiciones de iluminación Diurna.....75
Figura 6.8: Porcentajes de Especificidad obtenidos por cada video de prueba sin humo en condiciones de iluminación Diurna.....76
Figura 6.9: Porcentajes de Sensibilidad obtenidos por cada video de prueba con humo en condiciones de día nublado. 78
Figura 6.10: Porcentajes de Especificidad obtenidos por cada video de prueba sin humo en condiciones de día nublado. 78
Figura 6.11: Porcentajes de Sensibilidad obtenidos por cada video de prueba con humo en condiciones iluminación Nocturna.....80
Figura 6.12: Porcentajes de Especificidad obtenidos por cada video de prueba sin humo en condiciones iluminación Nocturna.....80
Figura 6.13: Gráfico de sensibilidad bajo diferentes condiciones de iluminación 81
Figura 6.14: Gráfico de especificidad bajo diferentes condiciones de iluminación. 82
Figura 6.15: Gráfico de variación de sensibilidad a partir de distintos valores del parámetro α. 87
Figura 6.16: Gráfico de variación de especificidad a partir de distintos valores de α 88
Figura 6.17: Gráfico comparativo de la sensibilidad obtenida por los algoritmos analizados.....91
Figura 6.18: Gráfico de comparativo de la Especificidad obtenida por los algoritmos analizados.....92
Figura 6.19: Gráfica comparativa de la variación de sensibilidad de cada algoritmo entre distintas condiciones de iluminación 96
Figura 6.20: Gráfica comparativa de la variación de especificidad de cada algoritmo entre distintas condiciones de iluminación 97

Figura 6.21: Gráfico comparativo de los tiempos de ejecución obtenidos por los algoritmos analizados en pruebas con cantidades diferentes de cuadros de entrada en condiciones de iluminación diurna.....99

1 INTRODUCCIÓN

Los métodos de detección de incendios utilizados comúnmente requieren de la presencia y el contacto con algún producto de la combustión, ya sea calor o humo; esto determina que dichos sistemas sean dependientes de una ubicación y de cierta concentración del elemento correspondiente. La detección de incendios en áreas abiertas es por lo tanto un problema cuando se trata del uso de los sensores convencionales, puesto que no existe un espacio delimitado en el que puedan trabajar de acuerdo a sus especificaciones (Gottuck & Dinaburg 2012).

Los sistemas de Visión Artificial son estudiados como una alternativa para la detección de incendios en áreas abiertas, pues permiten monitorear un área sin necesidad de contacto directo con ningún producto de la combustión. Por otro lado, estos sistemas permiten la detección de dos diferentes elementos de un incendio: el fuego y el humo (Martín-Borregón Domènech 2012). En el primer caso, las características de color e iluminación propias de las flamas, permiten un proceso de detección relativamente más simple; sin embargo, detectar la existencia de fuego es un indicador de que el incendio ya está en proceso y solo permite alertar sobre la necesidad de la extinción de las llamas. El segundo caso, la detección de humo, es un proceso más complicado debido a las características variables del humo, sin embargo permite generar una alerta de prevención, dado que la existencia de humo comúnmente precede a la del fuego (Dimitropoulos et al. 2016).

La detección de humo por Visión Artificial se basa en el análisis de sus características estáticas como el color o la textura, y dinámicas, como su movimiento, trayectoria o su crecimiento. Existen diversas técnicas de tratamiento de imágenes que permiten analizar una imagen y obtener los valores estáticos requeridos, para las características dinámicas es necesario analizar secuencias de imágenes, de modo que se establezca una máscara inicial y se puedan calcular las variaciones entre los distintos cuadros de la secuencia (Martín-Borregón Domènech 2012). Actualmente, proyectos como los presentados en (Deldjoo et al. 2015; Zhao et al. 2015; Labati et al. 2013; Shuai et al. 2016) combinan el análisis dinámico y estático para la detección de humo.

Los sistemas mencionados presentan también diversos problemas, como la interferencia de objetos similares al humo (vapor, niebla, nubes), problemas generados durante la captura de las imágenes (movimientos rápidos de objetos o alteraciones en la posición de la cámara) o los cambios de iluminación en el ambiente que representan un conflicto en sistemas que son estables bajo condiciones controladas.

En este trabajo se propone el desarrollo un sistema de Visión Artificial para la detección de humo, basándose en las características estáticas y dinámicas más significativas del mismo, el cual sea funcional ante cambios de iluminación en las imágenes.

2 PLANTEAMIENTO DEL PROBLEMA

Generalmente, la detección automática de incendios es realizada mediante el uso de distintos sensores de temperatura, presencia de flamas o de humo; sin embargo, estos limitan el área de búsqueda del incendio a un espacio cerrado, en donde además requieren de cierta concentración del elemento en cuestión para poder disparar una alarma, por lo que estas técnicas son dependientes de la orientación y la ubicación de los dispositivos sensores (Gottuck & Dinaburg 2012).

En áreas abiertas, la detección de incendios mediante Visión Artificial representa un área de oportunidad, por lo que es empleada para detectar posibles zonas que requieran acciones de extinción o prevención de un incendio.

El proceso de detección de incendios mediante sistemas de Visión Artificial es típicamente dividido en dos etapas: la primera, que plantea identificar fuego en una imagen, es relativamente sencillo por la diferencia en las características de color y textura en el fuego; la segunda etapa implica la detección de humo como elemento precursor de cualquier fuego. Si bien es importante involucrar ambos procesos, la detección de fuego no permite propiamente la prevención, sino solamente una reacción posterior, mientras que la detección de humo, debido a sus características, no ha demostrado ser completamente precisa (Martín-Borregón Domènech 2012).

Las técnicas de Visión Artificial se encuentran con distintos inconvenientes al detectar humo en imágenes, pues existen diferentes condiciones que pueden complicar el proceso y generar problemas en la detección, o falsas alarmas, que en la aplicación práctica deben ser evitadas; por ejemplo:

1. Las características del entorno, como variaciones en la iluminación durante el transcurrir del tiempo, u otros cambios ambientales (nublados, tormentas).
2. La posición del área de interés (la presencia de humo) dentro de la imagen o ubicación de la cámara que captura las imágenes.
3. Alteraciones en los objetos como distorsiones debidas a la distancia o a movimientos abruptos de los objetos capturados por la cámara empleada, sombras o bloqueos de iluminación generados por otras estructuras (obstáculos).

4. La presencia de elementos similares al humo de los incendios (vapor, nubes, niebla) (Brovko et al. 2013), Esta última de especial importancia, las características estáticas (color, textura) y dinámicas (movimiento, trayectoria) del humo (Deldjoo et al. 2015), permiten que elementos como nubes, niebla, vapor u otros gases, sean fuentes de interferencia en las imágenes analizadas.

Los sistemas detectores de humo, por sensores o por Visión Artificial, no representan una solución definitiva, por lo que continuamente se busca mejorar su exactitud reduciendo el número de falsas alarmas y de no-detecciones equivocadas. Dukuzumuremyi et al. (2014) describen la exactitud en función de cuatro factores: verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos; a partir de estos, definen dos indicadores de desempeño: la sensibilidad que describe la exactitud en las detecciones (cuántas detecciones fueron acertadas) y la especificidad que indica la exactitud en las no-detecciones. Por lo tanto, el objetivo de los sistemas de detección de humo es mejorar en estos índices de exactitud.

2.1 Definición del problema

La detección de humo en áreas abiertas, como mecanismo de detección de incendios, tiene aún problemas importantes que se deben afrontar. Emplear técnicas de Visión Artificial para este fin elimina la dependencia del sistema detector al contacto con el humo, lo que en principio permite una detección en áreas donde puede existir una rápida disipación del humo, pero involucra nuevos factores, como la iluminación y condiciones ambientales durante la captura de las imágenes, así como la existencia de falsas alarmas ocasionadas por la presencia de objetos con características similares al humo.

El estado del arte describe algunos proyectos de investigación en los que las condiciones de análisis de las imágenes son controladas, por ejemplo, utilizando un entorno con iluminación fija, lo que permite demostrar el funcionamiento de los algoritmos de análisis, pero no permite evaluar su desempeño ante condiciones reales, en las que los sistemas mantengan su funcionamiento ante cambios en el entorno físico.

Dukuzumuremyi et al. (2014) proponen un enfoque alternativo a las variaciones de iluminación, clasificando los videos e imágenes de prueba en ocho categorías, de las cuales cuatro son empleadas para la detección de fuego mientras que las restantes (empleadas para detectar humo) son divididas en condiciones de día y de noche (una con humo y una sin humo para cada condición de iluminación). Sin embargo existen diversas condiciones luminosas en cada uno de los grupos contemplados en el trabajo citado, por lo que la clasificación propuesta para la investigación que se pretende desarrollar se divide de la siguiente manera:

1. Luz diurna (LD)
2. Día Nublado (DN)
3. Luz Nocturna (LN)

Con lo anterior, se pretende tener una escala con mayor diversidad en las variaciones de iluminación que se aproxima más a las condiciones de una imagen capturada en un entorno real.

2.2 Objetivos de investigación

2.2.1 Objetivo General

Diseñar una herramienta computacional para detección de humo en áreas abiertas mediante Visión Artificial que mantenga un 80% en la sensibilidad (porcentaje de detecciones positivas correctas con respecto al total de detecciones) y un 85% de especificidad (porcentaje de “No detecciones” positivas correctas con respecto al total de “No detecciones”) bajo condiciones de iluminación diurna, tolerando adicionalmente los cambios de iluminación.

2.2.2 Objetivos Específicos

1. Caracterizar el humo en función de sus propiedades estáticas (color, textura, crominancia) y dinámicas (movimiento, trayectoria) en secuencias de imágenes.

2. Diseñar un algoritmo de análisis de secuencias de imágenes para detectar áreas de interés con base en las características dinámicas del humo.

3. Identificar un algoritmo de tratamiento de imágenes que permita la evaluación de las áreas de interés mediante las características estáticas.

4. Implementar una herramienta que permita decidir si existe o no humo a partir de los resultados de los análisis previos

5. Evaluar el comportamiento del sistema en distintos casos de prueba con las variaciones de iluminación propuestas para estimar su sensibilidad y especificidad.

2.3 Preguntas de investigación

1. ¿Cómo definir un modelo de las características del humo considerando sus posibles variaciones en color, textura u otros?

2. ¿Qué características permiten diferenciar al humo de objetos similares como nubes o vapor?

3. ¿Qué herramientas pueden ser utilizadas para identificar las características del humo en imágenes?

4. ¿Cuáles son las herramientas de cómputo que se pueden emplear para realizar tanto el algoritmo de tratamiento de las imágenes como la toma de decisiones para la detección?

5. ¿Cómo mejorar la detección de humo en diferentes condiciones de iluminación?

6. ¿Cómo debe ser el análisis de la imagen para detectar humo en ella?

2.4 Justificación

Los incendios son siniestros que generan pérdidas económicas, personales y ambientales alrededor del mundo. La detección de incendios mediante herramientas de Visión Artificial no ha alcanzado a resolver completamente el problema debido a que, por un

lado, identificar flamas permite implementar acciones reactivas solamente, aun cuando las herramientas utilizadas para esto presentan un buen desempeño, y por otra parte, la existencia de más objetos en el ambiente con características similares al humo representa un reto que no ha sido del todo resuelto (Dimitropoulos et al. 2016). Como sub-producto de la combustión, frecuentemente el humo se presenta de forma anticipada al fuego, por lo que diversas técnicas se han centrado en detectar este elemento para aportar un tiempo valioso para las tareas de extinción temprana.

Los sensores detectores de humo convencionales, si bien son económicos, no presentan un funcionamiento óptimo en exteriores e incluso en áreas cerradas de cierta extensión; más aún, requieren de cierta concentración de humo en el ambiente para poder alertar sobre un posible incendio. Una opción a estos sensores es la detección del humo a través de imágenes. Éste proceso permite la detección en áreas amplias y exteriores (Gottuck & Dinaburg 2012).

Diversos autores han enfocado sus investigaciones en mejorar este proceso, sin embargo, las características mismas del humo como su apariencia visual ampliamente variable en rangos de color, textura, e incluso de movimientos, lo convierten en un elemento difícil de caracterizar; además, las condiciones de iluminación, ambientales, fondos dinámicos en las imágenes y movimientos abruptos en las cámaras empleadas afectan el proceso (Besbes & Benazza-Benyahia 2016).

El presente proyecto se enfoca en proponer una herramienta de detección de incendios a través de secuencias de imágenes que permita identificar el humo en condiciones ambientales y de iluminación desfavorables, conservando una proporción de detecciones y de “no-detecciones” correctas igual a la media de los resultados presentados en el estado del arte y tolerando a la vez distintas condiciones de iluminación.

Las herramientas de análisis de movimiento y su dirección, así como el uso de transformadas wavelets para identificar características dinámicas, comprenden una primera etapa del algoritmo propuesto. Así mismo una etapa de análisis de características estáticas mediante el procesamiento en distintos espacios de color es propuesto como ajuste para depurar la detección del humo. Por último, se propone el uso

de un algoritmo de clasificación basado en Adaboost que permita discernir entre las imágenes que cuentan con humo y las que no.

Dada la naturaleza del problema, el funcionamiento en tiempo real del sistema no es fundamental para el alcance del presente proyecto, puesto que los análisis de las características dinámicas requieren de la comparación de cuadros en los que las variaciones como movimientos pueden no ser del todo detectables entre dos cuadros consecutivos (Deldjoo et al. 2015).

2.5 Impactos

- Científico: El desarrollo de una herramienta de tratamiento de imágenes y secuencias de imágenes bajo diferentes condiciones de iluminación que permita realizar la detección de humo por Visión Artificial y mantener los porcentajes de funcionamiento en luz diurna tolerando cambios en las condiciones de iluminación.

- Tecnológico: Un algoritmo computacional de tratamiento de imágenes que permitirá la detección de incendios mediante la identificación de humo en espacios abiertos, donde los sensores de concentración de este elemento no cuentan con el mejor funcionamiento y la Visión Artificial plantea una solución alternativa.

Cabe mencionar que la posible implementación de la herramienta propuesta podría involucrar un impacto ecológico, al permitir la detección temprana de incendios en áreas monitoreadas por el sistema.

3 HIPÓTESIS

La implementación de un algoritmo para el tratamiento de las características estáticas como el color, la textura y crominancia, en conjunto con las características dinámicas como la trayectoria o el crecimiento del humo en las secuencias de imágenes permitirá mantener un 80% de detecciones positivas correctas (sensibilidad) y 85% de "No detecciones" correctas (especificidad) en la detección de humo bajo condiciones de iluminación en luz diurna, tolerando también cambios de iluminación.

4 ESTADO DEL ARTE

La Visión Artificial se emplea para detectar incendios sin la dependencia del contacto directo con los productos de la combustión; si bien puede ser útil en espacios cerrados, es en áreas abiertas donde presenta ventajas sobre los mecanismos convencionales de detección de humo, calor o fuego, al permitir la detección a una mayor distancia, así como la identificación de la dirección del incendio (Dimitropoulos et al. 2016; Labati et al. 2013). A partir de esto, hay dos tendencias principales para la detección de incendios a través de las imágenes: detección del fuego y detección de humo.

En múltiples casos, se menciona a la detección de fuego como un proceso más sencillo que la detección de humo, debido a que las características propias de las flamas capturadas en las imágenes, facilitan su procesamiento. En diversas fuentes, se expone al análisis del color como método esencial para identificar fuego en una imagen; Borregon-Domenech (Martín-Borregón Domènech 2012) establece que debido a que el espectro de colores presente en las imágenes de fuego es diferente al que se encuentra en otros objetos, es posible realizar una separación de estas tonalidades mediante el análisis de los componentes de color RGB. Este espectro corresponde con los componentes primarios ampliamente utilizados en los sistemas de cómputo actuales, formado por los colores Rojo, Verde y Azul, por sus siglas en inglés, RGB (Chen et al. 2016).

En (Pritam & Dewan 2017) describen diversas técnicas para detección de fuego, como el análisis de textura de las flamas, el análisis temporal de la región de fuego, la detección de fluctuaciones en las flamas, la detección del área de dispersión de flamas, detección de movimiento y mediante análisis de color. Cada herramienta, en el caso de fuego, se enfoca en realizar una primera detección y analizar la región correspondiente de pixeles en los cuadros siguientes, para verificar la presencia del comportamiento característico de las flamas.

En (Dukuzumuremyi et al. 2014) se realiza la detección del fuego en interiores bajo dos estados de iluminación controlada (Día y Noche) con variaciones agregadas mediante el encendido/apagado de algunas luces. Inicialmente pre-determinan un mapa de disparidad para realizar la sustracción del fondo; utilizan una red neuronal difusa de gradiente descendiente para la toma de decisiones a partir de los valores de media

aritmética, media geométrica, desviación estándar, simetría y entropía obtenidas por la aplicación de la Transformada Discreta de Wavelets (DWT, por sus siglas en inglés) y la Transformada Discreta de Coseno (DCT, por sus siglas en inglés).

En otro proyecto, Wong y Fong (2014), realizan la detección de fuego en interiores mediante una etapa de segmentación, en la que extraen los pixeles de fuego a partir del análisis en RGB y YIQ. Este último es un especio de color compuesto por la luminancia (Y) la fase (I) y la cuadratura (Q), empleado comúnmente en sistemas televisivos de modulación por amplitud.

También se ha trabajado en detección de incendios a través de imágenes satelitales; por ejemplo en (Muñoz et al. 2007), utilizan la detección en éste tipo de imágenes considerando las características de color para entrenar una red neuronal de tipo perceptrón, que se encarga de la detección. Sin embargo, la interferencia atmosférica de nubes u otras condiciones climáticas impide la identificación de fuego, al bloquear la presencia de las llamas.

Por otro lado, la detección de humo, es comúnmente considerada como una técnica perceptiblemente más complicada que la detección de fuego, pues sus características son distintas; en general se consideran características estáticas y dinámicas. El color, la textura e incluso el fondo de la imagen se clasifican como estáticas, mientras que el movimiento, trayectoria, fluctuaciones o vacilaciones y su frecuencia, son consideradas como dinámicas (Deldjoo et al. 2015).

En (Labati et al. 2013), se consideran dos enfoques a partir de las técnicas de visión empleadas en la detección de humo: sistemas basados en imágenes sencillas, que incluyen herramientas de tratamiento como los histogramas o los sistemas basados en secuencias de imágenes, que permiten buscar y contrastar las características dinámicas.

4.1 Detección de humo mediante características estáticas

La detección de las características estáticas implica inicialmente un algoritmo para detectar el color o la textura, que a diferencia del fuego, pueden no estar del todo definidos. Las imágenes están compuestas por diversas intensidades que de acuerdo con

los sensores empleados para capturarlas, corresponden a un espectro de color; la combinación de diversos espectros puede permitir a las imágenes capturadas contener cierto número de colores. Estas combinaciones son conocidas como espacios de color. El más común es el espacio de color RGB donde cada pixel puede tener un valor de intensidad de entre 0 y 255 para cada canal, lo que resulta en un total de 16,777,216 colores posibles (Vieira et al. 2016).

Una herramienta en la detección del color es la evaluación de contrastes-RGB, que se basa en la descomposición de las imágenes en sus componentes de color R (Rojo), G (verde) y B (azul). En (Deldjoo et al. 2015) consideran que el humo en los incendios generalmente es de colores grises, blancos o negros, dependiendo del material que este quemándose. Dentro del espacio de color RGB, los tres colores comparten una característica; los valores de intensidad de cada matriz son muy cercanos entre ellos; es decir, hay poca diferencia entre la intensidad del canal rojo, la del canal verde y del canal azul en un tono blanco, gris o negro dentro del espacio RGB. Por consiguiente, una comparación entre las magnitudes de los tres canales permite identificar si el color es característico del humo. La ecuación (4.1) describe éste criterio.

$$R \cong G \cong B \tag{4.1}$$

En (Vieira et al. 2016), describen una metodología de análisis en el espacio RGB que parte de un entrenamiento con imágenes de humo en un tono específico; las posibles combinaciones de pixeles obtenidas del entrenamiento son clasificadas en diferentes clusters y posteriormente, cada pixel analizado es comparado con estos clusters.

Shuai et al. (2016) plantean tres criterios estáticos para la detección de humo:

1. Primero, se establece que en las combinaciones RGB de colores grises, los valores rojo, verde y azul de las matrices son muy cercanos entre ellos.

Las ecuaciones (4.2), (4.3) y (4.4), son propuestas como metodología para comprobar la condición esperada en una región de interés:

$$m = max\{R, G, B\} \tag{4.2}$$

$$n = min\{R, G, B\} \tag{4.3}$$

$$|m - n| < a \tag{4.4}$$

Donde a es un valor de umbral que establece entre 15 y 20.

2. El segundo criterio consiste en verificar la proporción entre el área de un bloque de pixeles de la región de interés y el rectángulo mínimo que lo encierra.
3. El último criterio parte del cálculo de la varianza de la región de interés, donde se verifica que ésta se encuentre dentro de un rango de 20 a 200 para comprobar la existencia de humo en la escena.

El principal inconveniente del análisis en espacio de color RGB es su alta sensibilidad a cambios de iluminación. Ya que los incendios analizados en espacios abiertos son propensos a encontrarse con iluminación ambiental variable, el resultado del análisis de RGB no puede ser considerado contundente por sí solo. Una forma de afrontar esta desventaja es complementar la herramienta con un análisis en otro espacio de color (Çetin et al. 2013).

El espacio de color YCbCr es utilizado debido a la separación entre la luminancia y crominancia que conforman las matrices de este espacio. El componente Y del espacio de color es conocido como canal de luminancia y contiene la información relativa a la iluminación de la imagen. Los otros dos componentes contienen la información de color o crominancia en el canal azul (Cb) y el canal rojo (Cr). En (Pandey & Singh 2014) describen una herramienta de comparación en el espacio YCbCr que a partir de una experimentación previa, verifica que el canal Cb se encuentre en un rango de entre 126 y 138, mientras el canal Cr esté en un rango entre 112 y 128..

En (Dukuzumuremyi et al. 2014), al igual que con el fuego, se propone una metodología para la detección de humo de día o de noche, con la condición de poseer un mapa de disparidad previamente establecido, esto es, capturado antes de la imagen que se analice para buscar humo. Posteriormente, se implementan DWT y DCT para obtener cinco valores estadísticos: media aritmética, media geométrica, desviación estándar, simetría y entropía.

Un caso similar se estudia en (Martín-Borregón Domènech 2012), donde entre otras etapas para detección, se utiliza la sustracción de fondo. Asumiendo que el humo genera

un efecto de transparencia, se calcula la distorsión del brillo en la imagen a partir de la ecuación (4.5):

$$\alpha_i = \frac{(B - H)^T \cdot (F - H)}{|B - H|^2} \tag{4.5}$$

Donde α_i es la distorsión del brillo para la imagen i, B es la matriz de colores del fondo pre-calculada, H es la matriz de colores resultantes estimados después de la distorsión (tiende a un color gris claro) y F es la matriz con los colores actuales de cada pixel. A partir de la distorsión del brillo, el color sustraído de la imagen es calculado como la distorsión del color, descrita en la ecuación (4.6):

$$CD_h = \|F - \alpha_i(B - H) - H\| \tag{4.6}$$

La propuesta presentada en (Agrawal & Mishra 2014) evalúa cada pixel en la imagen para posteriormente asignar valores binarios a cada uno, proceso conocido como Patrón Local Binario (LBP, por sus siglas en inglés), Se considera que mediante este algoritmo, es posible operar en diferentes condiciones ambientales y bajo diferentes situaciones de iluminación.

4.2 Detección de humo mediante características dinámicas

Las características dinámicas, detectadas en videos en lugar de imágenes sencillas, implican la posibilidad de un análisis robusto de las secciones candidatas a contener humo en la imagen. De acuerdo con Dimitropoulos et al. (2016), son las características dinámicas las que permiten una mejor descripción del humo y diferenciarlo de otros objetos.

Generalmente, los sistemas que detectan características dinámicas incluyen una etapa de análisis de movimiento, que trata de encontrar los cambios entre distintos cuadros de una misma secuencia, indicando la presencia de un nuevo objeto en la escena o un cambio de posición de uno ya existente. Kim y Wang (2009), establecen como un primer paso comprobar que la cámara no se encuentre en movimiento; mediante la diferencia entre dos cuadros, y la comparación de la matriz obtenida con un valor de umbral pre-

establecido, descartan cualquier parte de la secuencia donde se haya cambiado la posición de la cámara.

Las características dinámicas usualmente no dependen de un espacio de color, así que, se emplea la escala de grises como una representación de los cambios de intensidad en la imagen a través de una sola matriz. En el estado del arte, las herramientas que detectan movimiento requieren de una conversión a escala de grises para posteriormente operar sobre la matriz resultante.

La primera característica dinámica verificada por la mayoría de los sistemas es el movimiento. Se pueden emplear diversos algoritmos para detectar movimiento en una secuencia de imágenes. La solución más básica consiste buscar variaciones de intensidad entre los cuadros de la secuencia. El algoritmo conocido como diferencia de cuadros se basa en encontrar toda variación que supere un valor de umbral definido, esto es, obtener la diferencia entre la intensidad de un pixel en el cuadro A y el pixel con la misma posición en el cuadro B. Si la magnitud de esta diferencia es mayor al valor de umbral, se considera que hay un movimiento. La siguiente ecuación muestra el proceso descrito:

$$R(x,y) = \begin{cases} 1 & |A(x,y) - B(x,y)| > T \\ 0 & |A(x,y) - B(x,y)| \leq T \end{cases} \tag{4.7}$$

Dónde $R(x,y)$ es el valor resultante de la comparación, pertenece al mapa binario resultante R y su posición corresponde a la del pixel analizado (x,y); $|A(x,y) - B(x,y)|$ es la magnitud de la variación del pixel en dos cuadros distintos, es establecida como el valor absoluto de la diferencia entre la intensidad de los pixeles con posición (x,y) en el primer cuadro (A) y en el segundo cuadro (B); T es el valor de umbral con el que se contrasta la variación calculada. Cabe resaltar que para completar el proceso de comparación, la ecuación (4.7) debe ser aplicada a cada pixel de la imagen en cuestión; también es importante destacar que tanto el cuadro A, como el B deben poseer las mismas dimensiones $(m \times n)$, y por tanto, la matriz resultante R tiene las mismas dimensiones.

Si bien el mapa binario resultante de este algoritmo se puede emplear como una máscara para identificar el movimiento en la imagen original, también es sensible a variaciones

de iluminación que no representan movimiento, por lo que en (Collins et al. 2000) se recomienda utilizar más de dos cuadros y complementar con otra herramienta.

Entre las técnicas para la extracción del fondo de la imagen, el método de distribución Gaussiana es empleado para calcular la Función de Densidad de Probabilidad (FDP), y a partir de ésta clasificar los pixeles como primer o segundo plano; en (Martín-Borregón Domènech 2012) se menciona que el cálculo de la distancia de Mahalanobis permite delimitar una "región de confianza": cualquier pixel fuera de esta región es considerado primer plano. La ecuación (4.8), es empleada para calcular la FDP mediante distribución Gaussiana:

$$FDP_{pixel}(x) = \eta(x-\mu,\sigma) = \frac{1}{\sigma \cdot \sqrt{2\pi}} e^{\frac{1(x-\mu)^2}{2\sigma^2}} \tag{4.8}$$

Donde, η es la aproximación por distribución Gaussiana, μ es la media aritmética, σ^2 la varianza y x es la posición del pixel. En Kim *et al* (2014), se propone el cálculo de la matriz de fondo a partir del estadístico de Modelo de Mezcla Gaussiana.

En (Jinlan et al. 2016) se propone utilizar un algoritmo de extracción de fondo con variaciones respecto al método descrito anteriormente. Parte de comparar con un valor de umbral la diferencia entre dos cuadros como se muestra en la ecuación (4.9).

$$D_i = \begin{cases} 255 & |F_i - F_{i-1}| > T \\ 0 & |F_i - F_{i-1}| \leq T \end{cases} \tag{4.9}$$

Dónde D_i representa un mapa binario con la diferencia entre el cuadro actual F_i y el anterior F_{i-1} en escala de grises con respecto al valor de umbral T. Cabe mencionar que de manera similar a la ecuación (4.7), la sustracción planteada en (4.9) se realiza para cada pixel de la imagen, y el resultado es también una matriz con valores binarios.

La matriz de fondo es actualizada a partir del mapa binario de diferencia; de modo que para cada pixel en la matriz de fondo se cumpla:

$$B_i = \begin{cases} B_{i-1} & D_i = 255 \\ \varphi F_i + (1-\varphi) B_{i-1} & D_i = 0 \end{cases} \tag{4.10}$$

En donde B_i es el fondo correspondiente al cuadro actual F_i; B_{i-1} es el fondo del cuadro anterior; D_i es el mapa binario de diferencia calculado anteriormente y φ es un

parámetro que denota la velocidad de actualización del fondo. Éste es establecido como constante con un valor de 0.2 para el experimento descrito.

Posteriormente se calcula un nuevo mapa binario con la diferencia entre el cuadro actual y el fondo. Finalmente, la imagen actual es actualizada como:

$$I_j = \begin{cases} \varphi F_i + (1-\varphi) B_{i-1} & \eta > 8 \\ (B_0 + F_i + B_{i-1})/3 & \eta \leq 8 \end{cases} \quad (4.11)$$

También son empleadas para este objetivo la diferencia de tres cuadros y la sustracción de fondo. La sustracción de fondo consiste en el cálculo de una matriz de fondo que funciona como referencia para separar los pixeles en dos clases:

1. La primera clase corresponde a los objetos que se encuentran en primer plano, y que de acuerdo a la característica considerada, están formados por las regiones de interés dentro de la imagen. En el caso de la detección de movimiento el primer plano son todos aquellos pixeles que presentan una variación con respecto al fondo del cuadro previo en la secuencia.
2. La segunda clase corresponde al resto de los pixeles, que no coinciden con el valor característico del objeto que se busca.

En (Töreyin et al. 2005; Vieira et al. 2016) se emplea una herramienta de cálculo recursivo del fondo basada en un parámetro de actualización α. Todos los pixeles que cumplan la condición de movimiento mantienen el mismo valor del fondo anterior, mientras que los pixeles estáticos son actualizados de acuerdo con la ecuación (4.10); en este caso, la nomenclatura cambia, empleando el parámetro α; la ecuación empleada se muestra en (4.12):

$$B_i + 1 = \begin{cases} B_i(x,y) & (x,y) se\ mueve \\ \alpha B_i(x,y) + (1-\alpha) F_{i(x,y)} & (x,y) no\ se\ mueve \end{cases} \quad (4.12)$$

En (Deldjoo et al. 2015) se describen algunas alternativas para la detección de movimiento en las imágenes; una comparación entre dos cuadros es realizada mediante el cálculo de la diferencia entre los valores de los pixeles uno a uno. Ante la detección de un cambio, un valor binario de 1 es asignado a una nueva matriz resultante, si los pixeles se mantienen iguales, se asigna un 0, de este modo, las secciones con valores binarios de 1, pueden considerarse como secciones con movimiento.

Otra técnica útil para el tratamiento de imágenes es el uso de Transformadas Rápidas de Fourier (FFT, por las siglas en inglés de *Fast Fourier Transform*), que permite una conversión de la imagen a tratar a un dominio de frecuencias, de modo que cada valor obtenido define un espectro para la búsqueda de características dinámicas (Günay 2015).

La Transformada de Wavelets, presenta otra alternativa para obtención de características en el dominio de la frecuencia (Töreyin et al. 2005). El algoritmo se aplica por un lado a columnas y por otro a las filas de pixeles dentro de las imágenes para obtener matrices de salida con los coeficientes de detalle resultantes de la transformada.

Otra característica que describe el comportamiento dinámico del humo es la frecuencia con la que fluctúa en la escena. El análisis de energía en dominio Wavelet permite aprovechar esta información como un indicio de presencia de humo. De acuerdo con (Cetin et al. 2016; Günay 2015), la presencia de humo suaviza los bordes de la imagen y genera una diferencia entre la energía del fondo y de la imagen actual.

DWT es una metodología para descomposición de señales o imágenes a partir de bancos de filtros. Un filtro wavelet estándar, denominado de media banda, produce cuatro sub-imágenes wavelet: la banda baja, cuya información no es significativa para el caso de estudio y las imágenes de detalle horizontal, vertical y diagonal. Se puede calcular la energía wavelet como la suma de los cuadrados de los coeficientes de detalle resultantes de la transformación; esto es (Çetin et al. 2013):

$$w_i(x,y) = (LH_i(x,y))^2 + (HL_i(x,y))^2 + (HH_i(x,y))^2 \qquad (4.13)$$

Donde LH es el coeficiente de detalle vertical, HL es el coeficiente de detalle horizontal y HH el coeficiente de detalle diagonal; w_i es la imagen compuesta por los coeficientes wavelet correspondiente al cuadro i; (x, y) son las coordenadas del pixel evaluado.

La energía wavelet es calculada tanto para la imagen actual como para su fondo, de modo que se obtienen dos imágenes compuestas w_i para la imagen y wB_i para el fondo. Una comparación por bloques de tamaño pre-establecido es realizada para buscar las regiones en donde los bordes se han suavizado. La energía acumulada en los bloques es calculada de acuerdo con la ecuación (4.14) (Cetin et al. 2016).

$$E(b_1, b_2) = \sum_{(b_1,b_2)\epsilon RI} w_i(x + b_1P_1, y + b_2P_2) \tag{4.14}$$

Donde $E(b_1, b_2)$ es la energía acumulada en el bloque; (b_1, b_2) son las coordenadas del bloque analizado; (P_1, P_2) son las coordenadas de cada pixel dentro del bloque analizado (de 8×8 pixeles); RI es una región de interés definida a partir del mapa binario resultante de la detección de movimiento; w_i puede ser la imagen wavelet compuesta del cuadro actual o del fondo.

De acuerdo con la teoría utilizada en el análisis de energía de wavelet, el suavizado de los bordes generado por el humo implica un decremento de la energía wavelet. La diferencia entre los valores de energía del fondo y la imagen actual, indica un decremento en caso de resultar en un valor distinto a cero

En (Dimitropoulos et al. 2016), proponen una metodología para análisis de textura dinámica mediante un sistema lineal, es decir analizan la textura de las regiones de interés en diversos cuadros consecutivos de la secuencia.

Zhao, Zhou y Xu (Zhao et al. 2015), evalúan la consistencia espacio temporal, las fluctuaciones o parpadeos y la textura dinámica comparando diferentes imágenes en el tiempo. Con ello diferencian también el humo de la niebla mediante la trayectoria del movimiento. De acuerdo con su metodología, se identifican tres puntos, que son utilizados para calcular un orden de momento; la ecuación (4.15) define la forma en que se calculan los momentos de cada punto en la región.

$$m_{c_1c_2} = \sum_{x=H_s}^{H_f} \sum_{y=H_s}^{H_f} x^{C_1} y^{C_2} F_i(x, y) \tag{4.15}$$

Donde $m_{c_1c_2}$ es el momento; c_1 y c_2 son valores de 0 o 1 que en combinación indican la posición del punto utilizado dentro de la región, es decir, para el punto inicial c_1 y c_2 son iguales a 0, indicando el punto $(0,0)$; los cambios a valores de 1 indican respectivamente los límites vertical y horizontal; H_s y H_f son los valores inicial y final de las coordenadas horizontales de la región; V_s y V_f son los valores inicial y final de las coordenadas verticales de la región.

Una vez calculados los momentos, los valores de los centroides son calculados de acuerdo con las ecuaciones (4.16) y (4.17):

$$C_x = \frac{m_{10}}{m_{00}} \tag{4.16}$$

$$C_y = \frac{m_{01}}{m_{00}} \tag{4.17}$$

Donde C_x es el centroide horizontal y C_y el vertical. Empleando el teorema de Pitágoras y la función trigonométrica coseno, el ángulo Φ de dirección puede calcularse a partir de:

$$\cos \Phi = \frac{C_x}{\sqrt{C_x^2 + C_y^2}} \tag{4.18}$$

$$\Phi = \cos^{-1} \frac{C_x}{\sqrt{C_x^2 + C_y^2}} \tag{4.19}$$

Si el ángulo calculado es mayor a cero se considera que el bloque tiene un movimiento que tiende a lo vertical

Otro proyecto, descrito en (Luo et al. 2015), contempla un proceso de "condensación" de las imágenes, de modo que puedan identificar dos componentes de la trayectoria, uno vertical y uno horizontal; este enfoque se basa en que ningún otro objeto en las imágenes presentará un movimiento similar al humo, y al detectar el mismo patrón del humo, dispara una alarma de detección.

4.3 Herramientas para toma de decisión

La estructura básica de las herramientas para detección de humo mediante visión artificial es secuencial, es decir, cada una de las etapas consideradas por la herramienta propuesta filtra los resultados, y la siguiente etapa trabaja sobre las regiones que permanecen como candidatas a humo. Los trabajos de (Töreyin et al. 2005; Pandey y Singh 2014) son ejemplos de esto.

En propuestas alternativas a la mencionada estructura secuencial, algunos investigadores han optado por independizar sus etapas de detección, es decir, que cada etapa trabaje con

una misma entrada, para posteriormente implementar una nueva etapa de toma de decisiones que realice la clasificación entre aquellos cuadros con humo y sin humo.

Ko et al (2013) implementan un algoritmo denominado Clasificador de Bosque Aleatorio; con esta técnica ensamblan un grupo de árboles binarios de decisión que determinan si un cuadro pertenece a la clase "Con humo" o "Sin humo" a partir de las características extraídas previamente. Por otro lado, en (Deldjoo et al. 2015), hacen uso de un sistema de interfaz difusa con reglas de tipo Mamdani de múltiples entradas y una sola salida que les permite realizar la clasificación de los cuadros a partir del conocimiento definido por un conjunto de reglas que forman una base de conocimiento.

Dukuzumuremyi et al. (2014) utiliza un sistema difuso similar para entrenar una red neuronal que a partir de los pesos de entrenamiento y los valores de las características extraídas, determina si el cuadro tiene o no humo.

El trabajo de S. Wu y colaboradores (Wu et al. 2015), menciona el algoritmo Adaboost como un método para aumentar la relevancia de los resultados, se puede identificar el uso que se hace de esta herramienta en estos casos, pues es identificada como una opción de aprendizaje de máquinas bastante fiable; los trabajos propuestos en (Zhao et al. 2015; Kim et al. 2014), hacen uso de éste algoritmo también, obteniendo resultados de entre el 85 y el 95% de detecciones correctas.

El algoritmo AdaBoost (acrónimo del inglés Adaptative Boosting) es utilizado como un clasificador que parte de un conjunto de características denominadas "clasificadores débiles" (Kim et al. 2014). Posteriormente crea un conjunto de "aprendices" a través de la actualización de pesos y su ajuste basado en datos de entrenamiento (Wang 2012). Los detalles del diseño particular del algoritmo AdaBoost se exponen en la sección 5.3.

4.4 Métricas para evaluación de los algoritmos

Otra aportación importante en (Wu et al. 2015) son las métricas para la evaluación del sistema, pues identifican una tasa de detecciones y una tasa de falsas alarmas, lo que permite cuantificar el funcionamiento de los algoritmos propuestos en base al porcentaje de detecciones realizadas correctamente y el porcentaje de "no-detecciones" correctas,

trabajando sobre secuencias de imágenes previamente clasificadas con presencia o no de humo.

Aun cuando la validación de los algoritmos desarrollados no es constante en los diversos artículos de la literatura, algunos autores identifican características que pueden evaluar para cuantificar la eficiencia de sus soluciones: la Sensibilidad (relación de detecciones positivas correctas con respecto al total de detecciones) y la Especificidad (razón de los negativos reales y el total de "no-detecciones"), son considerados en los trabajos (Martín-Borregón Domènech 2012) y (Labati et al. 2013).

La Sensibilidad es medida como una proporción de las detecciones correctas, es decir, del número total de casos de prueba en los que el sistema ha detectado humo considerando solo aquellas detecciones correctas (en las que no se haya presentado una falsa alarma) y calcular el porcentaje que representan de todas las detecciones. Por otro lado, la especificidad implica el conteo de los casos de prueba en los que no se haya detectado humo y calcular el porcentaje de "no-detecciones" correctas. La Figura 1, muestra un gráfico con los resultados obtenidos en (Luo et al. 2015), donde se consideran los factores de Falsos positivos y Falsos negativos como parámetros complementarios de los valores de sensibilidad (Verdaderos Positivos) y especificidad (Verdaderos Negativos) descritos anteriormente.

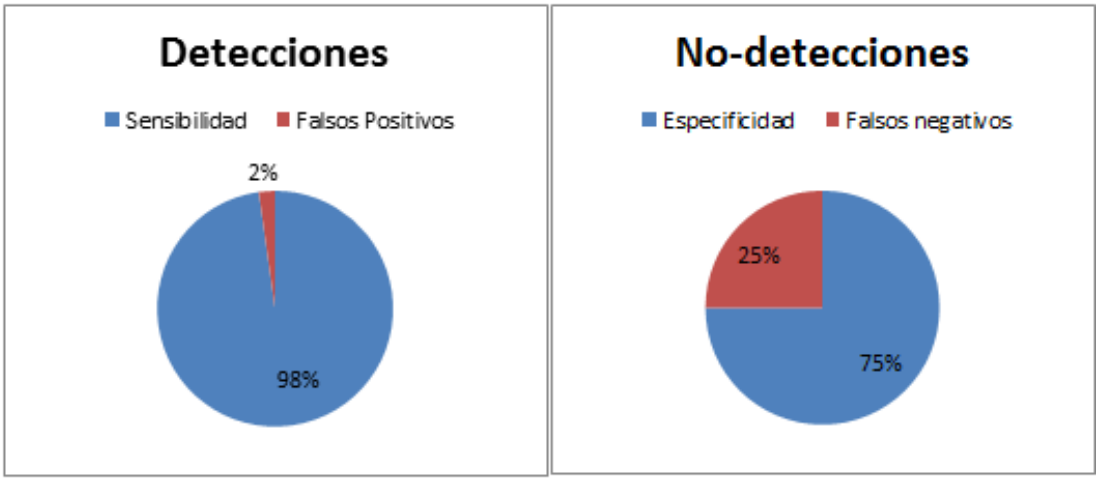

Figura 4.1: Gráficos de evaluación del sistema de Luo et al. (Luo et al. 2015): Sensibilidad respecto al total de detecciones (izquierda); Especificidad con respecto al total de No-Detecciones (derecha)

4.5 Acondicionamiento de las imágenes

En (Besbes & Benazza-Benyahia 2016), se propone un análisis previo de la imagen, donde se obtiene mediante una representación estimada en la matriz RGB la representación de la imagen sin la iluminación para detectar el humo sobre imágenes con "invariantes" de color (valores sin cambio después de una conversión de la matriz de colores RGB de la imagen a un espacio que descarta la iluminación global de la imagen). Se emplea también el componente de "crominancia" con la información de, Croma (C), Intensidad (I) y Saturación (S), que son calculadas a partir de la amplitud de los valores *R*, *G* y *B*, con las ecuaciones: (Besbes & Benazza-Benyahia 2016).

$$C = \max(R, G, B) - \min(R, G, B) \tag{4.20}$$

$$I = \frac{R + G + B}{3} \tag{4.21}$$

$$S = \sqrt{\frac{1}{2}(R - G)^2 + \frac{1}{6}(R + G - 2B)^2} \tag{4.22}$$

Es necesario realizar un acondicionamiento de las imágenes debido por un lado a los cambios en las condiciones de captura entre un video y otro, como los cambios en la resolución de la cámara. Por otro lado, también las condiciones ambientales en las que se captura el video, como la iluminación, pueden causar efectos no deseados en el proceso de detección.

4.6 Resumen de la discusión del estado del arte

De acuerdo con la matriz de referencia mostrada en la Tabla 1, es posible concluir por un lado que las condiciones de iluminación son aún una problemática en la detección de humo en imágenes; si bien en muchos casos se obtiene un porcentaje de detecciones correctas superior a la media de 80%, la iluminación es controlada o no se consideran interferencias adicionales.

Tabla 4.1: Matriz de referencias para determinar el estado del arte del proyecto de investigación

Fuente	Título	Resumen	Metodología Empleada	Áreas de Oportunidad
(Gottuck & Dinaburg 2012)	Fire Detection in Warehouse Facilities	Presenta una recopilación de las herramientas y procesos empleados en la detección de incendios en interiores		Presenta desventajas de los detectores en interiores; para detección por video, considera el fondo, cambios de iluminación, vapor, contaminación.
(Martín-Borregón Domènech 2012)	Sistema de detección de incendios forestales utilizando técnicas de procesado de imagen	Plantea un sistema de detección de incendios mediante el procesamiento de imágenes, detectando por un lado fuego, y por otro humo, obtiene demoras en la detección de entre 5 y 32 segundos, una tasa de detección de entre el 24 y el 98% debido a las distintas velocidades y opacidades del humo y 6 falsas detecciones encontradas en 3 casos de prueba con movimientos de nubes y árboles.	Emplea la detección de color para identificar fuego. Mientras que en la detección de humo desarrolla un algoritmo con los siguientes pasos: 1. Sustracción del fondo 2. Eliminación de elementos rápidos 3. Detección por color 4. Separación por tamaño	1. No emplea otra característica más que el color 2. Considera que el fondo genera un efecto en el humo, cuyos pixeles son semitransparentes.
(Dimitropoulos et al. 2016)	Higher Order Linear Dynamical Systems for Smoke Detection in Video Surveillance Applications	Aprovecha las características dinámicas del humo mediante un modelado de Sistema Dinámico Lineal que les permite analizar el cambio en los pixeles con respecto al tiempo	Emplean el análisis de las características espacio-temporales complementado con el análisis de movimiento obtenidos mediante LDS (Sistema Lineal Dinámico); posteriormente combina los clasificadores mediante Optimización por Enjambre de Partículas para identificar si hay presencia o no de humo	Aprovechan las características dinámicas en el cambio temporal de los pixeles. Emplea la Optimización por Enjambre de Partículas para combinar los clasificadores obtenidos

(Deldjoo et al. 2015)	A Novel Fuzzy-Based Smoke Detection System Using Dynamic and Static Smoke Features	Detección de humo mediante un sistema de inferencia difuso; como resultados, el humo es detectado en algunos marcos con la densidad necesaria.	Emplean 2 etapas de detección de características; en la primera detectan las características dinámicas como movimiento, orientación del movimiento y acumulación del movimiento; en la segunda, detectan características estáticas como color y textura. Posteriormente definen un sistema de inferencia difuso basado en reglas Mamdani MISO; por último, utilizan un proceso de decisión para activar una alarma	No definen métricas de detecciones o falsos positivos. Clasifican los resultados en imagen con Humo (SF) e imagen sin humo (NSF)
(Zhao et al. 2015)	Forest Fire Smoke Video Detection Using Spatiotermal and Dynamic Texture figures	Detección de humo mediante filtrado, segmentado y detección de movimiento, resultando en una tasa de positivos verdaderos y una tasa de negativos verdaderos superior a la obtenida por algoritmos EN-CI y MGM-WT	Utilizan un Filtrado de Kalman para obtener tres características principalmente: 1. Consistencia Espacio Temporal. 2. Característica de Revuelo. 3. Textura Dinámica. Posteriormente utilizan un algoritmo Adaboost para obtener clasificadores fuertes	Diferencian el humo de la niebla considerando que el primero siempre presenta un movimiento con trayectoria vertical
(Labati et al. 2013)	Wildfire Smoke Detection Using Computational Intelligence Techniques Enhanced With Synthetic Smoke Plume Generation	Realizan la detección de humo con cámaras de bajo costo; implementan una etapa de extracción de características en las imágenes y eligen mediante clasificadores de inteligencia computacional	Emplean 2 algoritmos que siguen las mismas etapas: - Detección de regiones con movimiento. - Análisis de color - Detección de ejes - Detección de crecimiento - Detección de elevaciones - Análisis del perímetro - Computo de características	Algoritmos dependientes de la iluminación. Proceso de simulación de humo que lo añade en imágenes para pruebas.

(Shuai et al. 2016)	A novel smoke detection algorithm based on Fast Self-tuning background subtraction	Realiza la segmentación de la imagen buscando las características estáticas del humo para posteriormente evaluar las características dinámicas	Realiza la segmentación de la imagen. - Lleva a cabo un proceso de decisión estático del humo. - Lleva a cabo un proceso de decisión dinámico del humo	No considera variaciones de iluminación ni interferencias. Realiza una conversión de los colores a una escala de grises para eliminar el color de las imágenes antes de ser tratadas.
(Brovko et al. 2013)	Smoke detection algorithm for intelligent video surveillance system	Se basa en dos etapas principales, la detección de vectores de movimiento y el análisis de contraste	Realiza una extracción de fondo. - Realiza un tratamiento previo de la imagen. - Las dos características analizadas utilizan etapas en paralelo.	Emplea la misma herramienta de extracción de fondo. Solamente espera a que los cuadros cumplan con las dos condiciones para generar una alerta
(Dukuzumuremyi et al. 2014)	A Novel Algorithm for Fire/Smoke Detection based on Computer Vision	Realizan la detección de fuego/humo en imágenes con diferentes grados de iluminación, obteniendo un 20% de mejora en exactitud, 27% en precisión en comparación con el método K-Means	Utilizan un método basado en 4 fases: 1. Pre procesamiento: ajuste de la imagen. 2. Segmentación: obtención de un área de la imagen. 3. Extracción de características: Cálculo de valores característicos mediante transformadas. 4. Clasificación: Sistema difuso para toma de decisiones	Establecen 8 clasificaciones para la iluminación, basados en la existencia de humo o fuego en día o noche. Adaptan la imagen a las características que manejan.
(Besbes & Benazza-Benyahia 2016)	A novel video-based smoke detection method based on color invariants	Hace el tratamiento de la imagen mediante indicadores de la iluminación y la crominancia para reducir las variaciones en la iluminación y posteriormente detectar los pixeles de humo	Realiza la estimación del fondo. - Obtiene una representación de las invariantes del color. - Detecta el movimiento por zonas. - Excluye las zonas sin humo usando la crominancia	Realizan pruebas de funcionamiento exclusivamente con una iluminación diurna. Realizan la separación de características del fondo lo cual facilita la detección de humo.
(Pritam & Dewan 2017)	Detection of fire using image processing techniques with LUV color space	Realizan la detección de flamas a través del análisis de color en espacio LUV	1- Extracción de cuadros. 2- Detección de color en LUV. 3- Detección de bordes. 4- Combinar resultados y segmentar la imagen final	Detectan solamente flamas. Utilizan un solo espacio de color que puede ser sensible a cambios de iluminación

(Wong & Fong 2014)	Study of Pool Fire Heat release using Video Fire Detection	Detecta fuego en interiores considerando el color e identifica también su origen y altura de las flamas	Obtiene una imagen a color y con funciones de tratamiento, convirtiéndola a escala de grises, usa distribuciones normales y Rayleigh para comparar los pixeles	Hace la detección exclusivamente de la presencia de fuego
(Muñoz et al. 2007)	Detección de incendios Forestales utilizando imágenes NOAA/16-LAC en la región de la Araucanía, Chile	Realizan la detección de fuego sobre imágenes satelitales, obteniendo un 100% de precisión en áreas mayores a 15 hectáreas y un 50% en áreas de 10 hectáreas debido a que en áreas menores no se alcanza el umbral de detección en el contraste de la imagen	Empleando un sensor de 5 “bandas”, entrenan una red neuronal de tipo “perceptron” para detectar las áreas con focos de incendio.	Solo realizan detección de fuego. Consideran solo 1 conjunto de condiciones ambientales, sin variantes. Su detección es anulada al encontrar nubes en la imagen. No consideran un cálculo de falsas alarmas
(Vieira et al. 2016)	Smoke Detection in Environmental Regions by Means of Computer Vision	Analizan las combinaciones de colores en espacio RGB; implementan un entrenamiento a partir de imágenes con solamente humo	1- Detección de movimiento. 2- Análisis de color en RGB. 3- Análisis de permanencia espaciotemporal	Requiere un entrenamiento específico para distintos tipos de humo.
(Çetin et al. 2013)	Video fire detection – Review	Hace una revisión de los algoritmos propuestos en el estado del arte para la detección del humo		Enlista diversas herramientas y las etapas que emplean.
(Pandey & Singh 2014)	Smoke and Fire Detection	Analizan las imágenes en busca de la información de luminancia y crominancia caracterizada por experimentación previa.	1- Extraer imagen. 2- Conversión a espacio YCbCr. 3- Analizar pixeles de la imagen. 4- Umbralizar	Analizan solo en interiores; utilizan solo la herramienta de análisis de color.
(Agrawal & Mishra 2014)	Smoke Detection Using Local Binary Pattern	Detecta humo ante diferentes tipos de iluminación	Combina los algoritmos LBP y Adaboost, evalúa cada pixel con su vecinos y elige aquellos que cumplen las características y tienen movimiento	Afirman que LBP puede trabajar en diferentes condiciones de iluminación

(Kim & Wang 2009)	Smoke Detection in Video	Proponen un sistema de detección de incendios desde una cámara fija, indicando como resultados el número de cuadros en los que se detectó humo	Detectan cambios de posición en la cámara de acuerdo con la iluminación; posteriormente, detectan bloques de interés de acuerdo al movimiento, comparando toda la imagen inmóvil con imágenes anteriores; detecta el humo mediante el color y la iluminación.	No hace distinción con elementos que puedan tener la misma iluminación
(Collins et al. 2000)	Introduction to the special section on video surveillance	Describe herramientas de visión artificial para la detección de objetos en movimiento	Propone un algoritmo hibrido entre la diferencia de tres cuadros y la sustracción de fondo	Herramienta implementada en diversos algoritmos; la umbralización se propone a partir de matrices de valores
(Kim et al. 2014)	Smoke Detection using GMM and Adaboost	Realiza la detección de humo en exteriores usando valores estadísticos del humo	Emplea un algoritmo de modelado de Mezcla Gaussiana para separar el humo del fondo y Adaboost para detectar el humo	Trabaja solo con imágenes de día y sin presencia de objetos como sombras, nubes o vapor
(Jinlan et al. 2016)	A Method of Fire And Smoke Detection Based on Surendra Background And Gray Bitmap Plane Algorithm	Segmentan la imagen por detección de movimiento umbralizada a partir del método de Otsu y con el cálculo de fondo de Surendra	1- Calculo de umbral; 2- Calculo de fondo; 3- Detección de movimiento; 4- Análisis de mapa de bits	Analiza los pixeles en un espacio de color de Ohta
(Töreyin et al. 2005)	Wavelet Based Real-time Smoke Detection in Video	Verifican la existencia de humo a partir de la características dinámicas en la información de frecuencia de wavelet	1- Detección de movimiento. 2- Análisis de wavelet espacial. 3- Análisis de wavelet temporal. 4- Análisis de color en espacio YUV. 5- Análisis de convexidad	Sigue una estructura secuencial, cada etapa filtra el resultado de la anterior.
(Luo et al. 2015)	Smoke detection based on condensed image	Mediante un proceso de “condensación” de video, identifican trayectorias, frecuencias y proporciones, incrementando la proporción de detecciones correctas hasta un 83%	Procesan las imágenes con una condensación que les permite separar los componentes de la trayectoria del humo en vertical y horizontal; el cruce entre los rangos con mayor movimiento en cada componente (igual al del humo) es el área con humo	Se basan en la premisa de que nada se mueve igual que el humo. Comparan trayectoria del humo con otros objetos

(Wu et al. 2015)	A Real-time images smoke detection using staircase searching-based dual threshold Adaboost and dynamic analysis	Detección de humo en video mediante algoritmo de escalera propuesto con resultados en Tasa de falsas alarmas 2%, y tasa de detecciones 95%	Utilizan un algoritmo de detección de características Haar. Posteriormente usan un método de Adaboost para aumentar la relevancia de los resultados y proponen un algoritmo de búsqueda en escalera, clasificando las secciones de la imagen para detectar posibles concentraciones de humo.	Requiere un proceso de entrenamiento con un gran número de ejemplos

Por otro lado, la inclusión de diferentes etapas de tratamiento que permiten buscar tanto las características estáticas en cuadros sencillos y las características dinámicas en secuencias de imágenes, parece ser una tendencia en el estado del arte; establecer una solución en la que se defina una máscara de fondo (background mask) es una herramienta útil para partir hacia una revisión de los movimientos y elementos nuevos en la secuencia de imágenes.

Herramientas como Adaboost permiten que considerando las características del humo y los valores detectados en la imagen, se emita una decisión sobre la existencia o no de humo en una imagen. Otra alternativa a esto presentada en el estado del arte es el uso de sistemas basados en lógica difusa.

A partir del análisis anterior del estado del arte, se propone emplear las herramientas de análisis de movimiento para filtrar las regiones de interés en las imágenes analizadas; análisis de contrastes en RGB y en espacio YCbCr para detectar secciones de las imágenes en las que las características estáticas coincidan con las del humo; posteriormente, emplear el análisis de dirección de movimiento y la transformada wavelet discreta para depurar la detección identificando si en la secuencia de imágenes se presenta un comportamiento similar al del humo en aquellas secciones marcadas como posibles positivos.

La herramienta propuesta involucra una primera etapa consiste en la extracción de cuadros a partir de la secuencia de imágenes de entrada; posteriormente, una etapa de análisis de movimiento similar a la utilizada en (Vieira et al. 2016) para definir regiones de pixeles en movimiento y evitar analizar secciones de las imágenes que permanecen estáticas y por tanto no son candidatas a tener humo. Las regiones de interés detectadas son procesadas por herramientas de análisis de energía wavelet, que permite encontrar regiones en las que los bordes de los objetos solidos son suavizados por la presencia de humo, y también un análisis de dirección de movimiento que facilita diferenciar el humo de elementos como nubes o niebla por la trayectoria con la que se mueve cada uno. Las características estéticas son analizadas por la herramienta de análisis de contraste de color (Deldjoo et al. 2015) con el que se busca detectar bloques de la imagen en los que los valores *R*, *G* y *B* sean similares (característica de las tonalidades grises). Como parte

de análisis de las características estáticas, el análisis de crominancia es realizado mediante el análisis de las matrices de luminancia (Y) y crominancia (Cb y Cr). La Figura 2, describe el diagrama de flujo de la herramienta propuesta, que se desarrolla con detalle en el siguiente capítulo.

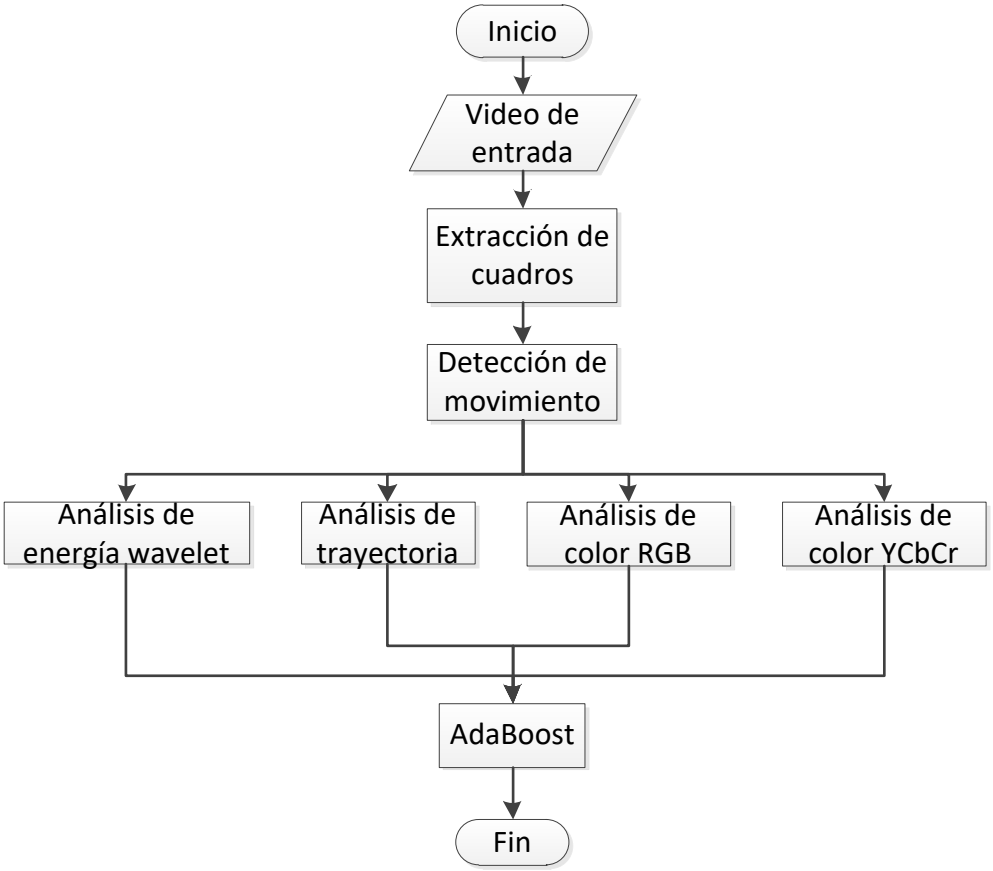

Figura 4.2: Diagrama del algoritmo propuesto para la detección del humo en secuencias de imágenes

El espacio de color YCbCr, presenta otra alternativa al análisis de color RGB, los componentes de crominancia (Cb y Cr), pueden ser separados de la luminancia (Y), permitiendo tratamientos distintos que permiten atenuar el efecto de los cambios de iluminación en las imágenes capturadas (Çetin et al. 2013).

En muchos casos, para la detección de movimiento se emplea el método de sustracción adaptativa de fondo, el cual permite que para cada cuadro de la secuencia se calcule una matriz de fondo, las diferencias significativas entre el cuadro analizado y el cuadro

actual, esto es, las diferencias que superan un valor de umbral, son clasificadas como movimiento (Vieira et al. 2016; Brovko et al. 2013). En (Zhao et al. 2015) emplean el filtrado de Kalman como una alternativa a este proceso.

De acuerdo con el estado del arte, el análisis de movimiento puede complementarse con la identificación de la dirección en la que ocurre éste, proceso realizado mediante el cálculo de un ángulo de trayectoria, en busca de un valor similar al del humo. En general, se busca encontrar una dirección vertical en el movimiento, con lo que en trabajos como (Zhao et al. 2015), realizan éste cálculo en una región de interés mediante centroides de movimiento.

Las transformadas wavelet son empleadas para analizar frecuencias de parpadeo o de fluctuaciones en secuencias de imágenes. Las señales obtenidas de las transformadas tienen un valor cercano a 0 para los pixeles estacionarios, cada parpadeo implica un pico en la señal (Labati et al. 2013). A partir de la información obtenida por la implementación de la transformada wavelet 2D (compuesta por los detalles vertical, horizontal y diagonal de la imagen), se ha propuesto el cálculo de energía de frecuencia en bloques de tamaño delimitado (Töreyin et al. 2005).

Los resultados de los análisis previos son ingresados como clasificadores débiles al algoritmo Adaboost, para que mediante la asignación de pesos a las características analizadas, sea posible indicar una detección o regresar a la extracción del siguiente cuadro.

5 MÉTODO

De acuerdo con el problema planteado, la herramienta para detección de humo debe cumplir esencialmente con dos características: lograr un nivel de detecciones por lo menos igual a la media del estado del arte, y tolerar distintos escenarios de iluminación. Por esto, los algoritmos empleados deben ser ajustados para trabajar bajo distintas condiciones, en especial aquellos que requieren del uso de valores de umbral que son definidos como constantes en el estado del arte, donde trabajan también con condiciones de iluminación estables.

5.1 Replica de algoritmos del estado del arte

Con el objetivo de identificar como se puede evaluar el desempeño de las herramientas de visión artificial y algunos de los algoritmos propuestos, se replicaron tres algoritmos que forman parte del estado del arte:

1. El algoritmo propuesto por Toreyin (Töreyin et al. 2005), seleccionado por su frecuente presencia como base de comparación para las herramientas propuestas en el estado del arte y por el manejo del análisis en el espacio de wavelets.
2. El algoritmo de Vieira (Vieira et al. 2016), elegido por la metodología de análisis en espacio RGB y el uso de etiquetado de las regiones de interés.
3. El algoritmo de Zhao (Zhao et al. 2015), escogido por el uso de la herramienta AdaBoost y los algoritmos de análisis espacio-temporal.

Las etapas correspondientes a cada uno de los algoritmos fueron replicadas para su posterior evaluación. Cada algoritmo fue evaluado con cuatro videos de ejemplo, descritos en la Tabla 5.1.

Los videos 1 y 2 fueron obtenidos del banco de prueba de la universidad de Bilkent (consultado en la dirección http://signal.ee.bilkent.edu.tr/VisiFire/ (Cetin 2003)); el video 1 fue elegido por la cercanía del humo a detectar en condiciones de iluminación diurna mientras que el video 2 fue elegido por su condición de iluminación nocturna. El resto de videos fueron obtenidos por medios propios; La figura 5.1 presenta las muestras representativas de los video utilizados (Nieto-González et al. 2017)

Tabla 5.1: Especificaciones de los videos de prueba

Video	Duración (mm:ss)	Ancho (pixeles)	Alto (pixeles)	Cuadros por segundo
Video 1 (Humo de día)	01:00	320	240	15
Video 2 (Sin humo de noche)	00:15	320	240	10
Video 3 (Humo de día)	01:00	1920	1080	29
Video 4 (Sin humo de día)	00:19	1280	720	30

Figura 5.1: Muestras representativas de los videos de prueba. De arriba abajo: Video 1, 2, 3 y 4, como aparecen en la tabla 5.1

En un primer ejercicio, se evaluaron los algoritmos en función de su exactitud, es decir, obteniendo los porcentajes de sensibilidad y especificidad a partir del pre-etiquetado de las imágenes contenidas en los videos de prueba.

Por otro lado, se realizó un estudio de la complejidad temporal de cada algoritmo replicado, a partir de la medición del tiempo de ejecución bajo las siguientes condiciones:

- Procesador Intel® Core™ i5 a 2.53 GHz
- 6 GB de memoria RAM, 533 MHz
- Sistema Operativo Windows 7 Professional de 64 bits (SP1).
- Tarjeta de gráficos ATI Mobility Radeon HD 550v a 400 MHz con 3762 MB de memoria

En cada caso de prueba las entradas del algoritmo evaluado se modificaron, incrementando la cantidad de cuadros y cambiando la resolución. Las funciones descritas por los valores de tiempo obtenidos, permitieron determinar el grado de complejidad para cada herramienta.

En cuanto a los algoritmos, el propuesto por (Töreyin et al. 2005), consta de cinco etapas de análisis organizadas de manera secuencial:

1. La primera etapa consiste en la detección de movimiento a partir de sustracción de fondo recursiva; el resultado de esta etapa es un mapa binario con los pixeles en movimiento dentro de la secuencia.
2. Posteriormente se emplea la Transformada de Wavelets espacial, la cual consiste en la implementación de la transformada DWT en dos dimensiones (es aplicada tanto al fondo como a la imagen actual); los coeficientes de detalle son utilizados para obtener una imagen compuesta con la información de bordes de la escena. La etapa verifica si existe algún decremento en la intensidad de la imagen compuesta a partir del cuadro actual con respecto a la del fondo.
3. La tercera etapa busca también un decremento, pero en la información de color empleando el espacio de color YUV entre la imagen y el fondo.
4. La siguiente etapa de análisis consiste en la implementación de la Transformada de wavelets temporal, esto es, aplicada a una señal compuesta por los valores de intensidad de un pixel en diferentes cuadros (diferentes momentos de tiempo), de este modo, es posible analizar las fluctuaciones de cada pixel candidato a humo; si estas se encuentran entre 1Hz y 3Hz, se indica presencia de humo.

5. Finalmente se establece un criterio de convexidad definiendo cinco líneas verticales y cinco horizontales sobre la región de interés; si en alguna de las líneas se encuentran al menos tres pixeles pertenecientes al fondo, la convexidad se rompe y los pixeles de la región se descartan.

La Figura 5.2 muestra el diagrama de flujo del algoritmo descrito. El código de la implementación realizada se encuentra en el Anexo 1.

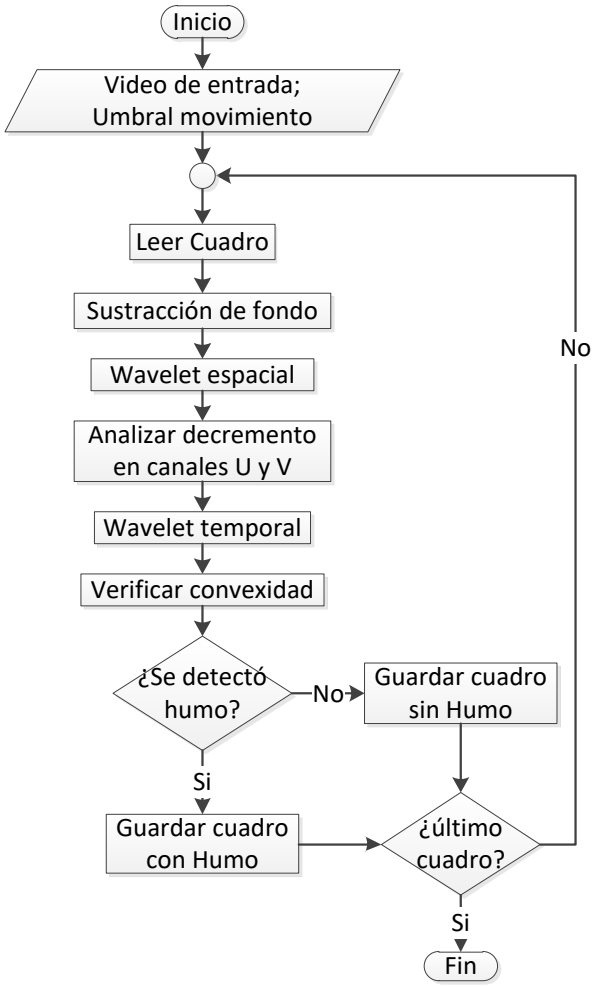

Figura 5.2: Diagrama de flujo del algoritmo propuesto en (Töreyin et al. 2005)

El segundo algoritmo replicado, corresponde al propuesto por (Vieira et al. 2016) consiste en tres etapas de análisis:

1. Al igual que el algoritmo anterior, se emplea la sustracción de fondo para filtrar las regiones de interés de cada imagen.
2. Posteriormente se realiza una comprobación del color en espacio RGB; a partir de una base de entrenamiento con combinaciones de pixeles pertenecientes a humo, se busca la pertenencia a uno de los grupos de pixeles de humo obtenidos para cada pixel verificado.
3. Finalmente, cada cuadro en el que se haya realizado una detección es conservado por tres segundos; si la detección se mantiene durante ese tiempo en la región detectada, el bloque de pixeles es identificado como humo, en caso contrario, la región de la imagen es descartada.

La Figura 5.3 muestra el diagrama de flujo del algoritmo, mientras que el código de la implementación realizada se muestra en el Anexo 2.

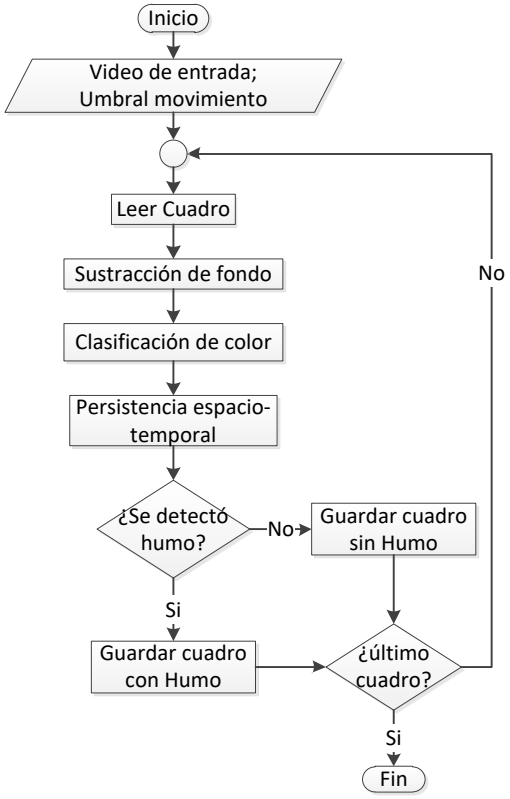

Figura 5.3: Diagrama de flujo del algoritmo propuesto en (Vieira et al. 2016)

Por último, el tercer algoritmo replicado distribuye sus etapas de manera paralela, para posteriormente pasar todos los resultados a AdaBoost, y que esta herramienta sea la encargada de definir si hay o no una detección (Zhao et al. 2015). En general consta de cinco etapas de análisis:

1. La detección de movimiento realizada mediante el filtrado de Kalmann
2. Análisis de consistencia espacio-temporal: Consiste en buscar decrementos entre la energía de alta frecuencia del fondo y de la imagen. Implementada mediante DWT.
3. Análisis de dirección de movimiento o parpadeo: Parte de la segmentación realizada por la detección de movimiento para calcular los centroides de movimiento de cada región de interés, así como el ángulo de movimiento; si dicho ángulo no describe una trayectoria vertical, la región es descartada.
4. Análisis de textura dinámica mediante Patrones Binarios Locales
5. AdaBoost para clasificar la imagen analizada a partir de los clasificadores obtenidos de cada una de las etapas anteriores.

La Figura 5.4 muestra el diagrama de flujo del algoritmo replicado. El código de la implementación se encuentra en el Anexo 3.

Tanto los resultados de la evaluación de desempeño mediante sensibilidad y especificidad obtenidos por los dos experimentos realizados con las réplicas de algoritmos del estado del arte como los resultados de la evaluación de complejidad temporal de los algoritmos se encuentran en la sección 6.1.

Por otro lado, se realizó la evaluación de complejidad temporal de los algoritmos descritos a partir de la medición del tiempo de ejecución empleado para procesar una cantidad determinada de cuadros de entrada; dicha cantidad fue establecida en distintos valores para observar el comportamiento de las herramientas. Los resultados de ésta evaluación son descritos también en la sección 6.1

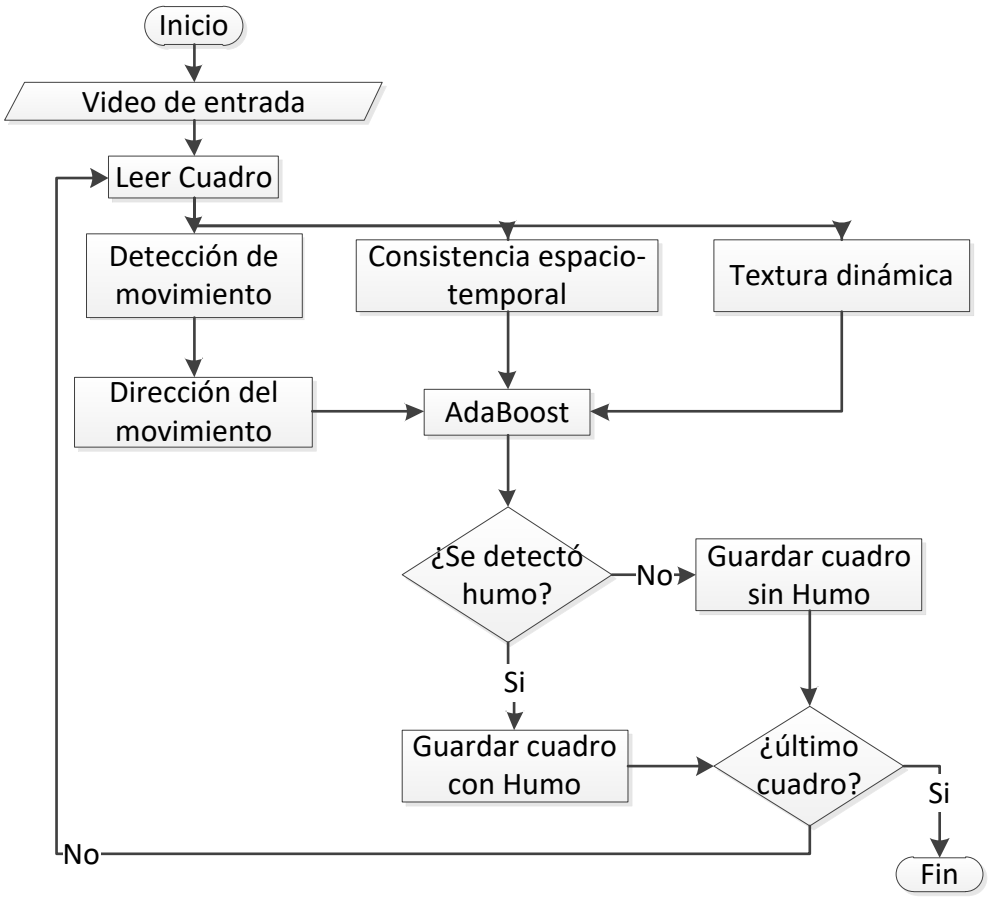

Figura 5.4: Diagrama de flujo del algoritmo propuesto en (Zhao et al. 2015)

5.2 Diseño e implementación del algoritmo propuesto

5.2.1 Funcionamiento general del algoritmo

El algoritmo propuesto consta de distintas etapas; se establece como inicio el ingreso de un video de prueba, que consiste en una escena grabada de un área abierta, en la que hay o no presencia de humo. Posteriormente, se cuenta con una etapa de pre-procesamiento, que se encarga de extraer los cuadros del video de origen, ajustar la resolución de cada imagen y disminuir el efecto de los cambios de iluminación mediante el ajuste estadístico de la imagen utilizando invariantes de color.

La detección de movimiento busca a aquellos pixeles dentro de cada imagen que presentan las variaciones adecuadas para considerar que hay un cambio de posición de un objeto. Esta resulta en un mapa binario que filtra los pixeles para que el resto de las

etapas trabajen solamente con las regiones de interés. Las etapas de análisis de energía wavelet, análisis de trayectoria, de color en RGB y YCbCr, generan clasificadores adicionales al movimiento para que la etapa AdaBoost determine la presencia o no de humo. La Figura 5.5, muestra el diagrama de secuencias del algoritmo propuesto, identificando cada una de las etapas que lo componen y que se explican con detalle a continuación.

5.2.2 Etapa de pre-procesamiento de secuencias de imágenes

La detección de objetos en movimiento es una parte esencial de la visión artificial. En muchos casos como lo es en el humo, los objetos que interesan al sistema de visión no son estáticos, y es esta misma cualidad la que permite diferenciarlos de otros que sí lo son. Para detectar movimiento es necesario utilizar los videos como fuente de información. Un video contiene una secuencia de imágenes (también llamadas cuadros) que pertenecen a una o más escenas.

Por otro lado, las herramientas de tratamiento de imágenes y visión artificial dependen en su mayoría de procesos matemáticos basados en la naturaleza matricial de las imágenes; por tanto, es necesario extraer las imágenes contenidas en los videos fuente para posteriormente analizarlas y realizar las detecciones requeridas.

La herramienta desarrollada para la extracción de cuadros a partir de un video de entrada emplea la información incluida en los metadatos del archivo de video de entrada como su duración y la velocidad de fotogramas (*frame rate* por su traducción al inglés) que representa la cantidad de imágenes que el video contiene por cada segundo de duración.

A partir de estos datos, se calcula el total de cuadros que componen al video mediante la ecuación (5.1).

$$N = d * frate \tag{5.1}$$

Dónde N representa el número total de cuadros de la secuencia; d la duración del video de entrada y $frate$ la velocidad de fotogramas de éste.

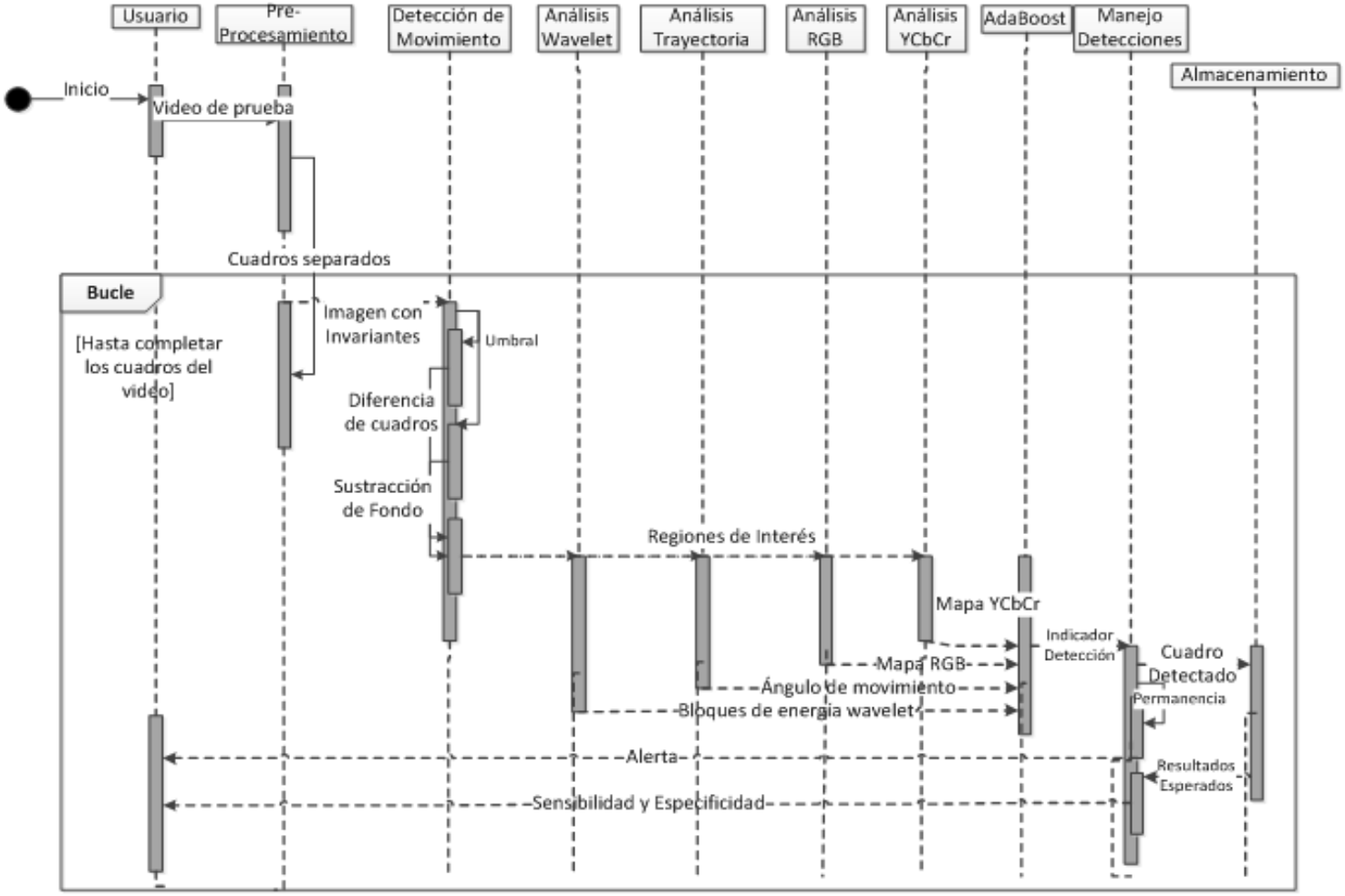

Figura 5.5: Diagrama de secuencias de la herramienta propuesta

Una estructura cíclica es dedicada a la lectura de cuadros y su almacenamiento bajo el nombre de imagen_i.png, donde i es la posición que el cuadro ocupa en la secuencia (entre 1 y el total de imágenes N), en una carpeta específica para el video. La Figura 5.6 muestra el diagrama de flujo de la herramienta descrita.

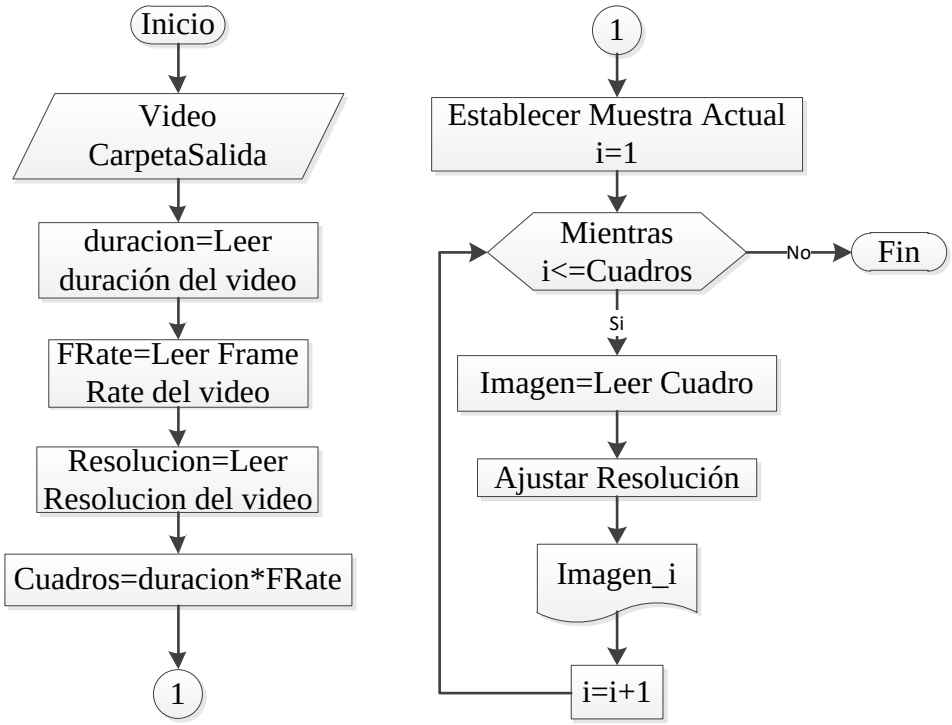

Figura 5.6: Diagrama de flujo de la etapa desarrollada para la extracción de cuadros a partir de un video de entrada.

Una vez que se han extraído los cuadros, la siguiente etapa extrae la información de duración y velocidad del video a partir de los metadatos del archivo de video ingresado. A partir de los metadatos, también es posible conocer la resolución del archivo de video; ésta es expresada mediante un par de números multiplicados entre sí, de manera que si se tiene un valor de resolución igual a 1280×720 indica que el ancho de la imagen está formado por 1280 pixeles, mientras el alto de la imagen contiene 720. Una mayor resolución implica más pixeles en la imagen y una mejor calidad en ésta; sin embargo, también conlleva una mayor cantidad de pixeles y para cada herramienta implica realizar más operaciones computacionales para comprobar todos los pixeles. Por tanto, es necesario reducir la cantidad total de pixeles en las imágenes mediante el proceso

conocido como redimensionamiento de imágenes que contempla una serie de alternativas para incrementar o reducir el tamaño de una imagen. La reducción implica un re-muestreo de la imagen, eliminando parte de la información contenida en las matrices, pero conservando los patrones originales muestreados durante la captura (Heras 2017). Considerando la resolución de los videos obtenidos a partir del banco de pruebas consultado en (Cetin 2003), se determinó estandarizar la resolución de los videos a 320×240, manteniendo una proporción rectangular de 4 a 3, esto es, cuatro pixeles de ancho por cada tres de alto.

La Figura 5.7 presenta una muestra del redimensionamiento de imágenes mediante la herramienta "imresize" del software MATLAB®, que realiza el re-muestreo de la imagen indicada bajo las dimensiones de salida especificadas.

Figura 5.7: Muestra del proceso de redimensionamiento de imágenes

Finalmente, la etapa de pre-procesamiento es empleada para afrontar los cambios de iluminación por medio de herramientas de constancia de color. El algoritmo propuesto combina dos técnicas distintas.

1. La técnica de parche blanco (Kaur & Sharma 2016; Francis 2016) (White patching por su traducción al inglés), con la que se busca tratar los cuadros con poca iluminación.

El cálculo del parche a aplicar comprende la detección del valor de intensidad máximo para cada canal de color, esto es:

$$I_c = MAX(F_c(x, y)) \tag{5.2}$$

Dónde I_c es el iluminante que corresponde al canal de color c; $F_c(x, y)$ se refiere al pixel del cuadro F ubicado en la posición (x, y) de la matriz del canal de color c; los canales de color utilizados corresponden al espacio RGB, por lo que $c = \{R, G, B\}$.

Con base en los iluminantes computados, las intensidades de los pixeles son re-escaladas como se describe en la ecuación (5.3).

$$F'_c(x, y) = \frac{F_c(x, y)}{MAX(I_c)} \tag{5.3}$$

2. Como complemento de la herramienta anterior, los cuadros con mayor iluminación son tratados mediante el cálculo de un iluminante canónico por el método del Mundo Gris, que establece que el color en una escena o cuadro es acromático y por tanto el valor medio de iluminación en una misma escena es constante; cualquier desviación de la media es considerada como un cambio de iluminación (Tek et al. 2016).

La herramienta calcula el valor medio de intensidad en el cuadro inicial para establecerlo como el valor constante gris. Para cuadros posteriores, los cambios en la intensidad media son detectados comparando contra el mencionado valor medio y posteriormente la diferencia es sustraída de los valores de intensidad de la imagen

El código correspondiente a la etapa de pre-procesamiento se encuentra en el Anexo 4. En éste, se muestra la codificación correspondiente a las tareas descritas, así como la condición evaluada por el algoritmo para asignar el proceso de parche blanco o el de mundo gris a un cuadro determinado.

5.2.3 Etapa de detección de movimiento.

Cuando se busca detectar un objeto que no se encuentra estático dentro de una escena, analizar las regiones que varían de un cuadro a otro y la forma en que lo hacen permite ubicar las regiones de interés que deben ser procesadas como candidatas. Dado que el humo se encuentra en movimiento, es necesario detectar las variaciones entre distintos cuadros de la secuencia analizada.

En (Vieira et al. 2016; Günay 2015; Cetin et al. 2016), se propone el uso de un algoritmo hibrido para detección de movimiento que consiste en una combinación de las técnicas de diferencia de cuadros y substracción de fondo. Para el monitoreo de una misma escena durante un tiempo determinado éste algoritmo ofrece una mejor adaptabilidad ante la presencia de nuevos objetos, pues estima de manera iterativa la matriz del fondo de la imagen.

La diferencia de cuadros consiste en obtener la matriz de valores absolutos de las diferencias que hay entre dos o más cuadros de una secuencia. Considerando a F_i como el cuadro actual y a F_{i-1} el cuadro anterior al actual, la ecuación (5.4) describe la diferencia entre ambos cuadros.

$$Diferencia = |F_i - F_{i-1}| \tag{5.4}$$

La principal desventaja de esta técnica es que al trabajar bajo condiciones de iluminación cambiantes, cualquier variación de intensidad debido a la iluminación puede ser interpretada como movimiento. La técnica conocida como diferencia de tres cuadros (*Three Frame Differencing*, TFD, por su traducción al inglés) es una variante de esta técnica con mayor tolerancia a estas alteraciones. Consiste en obtener también la diferencia entre F_i y F_{i-2} y contrastar con un valor de umbral ambas diferencias para descartar todos aquellos pixeles cuyas variaciones no mantengan cierta intensidad.

La ecuación (5.5) muestra la manera de calcular un mapa binario TFD en el que los pixeles que cumplan con la condición de diferencia entre el cuadro actual y los dos anteriores, esto es que las diferencias superen al valor de umbral T, son identificados con un 1 mientras que cualquiera que no la cumpla es identificado con un valor 0 (Genovese et al. 2011).

$$TFD = \begin{cases} 1 & |F_i - F_{i-1}| > T \;\&\; |F_i - F_{i-2}| > T \\ 0 & Cualquier\ otro\ caso \end{cases} \tag{5.5}$$

El algoritmo híbrido complementa el resultado de la diferencia de tres cuadros con la sustracción de fondo. Una nueva matriz B es calculada iterativamente como el fondo de los cuadros analizados. Cada cuadro actualiza el fondo, de modo que los pixeles que son detectado con movimiento se mantienen sin cambios en la matriz de fondo (pues

pertenecen al primer plano) mientras que los que no tienen movimiento son actualizados como:

$$B_{i+1} = \begin{cases} \alpha B_i(x,y) + (1-\alpha)F_i(x,y) & sin\ movimiento \\ B_i(x,y) & con\ movimiento \end{cases} \quad (5.6)$$

Donde B_i y B_{i+1} representan la matriz de fondo correspondiente al cuadro actual F_i y al cuadro siguiente F_{i+1} respectivamente; (x, y) son las coordenadas del pixel analizado dentro de las matrices del cuadro y su fondo; y α es un parámetro entre 0 y 1 que define la influencia que tiene el cuadro en la actualización del fondo.

De manera similar a la diferencia de cuadros descrita en la ecuación (5), la sustracción de fondo requiere obtener la diferencia entre el cuadro actual y el fondo; si esta diferencia supera al valor de umbral T, el pixel analizado tiene movimiento. La ecuación (5.7) describe la identificación de movimiento mediante sustracción de fondo.

$$SF = \begin{cases} 1 & |F_i - B_i| > T \\ 0 & |F_i - B_i| \leq T \end{cases} \quad (5.7)$$

Donde SF describe un mapa binario resultante; F_i es el cuadro actual; B_i es el fondo actual y T el valor de umbral. Si la diferencia supera al valor de umbral, es decir, si se cumple la condición para considerar que el pixel tiene movimiento, se asigna al pixel un valor de 1 en el mapa binario resultante; se asigna un 0 en caso contrario.

Los mapas binarios de la diferencia de tres cuadros y la sustracción de fondo son comparados, y en aquellos casos donde ambas técnicas hallan asignado un 1 a la posición del pixel, se indica que el pixel en cuestión tiene movimiento.

Un aspecto a considerar es la definición apropiada del valor de umbral T, pues su correcta definición permitirá el funcionamiento óptimo del algoritmo. Cada cuadro en la secuencia está formado por diferentes valores de intensidad que varían de un objeto a otro. El método de Otsu (Vala & Baxi 2013; Jinlan et al. 2016) plantea buscar a través de estadísticas de estos valores de intensidad el umbral (*threshold*, por su traducción al inglés) que maximice las varianzas entre clases para una imagen segmentada; esto es calcular un valor a partir del cual se puedan separar las distintas regiones de una imagen (fondo y primer plano).

Para el cálculo del umbral, el cuadro actual es convertido del espacio de color RGB a escala de grises, de modo que las variaciones de intensidad presentes en el cuadro se encuentran en el rango de 0 a 255. La suma de los pixeles del cuadro F_j es N, mientras que la suma de los pixeles correspondientes a cada nivel de intensidad i es n_i. La probabilidad de encontrar cada valor de intensidad en el cuadro es:

$$p_i = \frac{n_i}{N} \tag{5.8}$$

Los niveles de intensidad son separados en dos clases a partir de un pivote de modo que la primera clase C_1 comprenderá los valores del rango $[0, th]$, donde th es el valor pivote; la segunda clase C_2 contendrá por tanto los valores del rango $[th, 255]$. La suma de los valores de probabilidad de cada clase y la media son calculadas como:

$$s_1 = \sum_{x=0}^{th} p_i \tag{5.9}$$

$$s_2 = \sum_{x=th}^{255} p_i \tag{5.10}$$

$$m_1 = \frac{\sum_{x=0}^{th} ip_i}{s_1} \tag{5.11}$$

$$m_2 = \frac{\sum_{x=th}^{255} ip_i}{s_2} \tag{5.12}$$

Donde s_1 y s_2 son las sumas de los valores de probabilidad de las clases 1 y 2 respectivamente; m_1 y m_2 son los valores medios ponderados de las clases 1 y 2 respectivamente. Posteriormente la varianza del cuadro es calculada como:

$$\sigma^2 = s_1 s_2 (m_1 - m_2)^2 \tag{5.13}$$

Este cálculo de varianza es realizado 255 veces, modificando el valor del pivote entre 0 y 255; cuando la varianza alcanza su valor máximo, el valor pivote es asignado como el valor de umbral T. El código correspondiente a la etapa de detección de movimiento se encuentra en el Anexo 5.

5.2.4 Etapa de análisis en espacio Wavelets

La siguiente etapa consiste en analizar la información de frecuencia en dominio de Wavelets. Se propone examinar la diferencia entre la energía de los bloques del fondo y la imagen mediante un proceso de comparación con un valor de umbral, esto es (Töreyin et al. 2005):

$$bloque(b_1, b_2) = humo \leftrightarrow EB_i(b_1, b_2) - E_i(b_1, b_2) > Tw \quad (5.14)$$

Dónde EB_i es la energía del bloque (b_1, b_2) en la imagen compuesta del fondo; E_i es la energía del mismo bloque en la imagen compuesta del cuadro actual; Tw es el valor de umbral obtenido aplicando el método de Otsu en la imagen compuesta de cuadro actual. Si la condición se cumple, un mapa binario de salida identifica el bloque adecuado con un valor de 1; en caso contrario, tanto los bloques que no cumplen la condición como aquellos descartados por no pertenecer a la región de interés, son identificados con un valor 0. La Figura 5.8 muestra el diagrama de flujo de la etapa de análisis de energía wavelet.

El Anexo 6 contiene el código correspondiente a la etapa de análisis en dominio de Wavelets; en éste se muestra la configuración de la herramienta como la Wavelet empleada, la metodología utilizada para la división del cuadro analizado en bloques y su posterior comparación con los resultados obtenidos en la matriz de fondo correspondiente. Así mismo, el mecanismo de comparación con un valor de umbral para delimitar la presencia de humo en la imagen y reducir el margen de error.

5.2.5 Etapa de detección de la dirección del movimiento

La temperatura del humo genera una característica dinámica que permite diferenciar al humo de objetos similares como las nubes o la niebla. El humo generalmente tiene una temperatura mayor a la del aire que se encuentra alrededor del incendio. Esta diferencia de temperaturas provoca que el movimiento del humo tome una trayectoria vertical, generalmente en forma de columna, que se mantendrá hasta que la temperatura del aire iguale a la del humo. En ese punto, el humo detiene su movimiento vertical y su dispersión aumenta.

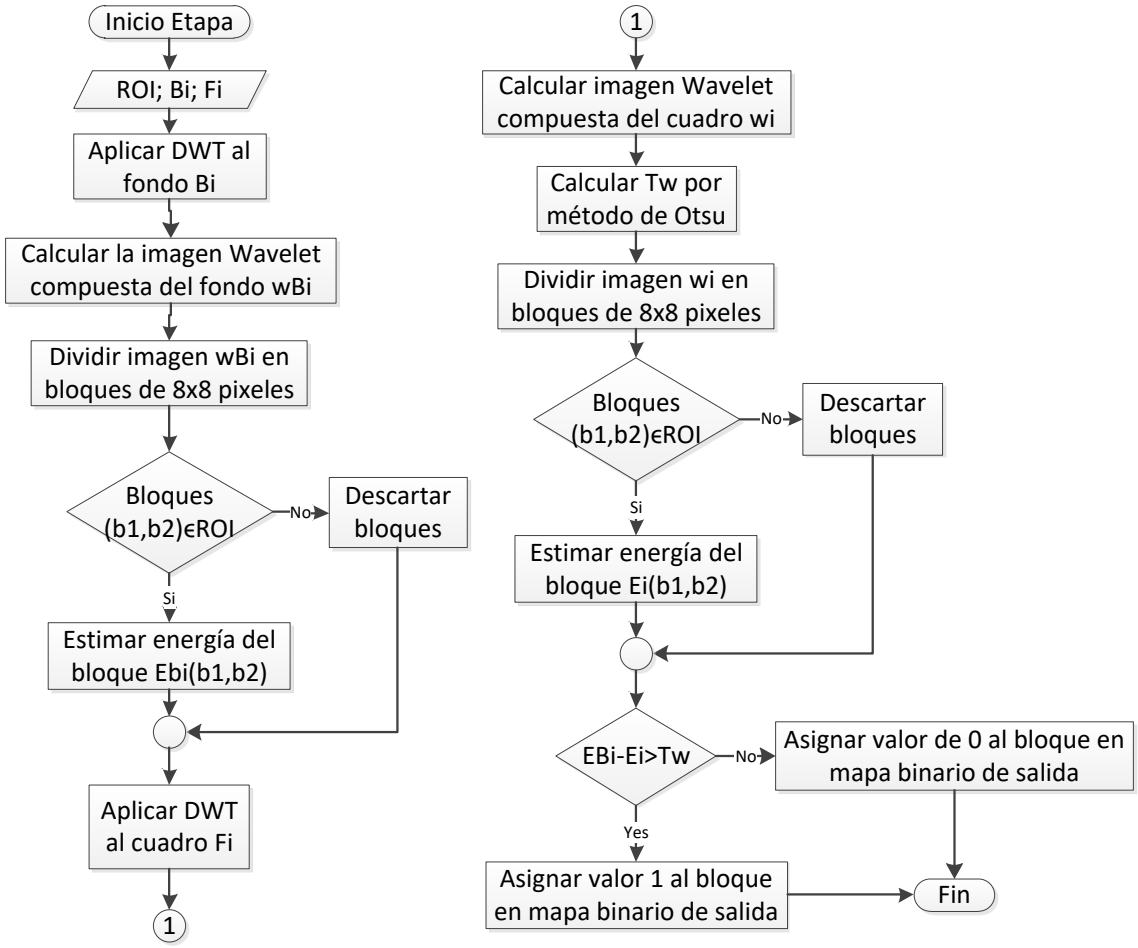

Figura 5.8: Diagrama de flujo de la etapa de análisis de energía Wavelets.

Por tanto, identificar la dirección en la que se mueve el humo permite diferenciarlo de otras regiones con características similares. Para determinar esto es necesario primero identificar una región de interés, formada por pixeles con movimiento. El mapa binario resultante de la etapa de detección de movimiento es utilizado para cubrir este requisito.

Posteriormente, el mapa binario es analizado en busca de pixeles con movimiento; cuando uno es encontrado, se define un bloque de interés a partir del pixel encontrado (punto (0,0)); los límites vertical y horizontal se definen posteriormente analizado las líneas verticales correspondientes a la fila y columna en las que está ubicado el pixel inicial. El límite horizontal (punto (1,0)) es el primer pixel con valor 0 en el mapa

binario que interrumpa la secuencia de pixeles con movimiento sucesivos a partir del pixel inicial. De la misma forma, el límite vertical (punto $(0,1)$) es primer pixel que interrumpa la secuencia de pixeles sucesivos con movimiento a partir del pixel inicial.

El ángulo de la dirección del movimiento es calculado mediante el proceso descrito en las ecuaciones (4.15)-(4.19) del estado del arte. Si el ángulo calculado es mayor a cero se considera que el bloque tiene un movimiento que tiende a lo vertical; En caso de que dos o más regiones sean adyacentes de manera vertical u horizontal, la suma de los ángulos calculados para cada región es considerada para definir la dirección real del conjunto de regiones.

La Figura 5.9 muestra el diagrama de flujo que describe el funcionamiento de la etapa de detección de la dirección de movimiento basado en el cálculo del ángulo de movimiento a partir de los centroides de movimiento de cada región de interés.

En el Anexo 7 se muestra el código correspondiente a la etapa de detección de dirección de movimiento, donde se puede observar la codificación del cálculo de momentos de inercia para el área analizada como parte de la región de interés.

5.2.6 Etapa de análisis de color

Un valor de umbral es necesario para definir el límite de diferencia entre los tres canales. A partir de un grupo de imágenes de entrenamiento obtenidas a partir de video de prueba con presencia de humo, fue posible estimar un valor de umbral para la comparación de colores en RGB. La Figura 5.10, muestra el diagrama de flujo para la estimación del valor de umbral.

Los valores de intensidad de las matrices R, G y B correspondientes a los pixeles con movimiento son contrastados con el umbral estimado y un nuevo mapa binario es generado como salida de la etapa de acuerdo con la ecuación (5.15).

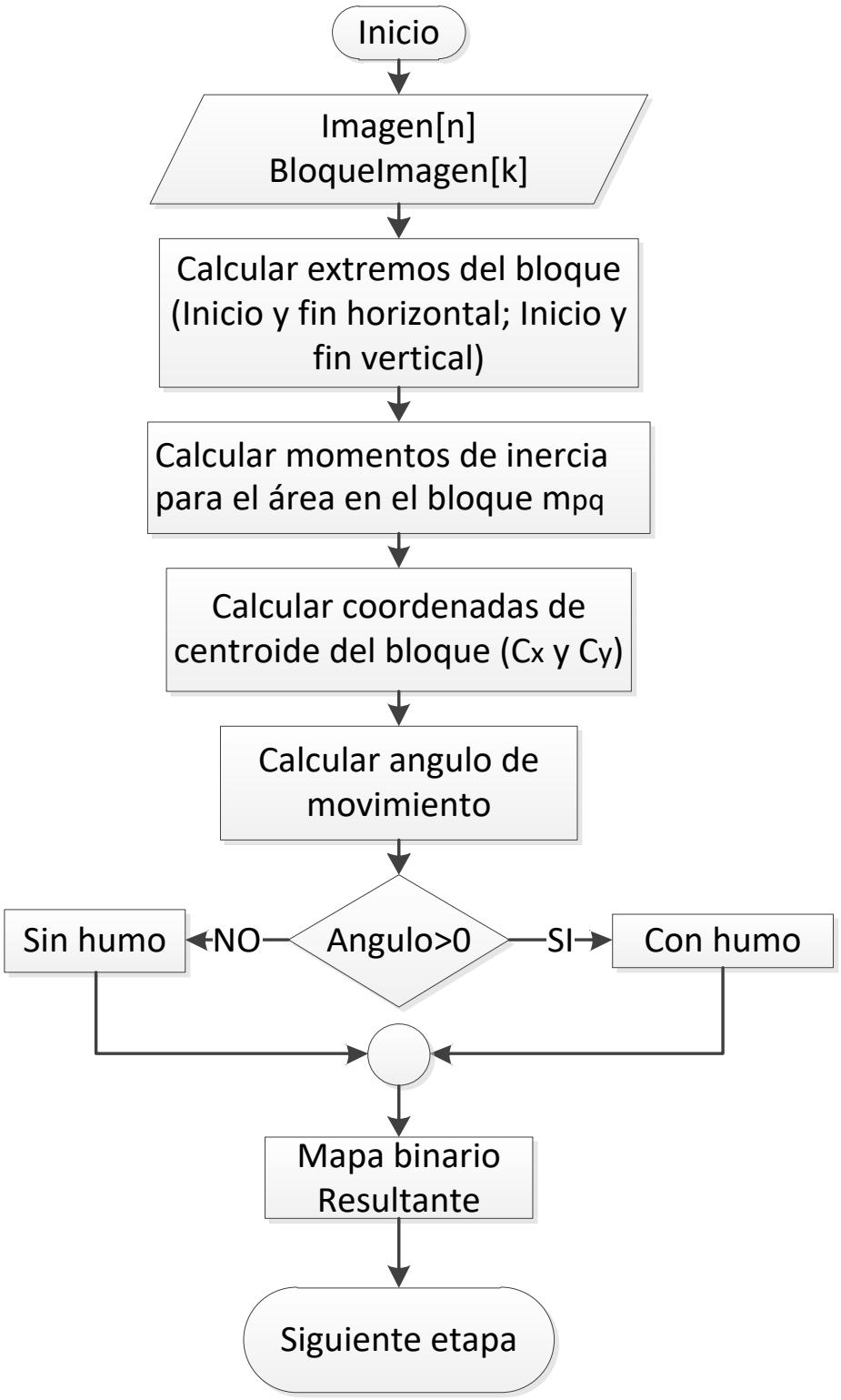

Figura 5.9: Diagrama de flujo correspondiente a la etapa de detección de la dirección del movimiento

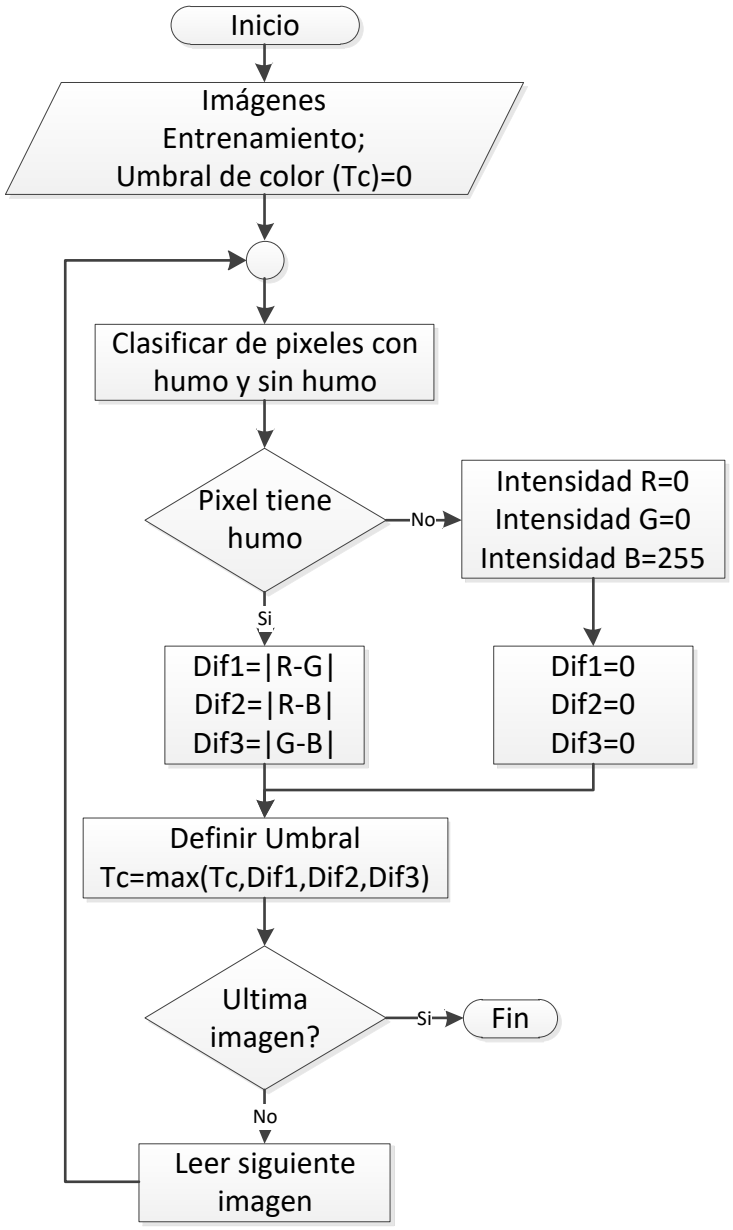

Figura 5.10: Diagrama de flujo para la estimación del umbral de análisis RGB

$$MBC1(x,y) = \begin{cases} 1 & |R(x,y)-G(x,y)| \le T_{RGB} \;\&|R(x,y)-B(x,y)| \le T_{RGB} \&|G(x,y)-B(x,y)| \le T_{RGB} \\ 0 & cualquier\ otro\ caso \end{cases} \quad (5.15)$$

Donde $MBC1$ representa el mapa binario resultante; T_{RGB} es el umbral estimado para la comparación de las matrices RGB; R, G y B son las intensidades correspondientes al pixel (x, y) de las matrices R, G y B respectivamente.

Mediante experimentación, es posible estimar los rangos de evaluación para cada canal de crominancia y luminancia, de forma similar a la estimación del umbral para RGB, un conjunto de imágenes de entrenamiento es utilizado para obtener y actualizar las diferencias entre los tres canales. En este caso se establecen tres umbrales: El umbral T_{YCb} que es utilizado para evaluar la diferencia entre los canales Y y Cb; El umbral T_{YCr} para evaluar la diferencia entre los canales Y y Cr; y el umbral T_{CbCr} utilizado para evaluar la diferencia entre los canales de crominancia.

Un nuevo mapa binario es definido con los resultados de la comparación de todos los pixeles en los que se detectó movimiento. Aquellos que cumplen las condiciones de diferencia para los tres umbrales son indicados con un valor 1, y los que no con un 0.

El código correspondiente a la etapa de análisis de color se encuentra en el Anexo 8, en donde se puede identificar la codificación empleada para realizar las distintas comparaciones de los pixeles analizados con los valores de umbral obtenidos a través del entrenamiento realizado.

5.2.7 Etapa de clasificación mediante el algoritmo AdaBoost

El algoritmo AdaBoost es una herramienta de aprendizaje ampliamente aplicada en el procesamiento de imágenes. Consiste en un procesamiento para entrenar lo que denominan como clasificadores débiles y obtener clasificadores fuertes a partir de estos (Wang 2012).

Inicialmente se requiere de un conjunto de imágenes de prueba donde mp es el total de muestras positivas (con humo) y mn el total de muestras negativas. Cada imagen de muestra será etiquetada formando un par $(imagen, muestra)$, donde el valor $muestra$ será igual a 1 si la imagen contiene humo y 0 si no. Posteriormente, se calcula el peso inicial de cada clasificador W_j como:

$$W_j = \begin{cases} \frac{1}{mp}, & muestra_i = 1 \\ \frac{1}{mn}, & muestra_i = -1 \end{cases} \tag{5.16}$$

Entonces es necesario procesar la imagen de entrenamiento EF_i con las etapas de detección anteriormente descritas. Los resultados de cada etapa son contrastados con la etiqueta y_i de la imagen; si el valor de $muestra_i$ es 1, se espera que encada etapa los resultados sean 1, si el valor de $muestra_i$ es -1, se espera que los resultados de las etapas sean 0. Si el valor de la etiqueta y el resultado de alguna etapa no coinciden, se establece que la diferencia d_j donde j lista los cinco clasificadores, es igual a 1; en caso contrario la diferencia es 0. El error del algoritmo es calculado en base a las diferencias de los clasificadores como se describe en la ecuación (5.17)

$$error_j = \frac{W_j * d_j}{\sum_j W_j} \tag{5.17}$$

Posteriormente se calcula el grado de actualización para cada clasificador como:

$$a_j = \ln\left(\frac{1 - error_j}{error_j}\right) \tag{5.18}$$

Donde a_j es el grado de actualización para cada clasificador j.

Para concluir la etapa de entrenamiento, los pesos calculados inicialmente son actualizados mediante la operación indicada en la ecuación (5.19):

$$W_j = W_j(i-1)\begin{cases} e^{-a_j}, & d_j = 0 \\ e^{a_j}, & d_j = 1 \end{cases} \tag{5.19}$$

Donde $W_j(i-1)$ se refiere al peso calculado para el clasificador hasta la imagen de entrenamiento anterior a la actual. El proceso se repite para tantas imágenes como sean necesarias para concluir el entrenamiento. La Figura 5.11 muestra el diagrama de flujo del entrenamiento de la herramienta AdaBoost.

La etapa de ejecución de AdaBoost parte del procesamiento de una imagen con las etapas de detección para obtener los clasificadores de dicha imagen. Una vez que todos los clasificadores débiles son obtenidos, se calcula el clasificador fuerte como:

$$SC_i = \sum_j clasificador_j * W_j \tag{5.20}$$

Donde SC_i es el clasificador fuerte del cuadro i. Si el valor del clasificador fuerte es mayor a 0, se clasifica a la imagen como con humo; si el valor es 0, se clasifica como

imagen sin humo. La Figura 5.12 muestra el diagrama de flujo de la ejecución de AdaBoost.

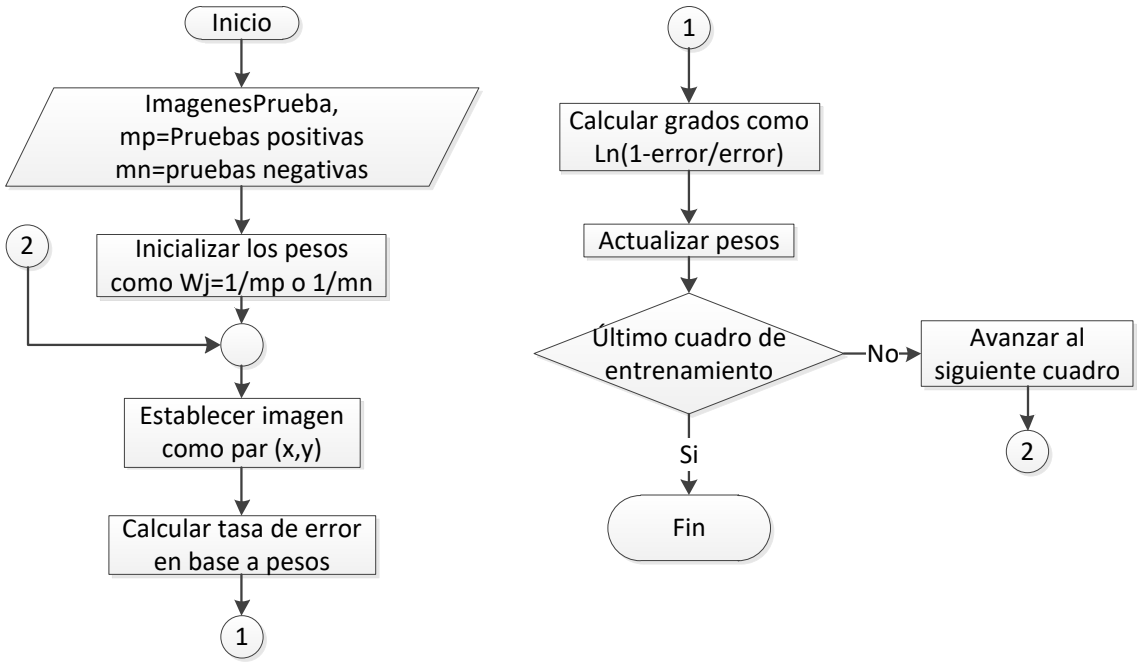

Figura 5.11: Diagrama de flujo del entrenamiento de AdaBoost

El Anexo 9 muestra el código correspondiente a la etapa de toma de decisiones mediante el algoritmo AdaBoost; en éste se muestra la etapa de entrenamiento para la actualización de los pesos de cada clasificador débil a partir de un video de entrada, así como la etapa de ejecución que toma los pesos almacenados por el entrenamiento para realizar la clasificación del cuadro en una de las dos clases disponibles (con humo o sin humo).

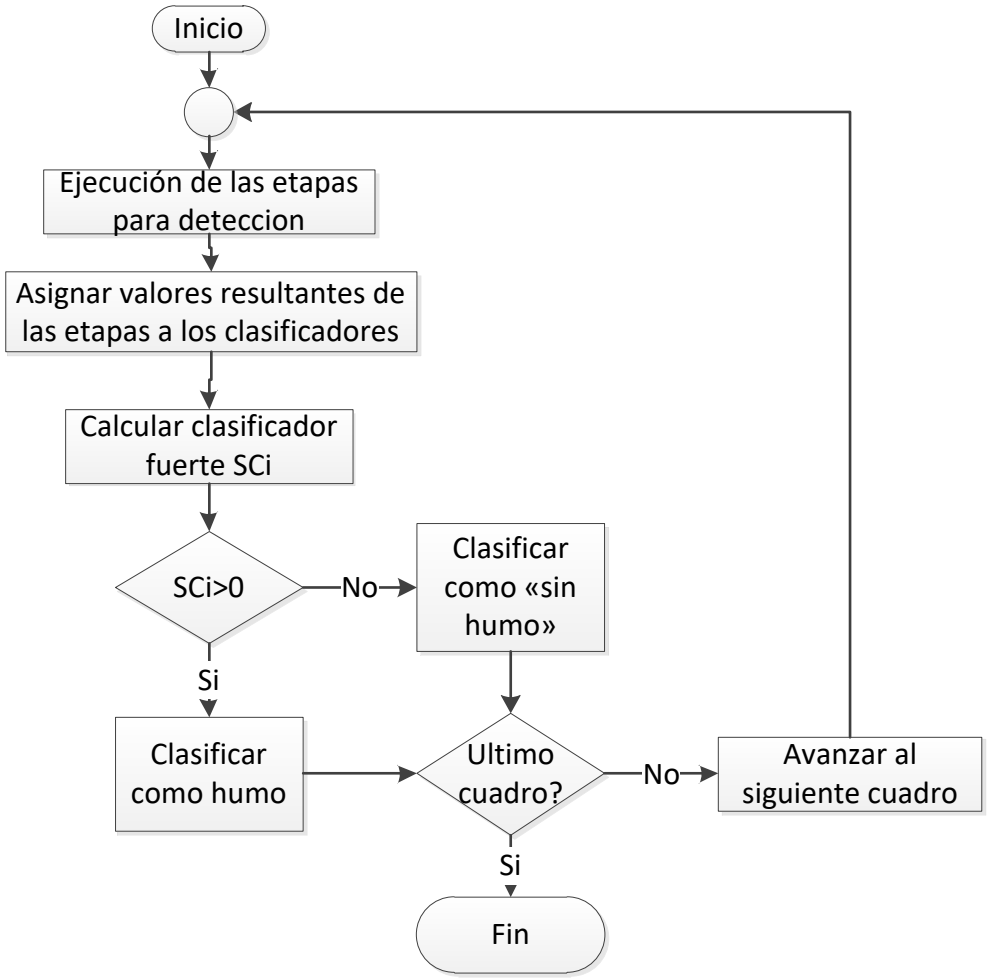

Figura 5.12: Diagrama de flujo de la ejecución de AdaBoost

5.3 Experimentación

Los cuadros reservados para la ejecución de pruebas de validación son etiquetados mediante la formación de pares $(imagen, muestra)$. Todas las muestras positivas son almacenadas en un directorio determinado, mientras que las muestras negativas son almacenadas en otro. La Figura 5.13 muestra el diagrama de flujo del etiquetado de los cuadros extraídos de los videos de prueba.

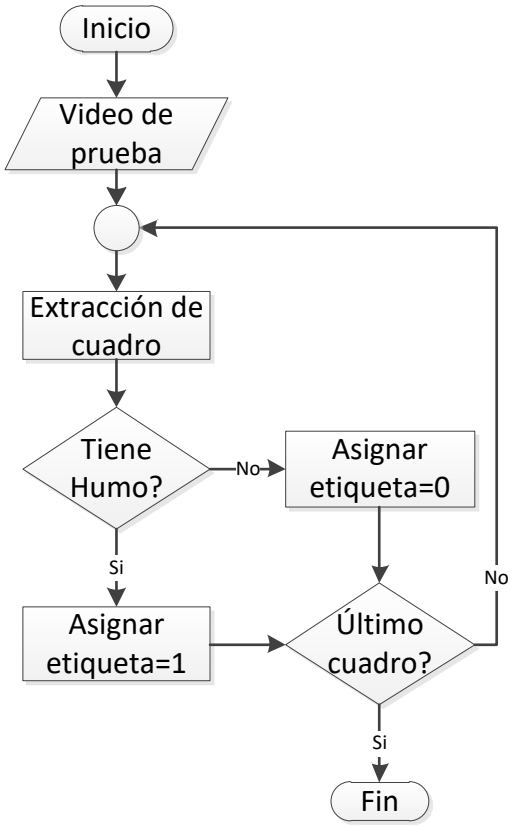

Figura 5.13: Diagrama de flujo del etiquetado previo a las pruebas del algoritmo

5.3.1 Casos de prueba.

Considerando los tipos de iluminación con los que se requiere que trabaje la herramienta descrita, se detectaron seis tipos de videos de prueba, diferenciados por cada uno de los estados de iluminación y la presencia o no de humo. Es decir, los videos de prueba son clasificados en 3 categorías: Luz diurna, Día nublado y Luz Nocturna. En cada categoría, se dividen los cuadros en imágenes con humo e imágenes sin humo. La tabla 5.2 presenta la clasificación y una muestra representativa de cada categoría.

Tabla 5.2: Tipos de videos de prueba para el algoritmo

Tipo de videos de prueba	Categoría	Sub-Categoría	Cuadro de Muestra
1	Luz Diurna	Con Humo	
2	Luz Diurna	Sin Humo	
3	Día Nublado	Con Humo	

4	Día Nublado	Sin Humo	
5	Luz Nocturna	Con Humo	
6	Luz Nocturna	Sin Humo	

5.3.2 Pruebas preliminares

Como punto de control, el funcionamiento de cada una de las etapas de detección debe ser verificado; los mapas binarios resultantes de cada herramienta son comprobados por dos métodos:

1. Se realiza una comprobación de exactitud: A partir del etiquetado previo a la ejecución de una prueba, se determina el número de cuadros con presencia de humo y el número de cuadros sin éste.

El mapa binario de cada etapa es almacenado en un directorio en caso de que éste contenga una región de interés. En caso contrario, se almacena en un directorio distinto. Los valores de sensibilidad y especificidad son calculados a partir de las ecuaciones (5.21) y (5.22), descritas en la sección 5.4.3.

El valor de sensibilidad para cada caso de prueba no debe ser menor al 80%; en caso contrario, la etapa evaluada debe ser ajustada modificando los parámetros de trabajo correspondientes a la etapa en cuestión para mejorar el resultado. La especificidad, del mismo modo, no debe ser menor al 85%.

2. El segundo proceso de validación preliminar, consiste en el muestreo de cuadros resultantes de la etapa (Mapas binarios). Cada uno de los cuadros seleccionados de manera aleatoria es validado tomando el cuadro originalmente analizado.

El mapa binario a evaluar es utilizado como mascara para filtrar los pixeles del cuadro original correspondiente. Una validación manual es realizada para verificar que las regiones de interés descritas por la máscara correspondan a los bloques de humo dentro de la imagen.

Los resultados correspondientes a las pruebas preliminares realizadas para cada etapa se muestran en la sección 6.2.

5.3.3 Evaluación de exactitud

Contemplando todas las etapas de detección, los resultados obtenidos por la herramienta propuesta a través de AdaBoost son almacenados en una carpeta específica, para posteriormente ser recuperados y contados. A partir de los conteos realizados por cada una de las etiquetas permiten obtener los totales de verdaderos positivos (VP), falsos

positivos (FP), verdaderos negativos (VN) y falsos negativos (FN). Utilizando estos valores, los índices apara evaluación del algoritmo son obtenidos como:

$$Sensibilidad = \frac{VP}{VP + FN} \tag{5.21}$$

$$Especificidad = \frac{VN}{VN + FP} \tag{5.22}$$

El proceso se repite para cada uno de los casos de prueba descritos anteriormente para comprobar la tolerancia de la herramienta propuesta a diferentes estados de iluminación.

Adicionalmente se realizan pruebas con distintas definiciones de los parámetros de ejecución como el umbral de movimiento, el de energía wavelet y el parámetro de actualización de fondo. Estos resultados, se muestran en la sección 6.3 de este documento.

5.3.4 Comparativa de resultados con el Estado del Arte

A partir de la metodología de prueba empleada con la réplica de algoritmos del estado del arte descrita en la sección 5.2 (Nieto-González et al. 2017), tanto los algoritmos replicados como el algoritmo propuesto son evaluados para los casos de prueba definidos en la sección 5.4.1. La exactitud de los algoritmos se contrasta mediante la comparación de la sensibilidad y especificidad obtenidas.

Posteriormente, se evalúa la complejidad del algoritmo propuesto, a partir de los tiempos de ejecución obtenidos para distintos valores de duración en los videos de entrada. De modo que se establece una tabulación entre la cantidad de cuadros de entrada y el tiempo de ejecución en segundos. Los valores tabulados son graficados para observar el comportamiento del algoritmo.

Finalmente, a partir de la función observada, se determina el grado de complejidad del algoritmo propuesto. Este grado es comparado con el obtenido para cada herramienta replicada del estado del arte.

Los resultados de las comparaciones descritas, se muestran en la sección 6.4.

5.4 Interfaz gráfica

El algoritmo propuesto es implementado en una interfaz gráfica de usuario desarrollada para permitir la validación de los resultados de cada etapa, agregar un manejo de alertas que para iniciar y detener la alarma de detecciones, y permitir la manipulación de los resultados de las pruebas.

A las etapas descritas, se añade la interacción con el usuario, quien tiene la posibilidad de elegir un video de entrada, especificar si éste es para entrenamiento o para prueba, iniciar y detener la ejecución del algoritmo, almacenar los mapas binarios de cada etapa de análisis y recibir la información correspondiente a los índices de exactitud, las detecciones y alarmas generadas por la herramienta. La Figura 5.14 muestra el diagrama de uso de la interfaz desarrollada.

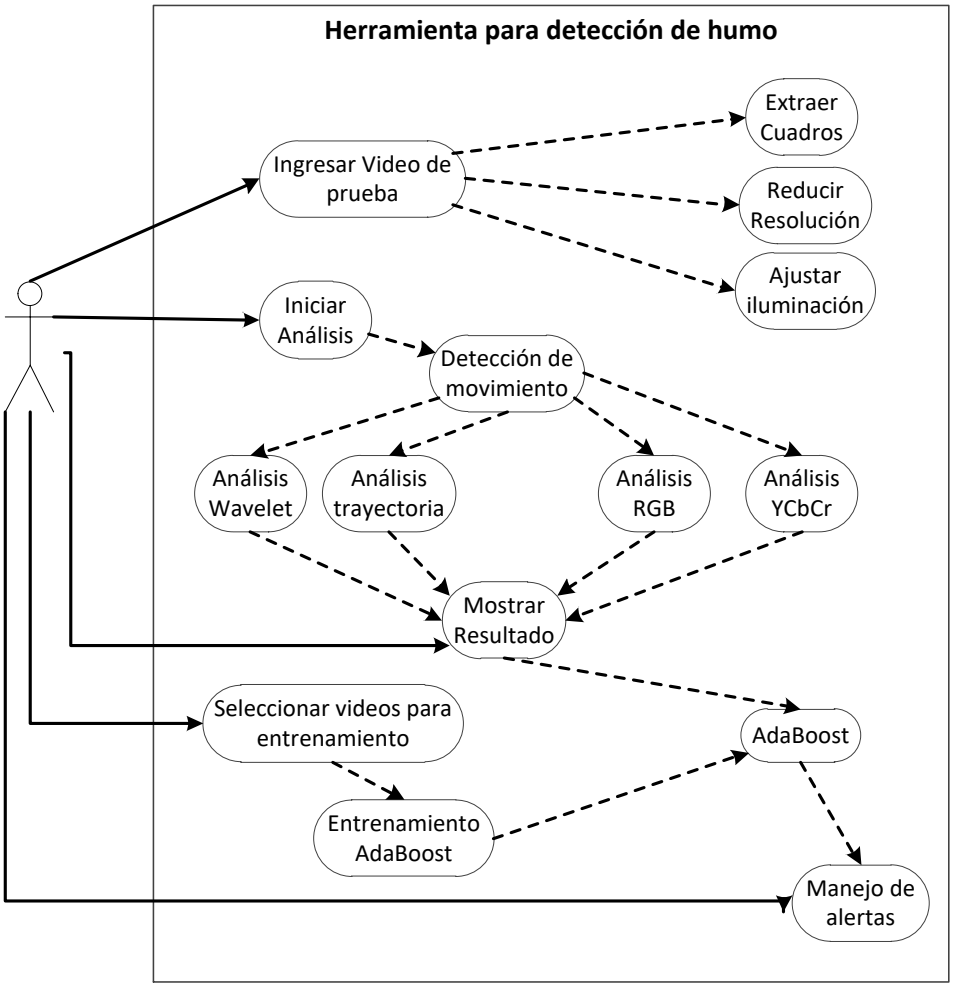

Figura 5.14: Diagrama de casos de uso para de la interfaz gráfica desarrollada

La interfaz consta de dos partes: la interfaz de entrada que muestra las opciones necesarias para que el usuario seleccione e ingrese los videos, así como el botón de inicio para empezar el procesamiento del video de entrada; y la interfaz de salida, que contiene los mapas binarios de las distintas etapas de detección, las opciones necesarias para que el usuario elija la etapa a mostrar y los resultados obtenidos.

La Figura 5.15 muestra el diseño de la ventana para la interfaz gráfica; el lado izquierdo de la ventana, contiene las opciones de manejo para la interfaz de entrada, mientras que los controles de la derecha corresponden a la interfaz de salida.

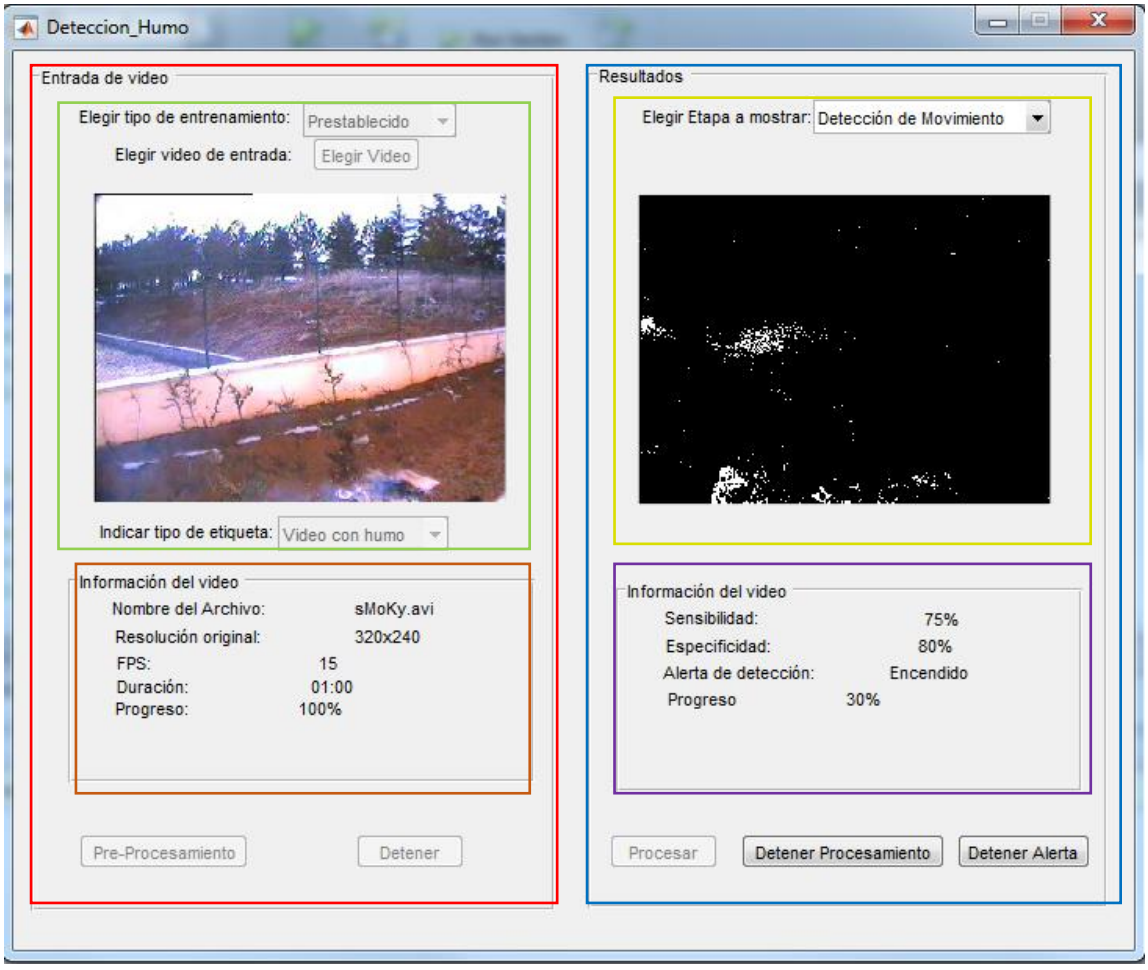

Figura 5.15: Estructura gráfica desarrollada para la interfaz de usuario.

La interfaz está dividida en dos paneles; el panel izquierdo (marcado en rojo dentro de la figura), incluye las opciones de entrada para videos de prueba. Un menú desplegable es empleado para seleccionar el tipo de entrenamiento, si la opción es “Preestablecido”, se utilizará el vector de pesos obtenido a través de los ejercicios de prueba descritos en el siguiente capítulo; mientras que seleccionar la opción “Nuevo”, implicará que el video se tomará como parte de un nuevo entrenamiento para la herramienta. Otro menú desplegable es utilizado para indicar si el video contiene humo o no, para indicar la etiqueta con la que se identificarán los cuadros extraídos. Estos menús, junto con un área de gráfico son identificados en la figura en color verde.

El sub-panel de datos identificado por el color naranja en la figura 5.5, muestra la información obtenida a partir de los metadatos del archivo de video seleccionado, así como el porcentaje de avance en el pre-procesamiento del video de entrada. Cada cuadro pre-procesado se muestra en el cuadro de imagen del panel.

El panel derecho, señalado en la figura 5.5 por el recuadro de color azul, incluye un sub-panel de información (señalado en color morado) con los resultados momentáneos de Sensibilidad y Especificidad (y los resultados finales una vez que se complete el procesamiento de los cuadros); también se incluye un indicador de alerta de incendio, que es activado al cumplir con el criterio de permanencia temporal. En la parte superior del panel, un menú desplegable permite elegir una de las etapas de detección del algoritmo propuesto; de acuerdo con la opción elegida, el cuadro de imagen muestra el mapa binario resultante de la etapa (sección marcada en color amarillo).

Finalmente, cada panel contiene botones para detener el procesamiento en cualquier momento; sin embargo, esto resultaría en entrenamientos parciales, menores cantidades de cuadros para prueba o resultados parciales de exactitud en el ejercicio de prueba.

Por último, se incluye una etapa de manejo de detecciones que consiste en una herramienta de análisis espacio-temporal. Considerando los índices de exactitud que se cuentan como valores de salida del sistema, cada cuadro de la secuencia de entrada tiene un efecto en las detecciones o en las no-detecciones, sin embargo, el disparo de una alarma se condicionó a la permanencia de las regiones de humo por un lapso de tres segundos.

Cuando una detección se mantiene durante los tres segundos, se dispara la alarma y se permite al usuario elegir entre mantenerla activa o detenerla (indicando generalmente una falsa alarma o que el incendio ya habría sido controlado).

Cada etapa del algoritmo propuesto es enfocada a tolerar las diferentes condiciones de iluminación establecidas para cumplir con el objetivo propuesto; la etapa de pre-procesamiento se enfoca en reducir la cantidad de información contenida en cada imagen, y reducir el efecto de los cambios de iluminación en las imágenes capturadas; posteriormente, la etapa de detección de movimiento se encarga de filtrar los pixeles para solamente analizar las regiones de interés en cada una de las etapas siguientes; los distintos valores de umbral para los análisis de características dinámicas y estáticas son adaptados de manera recursiva mediante procedimientos estadísticos, de modo que para cada condición de iluminación se pueden establecer los umbrales más adecuados. AdaBoost permite tomar una decisión a partir de todas las características analizadas en conjunto. El capítulo siguiente presenta los resultados obtenidos de las pruebas descritas, así como la discusión de estos.

El anexo 10 contiene el código correspondiente a la interfaz gráfica desarrollada, la configuración inicial, junto con los cambios en la interfaz con cada evento accionado; cabe resaltar que los códigos de las etapas de detección fueron adaptados para ser llamados como funciones externas que realizan el proceso una vez que son llamadas por una acción en la interfaz.

6 RESULTADOS Y DISCUSIÓN

6.1 Réplicas de algoritmos extraídos del estado del arte

De la forma descrita en la sección 5, las réplicas de los algoritmos extraídos del estado del arte fueron probadas con los videos descritos en la Tabla 5.1 con el objetivo de identificar y establecer una metodología de evaluación para los algoritmos de Visión Artificial implementados para la detección de humo. Se evaluó la sensibilidad y especificidad de cada algoritmo, así como la complejidad de cada herramienta.

A partir de los videos de prueba 1 y 3 descritos en la sección 5.2, se extrajeron 2640 imágenes con presencia de humo utilizados para calcular la sensibilidad de cada algoritmo. La Figura 6.1 muestra la gráfica con los resultados de sensibilidad obtenidos.

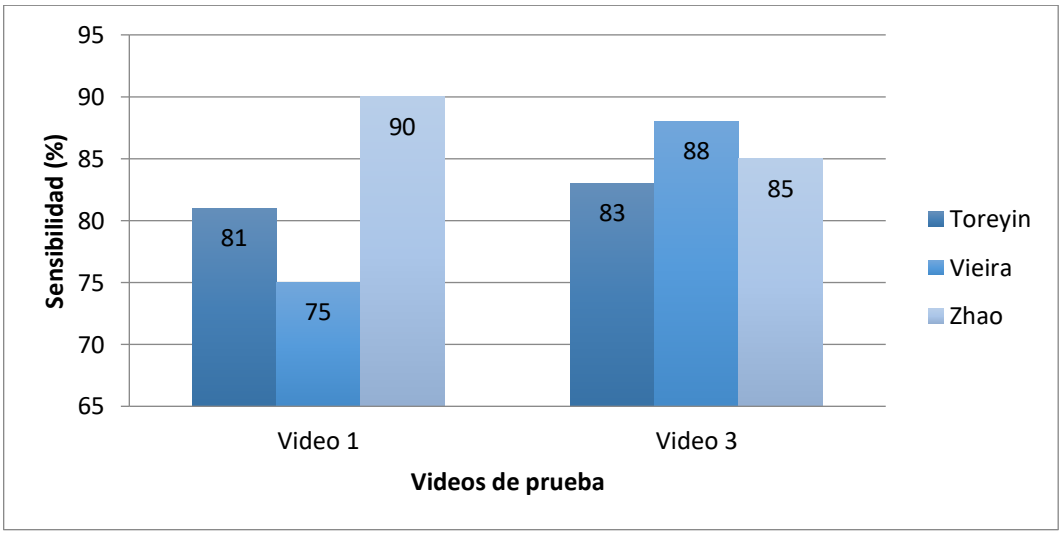

Figura 6.1: Gráfica comparativa de sensibilidad de los algoritmos en pruebas con videos con humo.

El valor más bajo de sensibilidad fue obtenido por el algoritmo de (Vieira et al. 2016), que para el video 1, detectó solamente el 75% de los cuadros con humo. El entrenamiento de la herramienta utilizada para el análisis RGB hace que el algoritmo trabaje para un tipo específico de humo. El mismo algoritmo alcanza un 88% con el video 3, pues las combinaciones de humo se asemejan más a las utilizadas como entrenamiento.

El algoritmo de (Zhao et al. 2015), obtuvo el valor de sensibilidad más alto en el primer video, con el 90% de detecciones correctas y el 85% en el segundo video. El entrenamiento de la herramienta AdaBoost es un modificador del comportamiento de la herramienta, a partir del conjunto de imágenes de entrenamiento el algoritmo se puede ajustar para trabajar mejor bajo diferentes condiciones.

El algoritmo de (Töreyin et al. 2005) presentó una variación menor en las condiciones de los video con humo. Se obtuvo un 89% de detecciones correctas en el video 1, y un 93% en el video 4. A diferencia de las otras dos herramientas, ésta no depende de un conjunto de entrenamiento. Condiciones similares de iluminación en los videos de entrada representan características similares detectadas en los cuadros, y con ello, una menor variación del resultado.

Por otro lado, la Figura 6.2 muestra el gráfico con los porcentajes de especificidad resultantes de las pruebas realizadas.

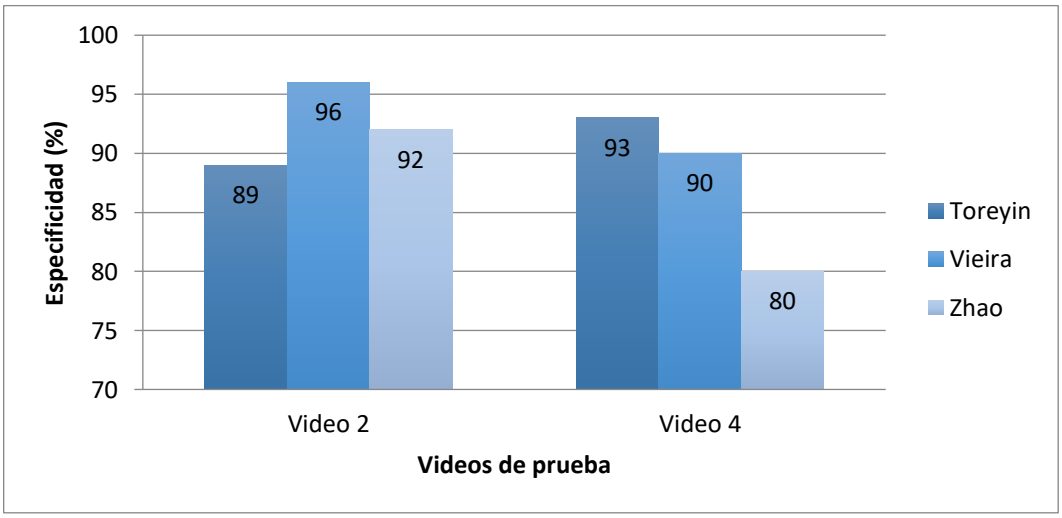

Figura 6.2: Gráfica comparativa de especificidad de los algoritmos en videos sin humo.

Las condiciones del entrenamiento para la herramienta de análisis RGB permiten que el algoritmo de (Vieira et al. 2016) mantenga porcentajes de no-detecciones correctas de 90% para el video 4 y 96% para el video 2. El video 2, que presenta condiciones de iluminación nocturna, obtiene un mayor porcentaje de no detecciones correctas, pues el

conjunto de entrenamiento, basado en columnas de humo gris y blanco, permite que no se detecten de manera errónea los pixeles de la escena.

La especificidad obtenida por el algoritmo de (Zhao et al. 2015) presenta la mayor variación entre las dos condiciones de iluminación analizadas, con el 92% en el video 2 y el 80% en el video 4. Las características dinámicas presentes en las escenas y la falta de una herramienta de análisis de color permiten que las variaciones provocadas por otros objetos generen falsos positivos que afectan la métrica.

Al igual que con la sensibilidad, el algoritmo de (Töreyin et al. 2005) presenta la menor variación entre las dos condiciones evaluadas, con el 89% de especificidad en el video 2 y el 93% en el video 4. Este comportamiento se debe a que el algoritmo realiza una menor cantidad de detecciones bajo condiciones de baja iluminación; la interferencia de otros objetos genera una mayor cantidad de falsos positivos, por lo que la especificidad del algoritmo es menor en el video que presenta la iluminación nocturna.

Los algoritmos de (Zhao et al. 2015) y de (Töreyin et al. 2005) tienen un mayor enfoque hacia la detección de características dinámicas, y dejan en un segundo plano el análisis del color. Esto reduce la dependencia del algoritmo a trabajar bajo condiciones específicas o para humo con características previamente definidas. El principal factor que permite al algoritmo de (Töreyin et al. 2005) presentar menos variaciones en los casos de prueba establecidos es que no requiere de un entrenamiento para realizar las detecciones, complementa el funcionamiento de las herramientas para análisis dinámico con el análisis de color en espacio YUV. En este sentido, se puede considerar que dicho algoritmo es la mejor opción para un sistema de detección de humo.

Por otro lado, el algoritmo de (Vieira et al. 2016) presenta una gran área de oportunidad para la implementación de sistemas reales, donde con un análisis adecuado del área monitoreada, las posibles fuentes de combustión y el color de humo generado por estas, puede ser utilizado para obtener una herramienta que supere el 80% de detecciones reales a un bajo costo computacional.

La principal ventaja del algoritmo de (Zhao et al. 2015) es la posibilidad de ejecutar cada etapa de manera paralela. Esto permite que se trabaje sobre la mayor cantidad

posible de regiones de interés. AdaBoost utiliza los resultados de cada etapa para generar un clasificador global. De esta manera, los resultados de cada etapa no son afectados por otras etapas y se puede realizar una detección aun cuando una de ellas no tenga un resultado positivo. Sus principales desventajas son la necesidad de un amplio entrenamiento para asegurar el correcto funcionamiento y que no considera características estáticas.

Se puede atribuir la variación entre los resultados obtenidos a la necesidad de entrenamiento de los algoritmos de (Vieira et al. 2016) y (Zhao et al. 2015). Mientras el primero emplea el entrenamiento como la parte central de su funcionamiento, por lo que su variación en diferentes escenarios es mayor, el segundo solamente utiliza el entrenamiento para actualizar los pesos de los indicadores de AdaBoost, el efecto de variación en los resultados es menor debido a que el funcionamiento de las etapas no es afectado por el entrenamiento.

Por otro lado, como se describe en la sección 5.2, los tiempos de ejecución son monitoreados con distintas cantidades de cuadros de entrada. La Figura 6.3 muestra los tiempos de ejecución resultantes de las pruebas realizadas con el video 1. Es posible identificar una tendencia similar en la respuesta de los algoritmos (Vieira et al. 2016) y (Zhao et al. 2015), aun cuando el tiempo de ejecución del primero es 2.5 veces mayor cuando se ejecuta la prueba con todos los cuadros disponibles. En ambos casos el tiempo de ejecución se eleva hasta aproximarse a un mismo valor; se puede inferir que existe una tendencia hacia un grado de complejidad proporcional logarítmica $O(nlogn)$. El algoritmo (Töreyin et al. 2005), presenta un menor tiempo de ejecución, con una tendencia hacia un grado de complejidad lineal $O(n)$.

Las gráficas de la respuesta en tiempo de ejecución de los algoritmos evaluados mediante las pruebas con el video 2, se muestran en la Figura 6.4. Dado que se procesan imágenes con la misma resolución que en el video 1, pero bajo diferentes condiciones de iluminación, y que este video no cuenta con humo que detectar, permite que los algoritmos realicen menos operaciones, aproximándose los tres a un comportamiento lineal.

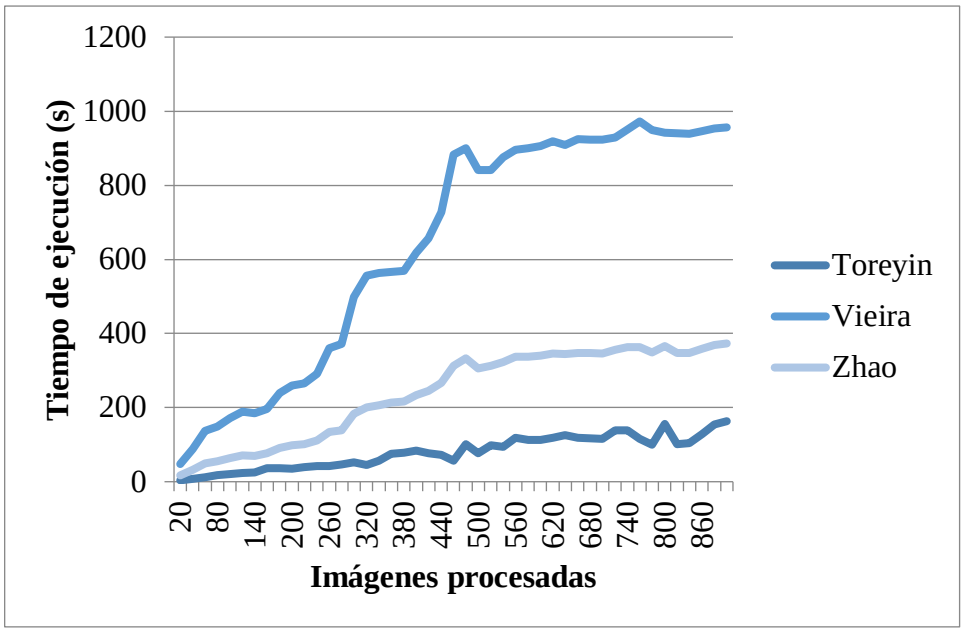

Figura 6.3: Gráficas de los tiempos de ejecución de los algoritmos evaluados con el video de prueba 1.

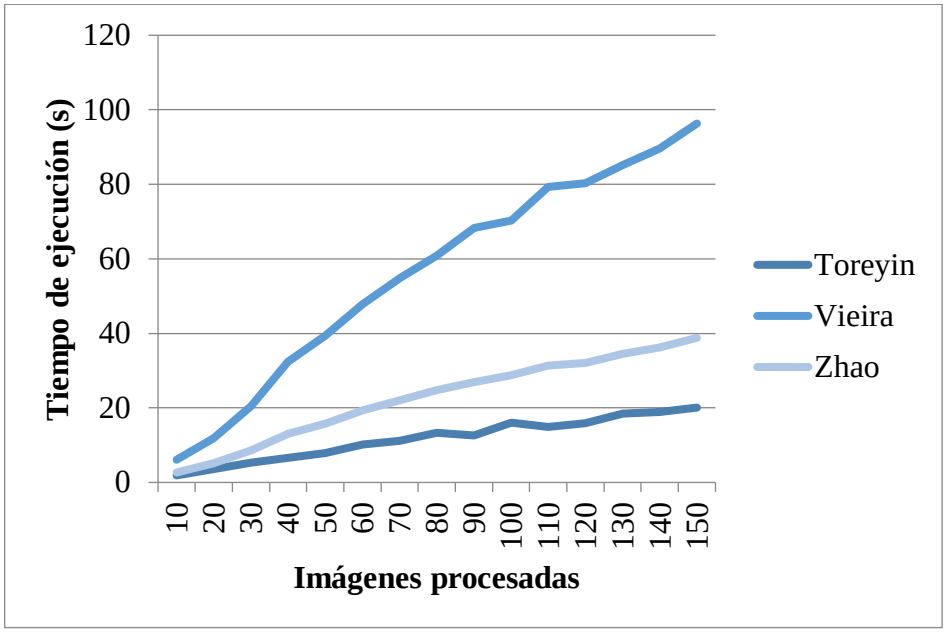

Figura 6.4: Gráficas de los tiempos de ejecución de los algoritmos evaluados con el video de prueba 2.

La Figura 6.5, muestra la respuesta de los algoritmos en las pruebas realizadas utilizando el video 3. En un caso similar al video 1, en el que existe presencia de humo, el

comportamiento de las tres herramientas es similar, con el algoritmo (Töreyin et al. 2005); presenta una tendencia a comportamiento lineal, mientras los otros dos muestran un comportamiento proporcional logarítmico. Sin embargo, cabe resaltar que el incremento en el número de pixeles debido a la resolución empleada en la captura del video 3, incrementa los tiempos de ejecución hasta 20 veces.

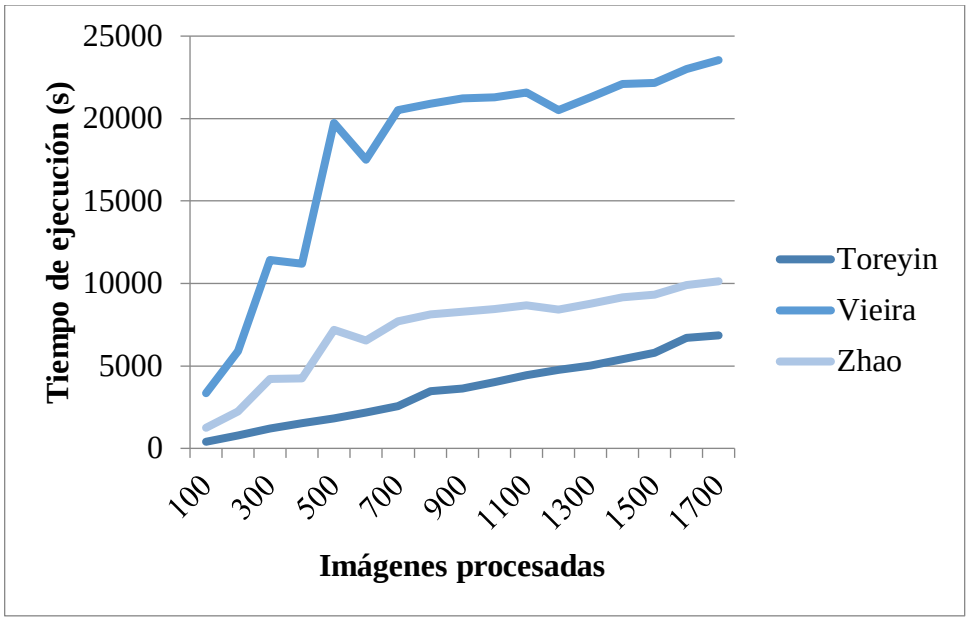

Figura 6.5: Gráficas de los tiempos de ejecución de los algoritmos evaluados con el video de prueba 3.

Por último, la Figura 6.6 muestra las gráficas de la respuesta en tiempo de ejecución de los algoritmos evaluados en pruebas con el video 4. En éstas, es apreciable que debido a que el proceso no requiere de detecciones, los tiempos de ejecución son menores a los del video 3, aun cuando la resolución es mayor a la de los videos 1 y 2. El algoritmo (Töreyin et al. 2005), presenta un comportamiento logarítmico hasta los 350 cuadros, con tiempos mayores de ejecución que los otros dos algoritmos; sin embargo, posteriormente es superado este tiempo de ejecución por el algoritmo (Vieira et al. 2016), que si bien muestra un comportamiento que tiende a lo lineal, la pendiente que describe se respuesta termina superando a las otras herramientas.

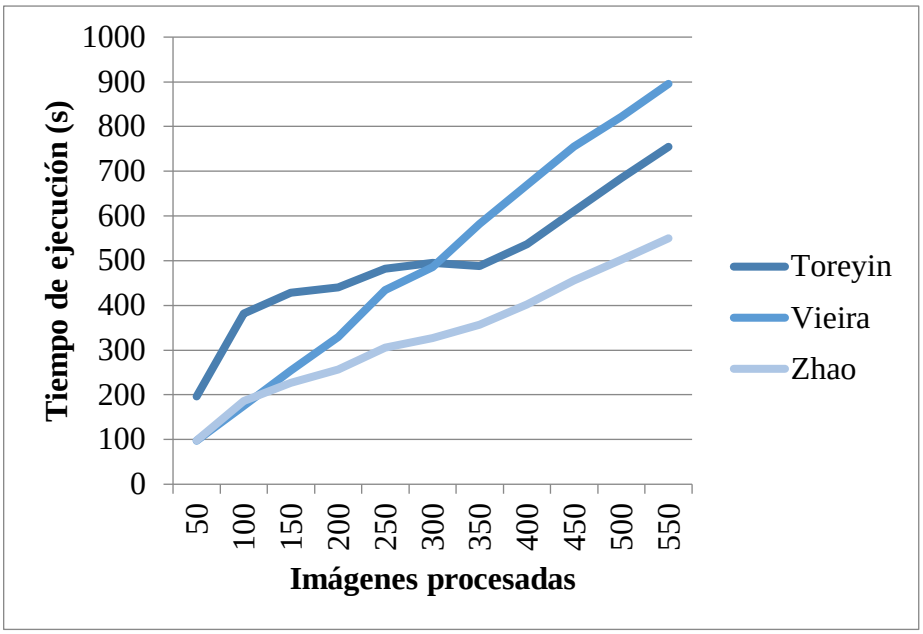

Figura 6.6: Gráficas de los tiempos de ejecución de los algoritmos evaluados con el video de prueba 4.

6.2 Pruebas Preliminares

Como se describe en la sección 5.2, cada etapa del algoritmo que se ha diseñado en este trabajo, y que ha sido propuesta para detección de características del humo, fue sometida a una evaluación preliminar con el cálculo de sensibilidad y especificidad. Los resultados de estas pruebas se muestran en la Tabla 6.1.

Tabla 6.1: Porcentajes de sensibilidad y especificidad de las pruebas preliminares realizadas a las etapas para detección

Etapa	**Sensibilidad (%)**	**Especificidad (%)**
Detección de Movimiento	95.23	98.74
Análisis de Dirección de Movimiento	90.35	96.5
Análisis en dominio de Wavelets	94.12	98.65
Análisis de color en espacio RGB	88.7	95.24
Análisis de color en espacio YCbCr	87.33	96.42

Cada una de las etapas obtuvo valores cercanos o superiores a las medias propuestas de sensibilidad (80%) y especificidad (85%) para el algoritmo. Cabe resaltar que cada una de las etapas posteriores a la detección de movimiento depende de los resultados de esa etapa, que es empleada como filtro.

Las etapas fueron validadas posteriormente mediante la selección de cuadros de manera aleatoria para verificar que las regiones detectadas coincidan con bloques de pixeles con humo.

Cabe aclarar que para estas pruebas preliminares, el etiquetado descrito en la sección 5.4 del presente documento fue modificado, en lugar de etiquetar los cuadros con y sin humo, la etiqueta se colocó para los cuadros con y sin la presencia de la característica analizada, esto es, para la primera etapa la etiqueta "S" se colocó en los cuadros que presentan movimiento, y "N" para aquellos que no. De esta forma, se valida el funcionamiento de cada etapa por separado, sin afectar los porcentajes de exactitud con el uso de los resultados generales esperados para la herramienta completa.

Con lo anterior, se justifica que el porcentaje de especificidad sea superior al 85% propuesto aun considerando los cuadros con sin humo y con movimiento de otros objetos. Puesto que el porcentaje de cada etapa es mayor al propuesto, se considera que su aporte al funcionamiento de la herramienta completa es suficiente, ya que no descarta más cuadros de los que permiten mantener la exactitud propuesta.

6.3 Resultados de las pruebas de exactitud

El algoritmo compuesto por todas las etapas para detección, el pre-procesamiento y el clasificador AdaBoost, fue evaluado para cada uno de los casos de prueba descritos en la sección 5.4.1. Cabe resaltar que tanto el criterio de sensibilidad como el de especificidad fueron calculados para cada condición de iluminación, considerando dos casos de prueba para la misma condición (con humo y sin humo).

El número de cuadros utilizados para la evaluación varía debido a la duración y el número de cuadros por segundo de cada video empleado como entrada. La iluminación diurna fue evaluada a partir de 5488 cuadros sin humo y 17905 cuadros con humo.

15619 cuadros fueron detectados con humo. Considerando la ecuación 5.21, la sensibilidad resultante se obtuvo como:

$$Sensibilidad_{Luz\ Diurna} = \frac{15619}{15619 + 2286} \quad 6.1$$

$$Sensibilidad_{Luz\ Diurna} = 87.2326 \quad 6.2$$

La tabla 6.2, muestra los resultados obtenidos por las pruebas realizadas en video de humo bajo iluminación diurna; en ésta se indican el número de cuadros del video tomados para la prueba, así como la cantidad de cuadros detectados, correspondiente al número de Verdaderos Positivos y los no detectados, indicados como Falsos Negativos. La columna Sensibilidad muestra el porcentaje obtenido para cada video, calculado de manera análoga a la ecuación 6.1; Por último, se muestran las cantidades totales obtenidas por la prueba, junto con el porcentaje de sensibilidad mostrado en la ecuación 6.2.

Tabla 6.2: Pruebas con videos de humo bajo iluminación diurna.

Video de prueba	**Total de cuadros**	**Verdaderos Positivos**	**Falsos Negativos**	**Sensibilidad**
Video con Humo 1	6815	5922	893	86.8965
Video con Humo 2	9350	8156	1194	87.2299
Video con Humo 3	1740	1541	199	88.5632
Total	17905	15619	2286	87.2326

La Figura 6.7 muestra el gráfico comparativo de los resultados obtenidos durante las pruebas de Sensibilidad bajo iluminación Diurna.

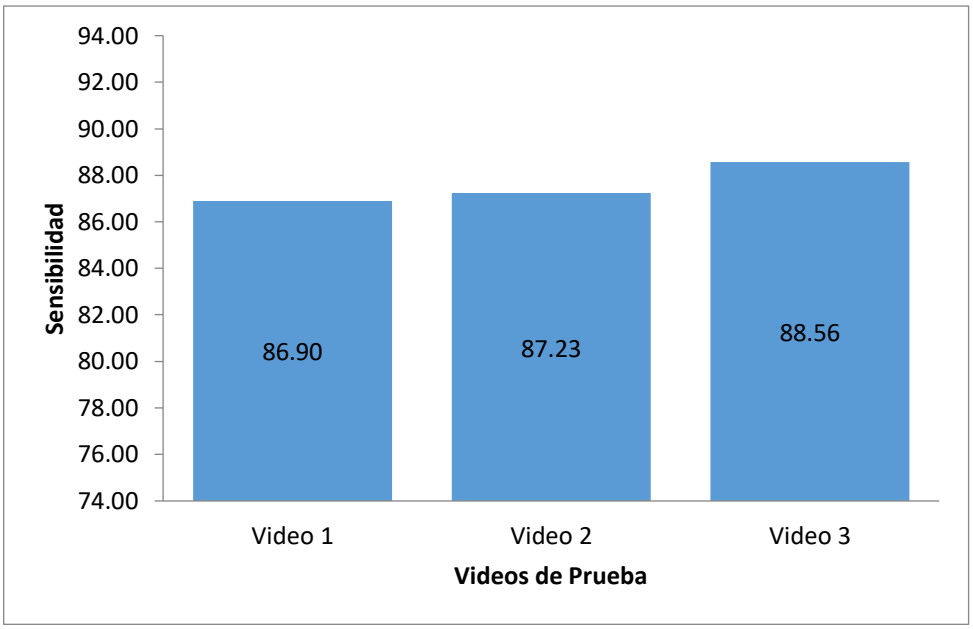

Figura 6.7: Porcentajes de Sensibilidad obtenidos por cada video de prueba con humo en condiciones de iluminación Diurna.

Por otro lado, bajo las mismas condiciones, el algoritmo detectó 540 falsos positivos, en contraste con los 4948 cuadros sin humo que no fueron detectadas de manera acertada. A partir de la ecuación 5.22, la especificidad fue calculada como:

$$Especificidad_{Luz\ Diurna} = \frac{4948}{4948 + 540} \tag{6.3}$$

$$Especificidad_{Luz\ Diurna} = 90.1603 \tag{6.4}$$

La tabla 6.3, muestra los resultados de especificidad obtenidos por las pruebas realizadas en videos sin humo bajo iluminación diurna; se indican el número de cuadros del video, la cantidad de cuadros no detectados, correspondiente al número de Verdaderos Negativos y los detectados o Falsos Positivos. La columna Especificidad muestra el porcentaje obtenido para cada video.

Tabla 6.3: Pruebas con videos sin humo bajo iluminación diurna.

Video de prueba	Total de cuadros	Verdaderos Negativos	Falsos Positivos	Especificidad
Video sin Humo 1	1979	1704	275	86.104
Video sin Humo 2	1769	1655	114	93.5556
Video sin Humo 3	1740	1589	151	91.3218
Total	5488	4984	540	90.1603

La Figura 6.8 muestra el gráfico comparativo de los valores de especificidad obtenidos por los videos empleados para la prueba.

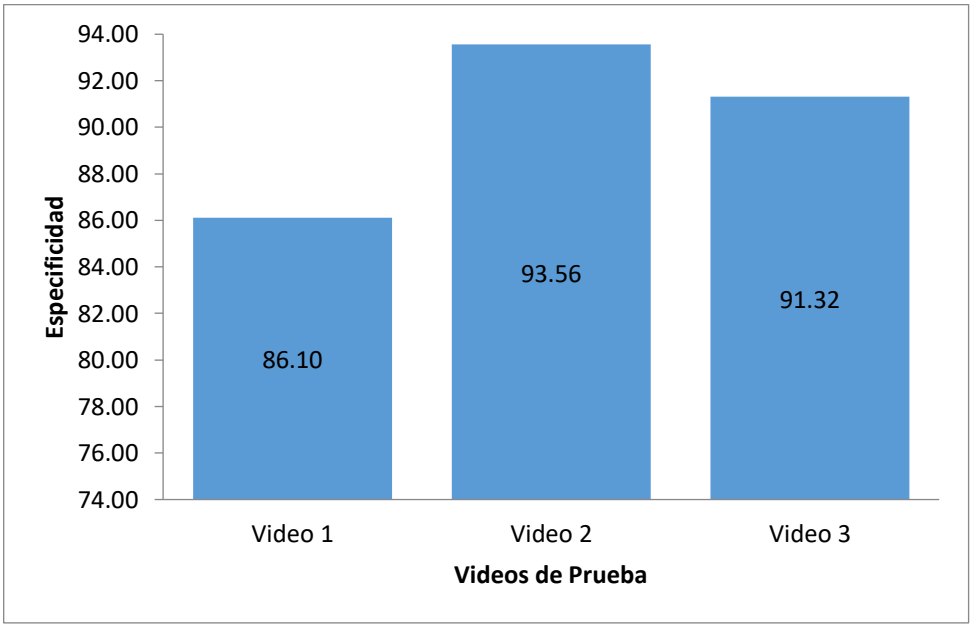

Figura 6.8: Porcentajes de Especificidad obtenidos por cada video de prueba sin humo en condiciones de iluminación Diurna.

De la misma forma, bajo condiciones de iluminación en día nublado, se evaluaron 10800 cuadros sin humo y 7860 cuadros con humo. La ejecución del algoritmo resultó en 8059 cuadros detectados, con 6790 verdaderos y 1269 falsos positivos. Del resto de cuadros evaluados, se obtuvieron 9531 verdaderos negativos y 1070 falsos negativos. La Sensibilidad y Especificidad obtenidas fueron:

$$Sensibilidad_{Día\ Nublado} = \frac{6790}{6790 + 1070} = 86.3867 \quad 6.5$$

$$Especificidad_{Día\ Nublado} = \frac{9531}{9531 + 1269} = 88.25 \quad 6.6$$

Las Tablas 6.4 y 6.5 muestran los resultados obtenidos por las pruebas de Sensibilidad y Especificidad bajo condiciones de iluminación en día nublado, las cantidades de cuadros de cada video, los cuadros detectados o no detectados según corresponde al cálculo del criterio de exactitud descrito por la tabla y los porcentajes de cada criterio para el total de cuadros empleado en la prueba y para cada video empleado.

Tabla 6.4: Pruebas en videos con humo bajo iluminación de día nublado.

Video de prueba	**Total de cuadros**	**Verdaderos Positivos**	**Falsos Negativos**	**Sensibilidad**
Video con Humo 1	3930	3303	627	84.0458
Video con Humo 2	3630	3220	410	88.7052
Video con Humo 3	300	267	33	89.0000
Total	7860	6790	1070	86.3868

Tabla 6.5: Pruebas con videos sin humo bajo iluminación de día nublado.

Video de prueba	**Total de cuadros**	**Verdaderos Negativos**	**Falsos Positivos**	**Especificidad**
Video sin Humo 1	3960	3386	574	85.5050
Video sin Humo 2	3960	3549	411	89.6212
Video sin Humo 3	2880	2596	284	90.1388
Total	10800	9531	1269	88.2500

Las Figuras 6.9 y 6.10, muestran los gráficos con los porcentajes de sensibilidad y especificidad obtenidos por cada video bajo condiciones de iluminación en día nublado

(videos con humo para el cálculo de sensibilidad y videos sin humo para la especificidad).

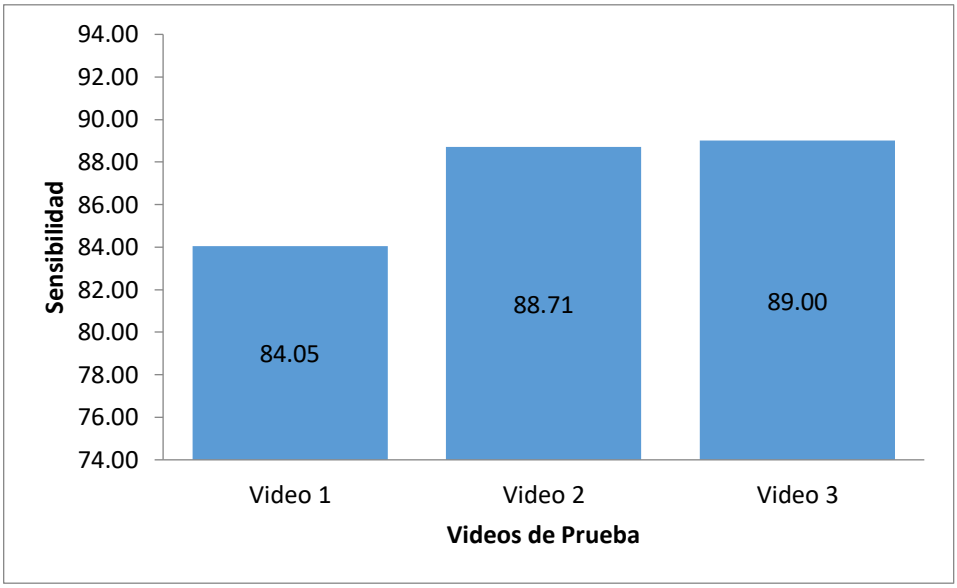

Figura 6.9: Porcentajes de Sensibilidad obtenidos por cada video de prueba con humo en condiciones de día nublado.

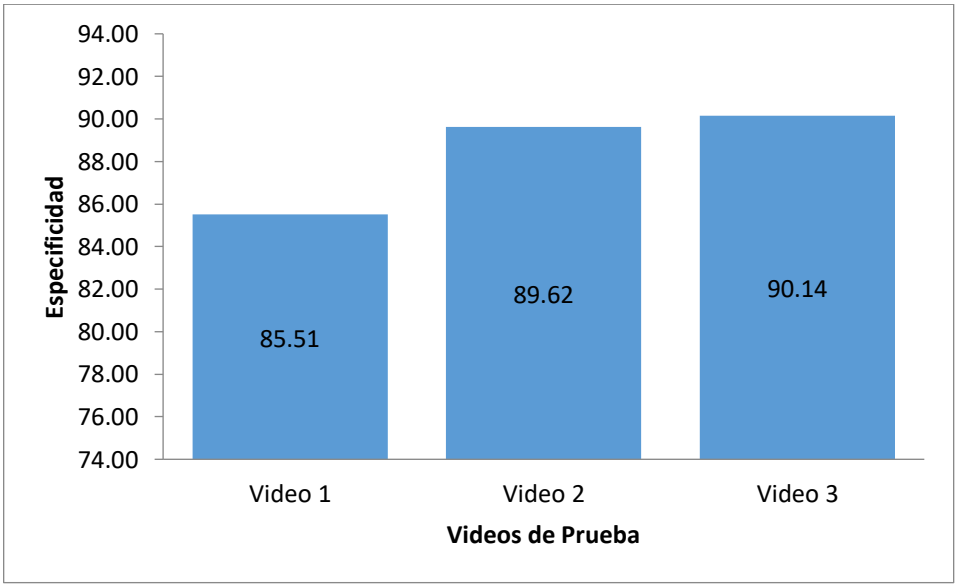

Figura 6.10: Porcentajes de Especificidad obtenidos por cada video de prueba sin humo en condiciones de día nublado.

Finalmente, la condición de luz nocturna fue evaluada a partir de 3910 cuadros sin humo y 5340 cuadros con humo. Se obtuvieron 4567 detecciones, con 248 falsos positivos y 4319 acertadas. También se obtuvo un total de 4683 no detecciones, con 1021 falsos negativos. La Sensibilidad y Especificidad obtenidas fueron:

$$Sensibilidad_{Luz\ Nocturna} = \frac{4319}{4319 + 1021} = 80.8801 \qquad 6.7$$

$$Especificidad_{Luz\ Nocturna} = \frac{3662}{3662 + 248} = 93.6572 \qquad 6.8$$

Los resultados obtenidos por las pruebas de Sensibilidad y Especificidad bajo condiciones de iluminación nocturna se muestran en las Tablas 6.6 y 6.7, así como el número de cuadros detectados y no detectados de cada video y los porcentajes del criterio de exactitud correspondiente al caso de prueba descrito por la tabla.

Tabla 6.6: Pruebas en videos con humo bajo iluminación nocturna.

Video de prueba	**Total de cuadros**	**Verdaderos Positivos**	**Falsos Negativos**	**Sensibilidad**
Video con Humo 1	1900	1537	363	80.8947
Video con Humo 2	1160	943	217	81.2931
Video con Humo 3	2280	1839	441	80.6578
Total	5340	4319	1021	80.8801

Tabla 6.7: Pruebas con videos sin humo bajo iluminación nocturna.

Video de prueba	**Total de cuadros**	**Verdaderos Negativos**	**Falsos Positivos**	**Especificidad**
Video sin Humo 1	150	140	10	93.3333
Video sin Humo 2	160	155	5	96.8750
Video sin Humo 3	3600	3367	233	93.5277
Total	3910	3662	248	93.6572

Los gráficos comparativos de los porcentajes obtenidos por cada video durante las pruebas realizadas bajo iluminación nocturna se muestran en la Figuras 6.11 y 6.12

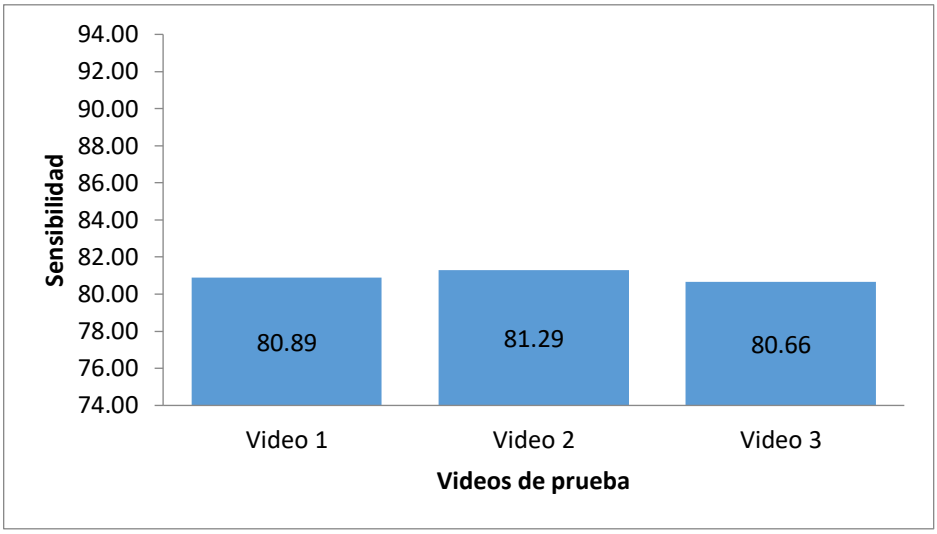

Figura 6.11: Porcentajes de Sensibilidad obtenidos por cada video de prueba con humo en condiciones iluminación Nocturna.

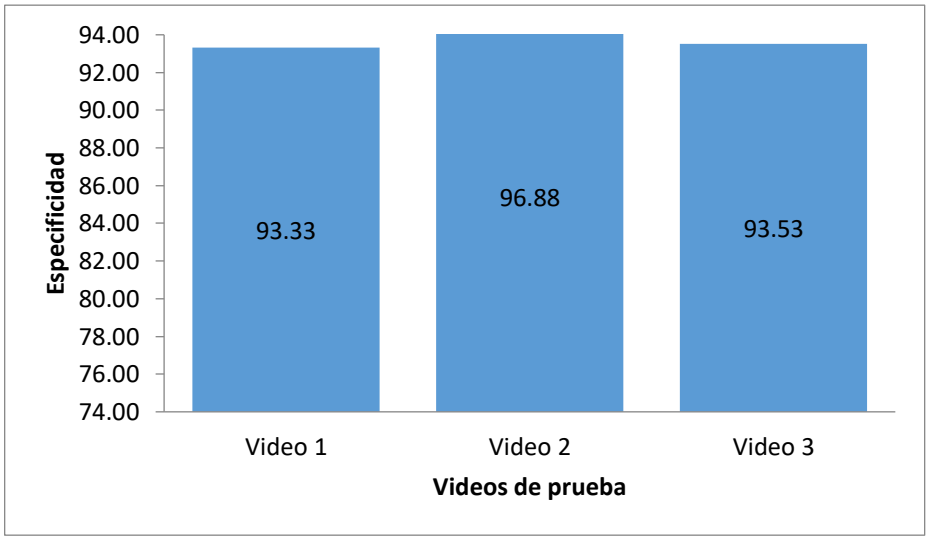

Figura 6.12: Porcentajes de Especificidad obtenidos por cada video de prueba sin humo en condiciones iluminación Nocturna.

La sensibilidad obtenida por el algoritmo es menor cuando la iluminación de la escena es reducida. La Figura 6.13 muestra un gráfico con el comportamiento del algoritmo ante cambios de iluminación.

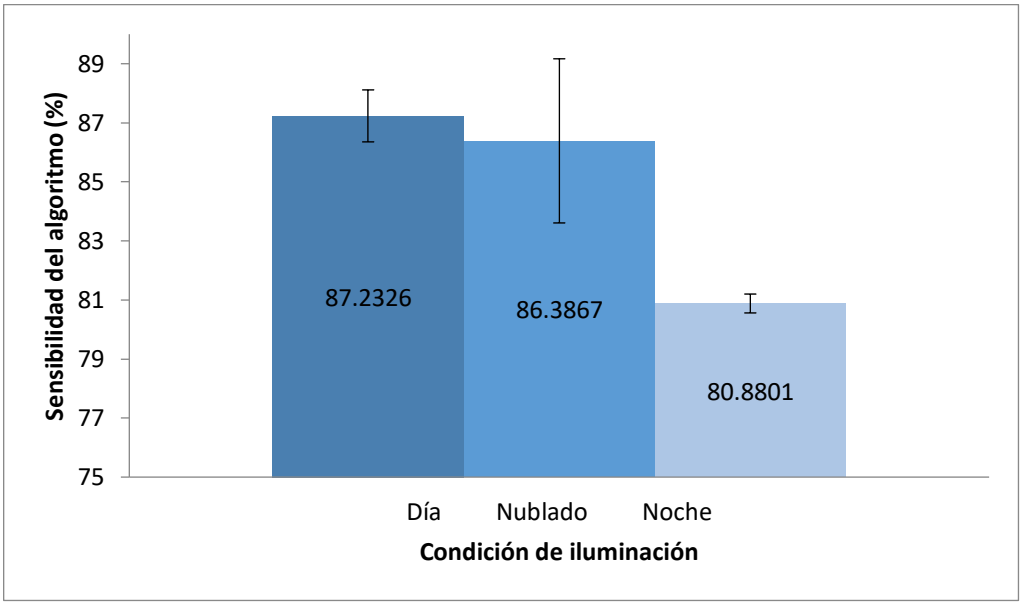

Figura 6.13: Gráfico de sensibilidad bajo diferentes condiciones de iluminación

La sensibilidad promedio obtenida por el algoritmo propuesto en condiciones de iluminación diurna es de 87.2326%, con un error del 5.12%. Para la condición de día nublado, la sensibilidad es de 86.3867% con un error de 5.04%. Por último, Los resultados de los videos con iluminación nocturna, resultan en 80.8801% de sensibilidad, con un error de 0.69% entre las pruebas realizadas.

Existe una variación de 0.9697% entre la sensibilidad obtenida por el algoritmo en condiciones de luz diurna y condiciones de luz en día nublado. Esto significa que existe una reducción menor al uno por ciento de detecciones correctas. Por otro lado, la sensibilidad en luz nocturna muestra una reducción del 7.2822% con respecto al resultado de luz diurna.

La especificidad muestra un comportamiento distinto, pues alcanza su valor mínimo en condiciones de día nublado, donde las condiciones atmosféricas generar un incremento en el número de pixeles con condiciones similares al humo, y con ello un incremento en el porcentaje de falsos positivos. La Figura 6.14 muestra un gráfico con el comportamiento de la especificidad del algoritmo.

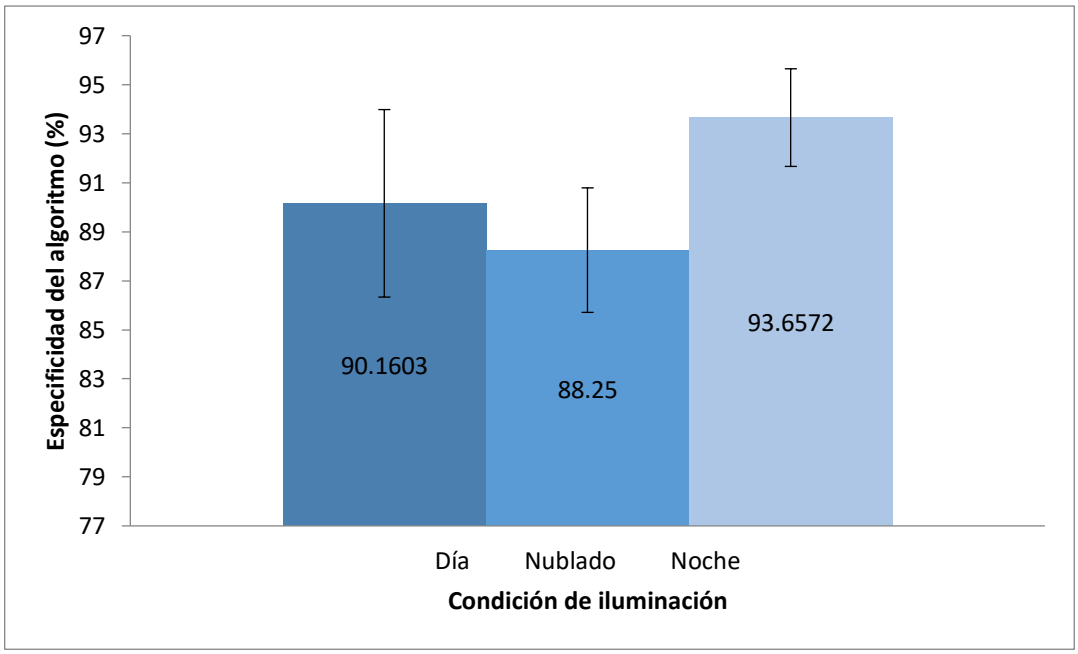

Figura 6.14: Gráfico de especificidad bajo diferentes condiciones de iluminación.

La especificidad del sistema es mayor en condiciones de iluminación nocturna, ya que tanto la herramienta para detección de color, como el entrenamiento de la herramienta AdaBoost, permiten descartar diversas regiones que si bien poseen características similares a las buscadas por la herramienta, no representan una presencia de humo. El porcentaje de no-detecciones es mayor en esas condiciones, incrementando también el número de no-detecciones realizadas correctamente. Se obtuvo una variación de 2.1187% entre las condiciones de iluminación diurna y de día nublado, y un incremento de 3.8785% en las condiciones de luz nocturna con respecto al resultado en luz diurna.

A partir de los resultados obtenidos, se puede observar que la sensibilidad del algoritmo propuesto se mantiene con muy poca variación entre distintos ejercicios con condiciones similares de iluminación, con una variación menor a un punto porcentual tanto para iluminación diurna como para nocturna. Esto implica que los ajustes realizados a las etapas para detección, permiten que bajo condiciones similares de iluminación, donde los escenarios comparten las combinaciones de colores del humo, los umbrales calculados estadísticamente sean valores muy parecidos. Por otro lado, bajo condiciones de día nublado, la interferencia de un escenario con las mismas condiciones estáticas del

humo modifica el porcentaje de falsos positivos, generando una variación mayor a los dos puntos porcentuales entre cada prueba realizada.

La Especificidad, sin embargo, es afectada de distinta manera, el porcentaje de falsos positivos es incrementado debido a la coincidencia de objetos que dentro de una escena poseen las características buscadas. Bajo condiciones de día nublado, donde se encuentran más interferencias de este tipo, el porcentaje encuentra sus resultados más bajos. Aun así, la desviación de los resultados con respecto a la media es menor al 3%. En condiciones de iluminación nocturna, se encontró una menor variación, con menos del 2% con respecto a la media; la herramienta de análisis de color juega un papel fundamental para descartar las regiones de pixeles que no son humo. La mayor variación en la especificidad se encontró en condiciones de luz diurna, donde los resultados tienen una varianza de 3.8 con respecto a la media; con una mayor iluminación, el efecto de sombras presentes en la imagen no es completamente eliminado por el ajuste en la herramienta de análisis Wavelet. Los resultados de dicha etapa, aunados a la presencia de nubes dispersas en la escena, pueden ocasionar variaciones en los resultados de la detección.

Considerando los resultados obtenidos entre las distintas condiciones de iluminación, como se muestra en la Figura 6.9, se cumplió con el porcentaje propuesto en la hipótesis de 80% en la sensibilidad. De igual forma, la figura 6.14, permite observar que el porcentaje de especificidad propuesto (85%), también es alcanzado por la herramienta. Cabe mencionar que la mayor área de oportunidad para la herramienta propuesta, es reducir el número de falsos positivos para incrementar la exactitud del sistema. En (Çetin et al. 2013) se presenta una revisión de diversas propuestas del estado del arte que se pueden adaptar para mejorar el resultado obtenido. Por un lado, complementar el análisis de color con un espacio de saturación como HSV o HSI, puede reducir la sensibilidad de la etapa a los cambios de iluminación; el uso de imágenes capturadas mediante dispositivos con iluminación infrarroja, posibilita mejorar la detección de humo en luz nocturna. Por otro, la detección de características dinámicas se puede mejorar con la implementación del relleno de manchas, descrito en (Collins et al. 2000)

permitiendo ajustar la forma en que se definen las regiones de interés y su posterior análisis.

6.4 Parámetros de calidad

Como una prueba adicional del funcionamiento de la herramienta propuesta y con la intención de evaluar la posible mejora de los resultados mediante un mejor ajuste paramétrico de la herramienta para detección de movimiento, que realiza el filtrado para descartar regiones que no son analizadas por las etapas posteriores, se evaluó el resultado estableciendo distintos valores en los parámetros de asimilación y umbrales predefinidos que no alteran la caracterización realizada del humo.

Considerando solamente las condiciones de iluminación diurna, que fungen como caso de prueba base para la herramienta propuesta, se evaluaron los mismos videos empleados para la prueba anterior.

El primer parámetro evaluado es α, que determina el grado de influencia del nuevo cuadro en la actualización de la matriz de fondo. El parámetro fue establecido en diferentes valores, desde cero hasta uno, con un tamaño de paso de 0.1. La Tabla 6.8 muestra los resultados de sensibilidad y especificidad correspondientes a cada prueba realizada.

Tabla 6.8: Criterios de exactitud obtenidos para los distintos valores en el parámetro α

Criterio	**Video de prueba**	$\boldsymbol{\alpha = 0.0}$	$\boldsymbol{\alpha = 0.1}$	$\boldsymbol{\alpha = 0.2}$	$\boldsymbol{\alpha = 0.3}$	$\boldsymbol{\alpha = 0.4}$	$\boldsymbol{\alpha = 0.5}$	$\boldsymbol{\alpha = 0.6}$	$\boldsymbol{\alpha = 0.7}$	$\boldsymbol{\alpha = 0.8}$	$\boldsymbol{\alpha = 0.9}$	$\boldsymbol{\alpha = 1.0}$
Sensibilidad	Video con Humo 1	55	58	61	65	75	77	82	85	83	83	78
	Video con Humo 2	56	62	66	71	78	80	83	86	84	84	82
	Video con Humo 3	60	67	75	83	81	86	85	91	87	86	87
	Media	**57%**	**62%**	**67%**	**73%**	**78%**	**81%**	**83%**	**87%**	**85%**	**84%**	**82%**
Especificidad	Video sin Humo 1	95	93	94	90	89	87	86	86	78	75	74
	Video sin Humo 2	97	96	96	96	95	93	91	91	85	81	79
	Video sin Humo 3	99	99	98	97	96	94	93	92	88	83	83
	Media	**97%**	**96%**	**96%**	**94%**	**93%**	**91%**	**90%**	**90%**	**84%**	**80%**	**79%**

La sensibilidad obtenida por el algoritmo propuesto alcanza su valor más bajo con el parámetro establecido en 0.0, pues el fondo no recibe una actualización, más bien, es igualado al cuadro anterior al procesado, lo que dificulta el filtrado de movimiento para el algoritmo. El resultado de sensibilidad se incrementa junto con el parámetro, sin embargo, cuando α alcanza un valor superior a 0.7, la sensibilidad desciende nuevamente, debido a que con el valor cercano a uno, nuevamente, la actualización del fondo se convierte en una réplica del cuadro. La Figura 6.15 muestra el comportamiento de la sensibilidad ante los cambios en α.

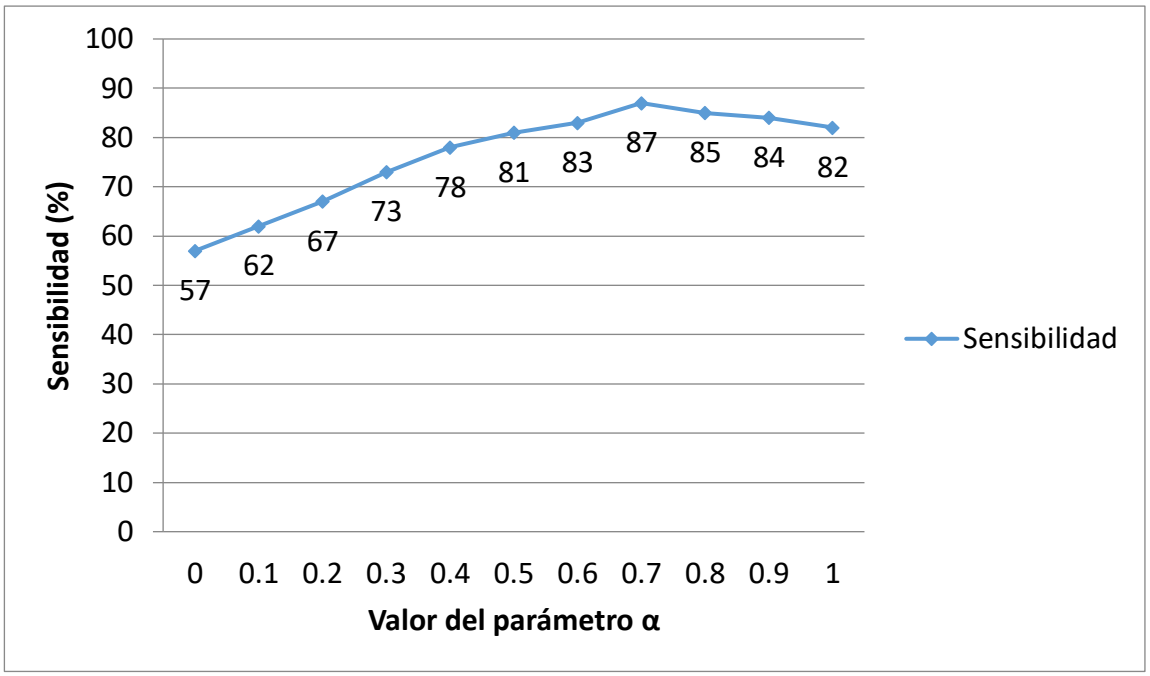

Figura 6.15: Gráfico de variación de sensibilidad a partir de distintos valores del parámetro α.

Dado que el principal aporte de la herramienta para detección de movimiento dentro del algoritmo propuesto, es el filtrado de la imagen para obtener regiones de interés, la actualización del fondo se vuelve un componente clave para el algoritmo. Si el parámetro α se establece en un valor cercano a cero, la sustracción de fondo se reduce a una diferencia entre el cuadro analizado y el cuadro anterior, puesto que la actualización del fondo asigna el valor del cuadro actual a todos los pixeles sin movimiento. Conforme se incrementa el valor del parámetro, el valor del fondo implica una mayor diferencia

con respecto al cuadro. Cuando el valor del parámetro se aproxima a 1, se descarta el valor de la imagen y el fondo no recibe actualización, con lo que la sustracción del fondo se vuelve una diferencia entre el cuadro actual y el primer cuadro. Lo anterior explica que con el incremento en el valor del parámetro se muestre una reducción en la sensibilidad, al verse afectada la cantidad de pixeles filtrados por la etapa en cuestión.

La especificidad por el contrario, se reduce mientras el valor de α es incrementado. Con el parámetro establecido en un valor cercano a cero, se incrementa el número de cuadros no-detectados correctamente; en caso contrario cuando el parámetro se establece en un valor cercano a 1, la cantidad de no-detecciones correctas disminuye. La Figura 6.16 muestra el gráfico con el comportamiento de la especificidad en la prueba.

El comportamiento de este criterio de exactitud se debe a que con un mayor valor en el parámetro de actualización para el fondo (α), son más las regiones de pixeles sin filtrar, y con ello una mayor cantidad de pixeles es analizada por las etapas posteriores. Así mismo, distintos objetos estáticos que podrían ser descartados, alteran los resultados de la herramienta, incrementando el porcentaje de falsos positivos.

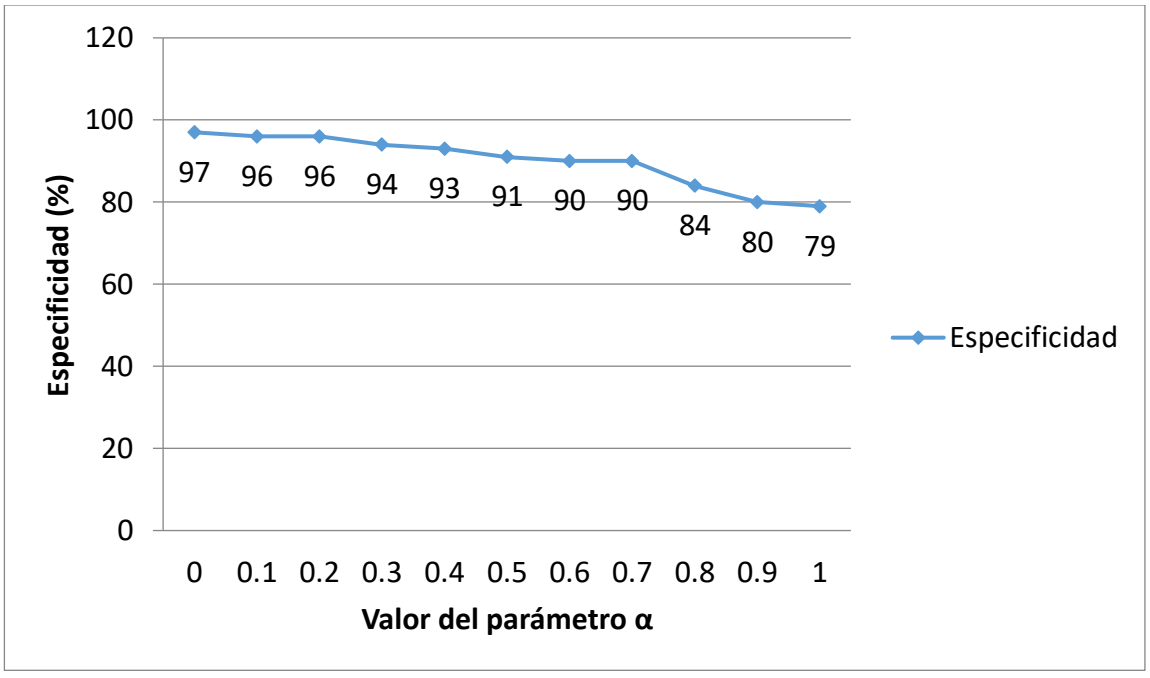

Figura 6.16: Gráfico de variación de especificidad a partir de distintos valores de α

Posteriormente, se evaluó el comportamiento del algoritmo con diferentes definiciones del valor de umbral para el análisis de reducciones en energía de frecuencia de Wavelets. Se comparó por un lado el funcionamiento del algoritmo con la implementación convencional propuesta en (Töreyin et al. 2005; Zhao et al. 2015), donde se busca únicamente un decremento en el valor de energía entre el fondo y la imagen, y por otro lado el funcionamiento con la comparación con un valor de umbral propuesta, tanto mediante la definición del valor de umbral constante como su obtención por el método de Otsu.

La Tabla 6.9 muestra los resultados de la comparación del funcionamiento del algoritmo propuesto con las condiciones descritas. Al realizar el cálculo del valor de umbral mediante el método de Otsu, la sensibilidad mejora en 2.35% con respecto a la detección de reducciones en la energía de Wavelets. Cuando el umbral es definido como constante, la sensibilidad es reducida en 5.88%, debido a la falta de adaptación del umbral a diferentes escenarios.

Tabla 6.9: Valores de exactitud obtenidos mediante los diferentes métodos para determinar el umbral de energía de frecuencia

Caso evaluado	**Sensibilidad**	**Especificidad**
Detección de reducción de energía	85%	87%
Contrate con umbral constante	80%	89%
Contraste con Umbral calculado por método de Otsu	87%	90%

La especificidad también se ve incrementada en 3.44% utilizando el método de Otsu, debido a la reducción de la cantidad de falsos positivos generados por obstrucciones de otros objetos en los bordes de las imágenes.

Como se muestra en la Tabla 6.9, el cálculo de umbral mediante el método de Otsu genera mayores porcentajes de sensibilidad y especificidad que los obtenidos por la herramienta con un umbral constante y la detección de reducción de energía Wavelet.

Considerando los parámetros evaluados en ésta sección, se puede determinar también que el correcto ajuste del parámetro de actualización del fondo presenta una oportunidad de mejora para la herramienta propuesta; si bien, el valor ajustado como constante (obtenida de acuerdo al estado del arte), presenta porcentajes de sensibilidad y especificidad que superan los propuestos en la hipótesis, reajustar el parámetro de manera dinámica, podría permitir mejorar el ajuste bajo distintas condiciones de iluminación.

6.5 Comparativa de resultados con el estado del arte

Los resultados descritos en la sección 6.3, fueron contrastados con los obtenidos en el estado del arte. Para esto, se reevaluaron los algoritmos replicados para cada caso de prueba propuesto. La Tabla 6.10 muestra los resultados obtenidos por los algoritmos en cada caso de prueba.

La Figura 6.17 muestra el gráfico con los resultados de sensibilidad obtenidos por los algoritmos analizados. En ésta se puede observar el comportamiento general de las herramientas evaluadas, que presentan variaciones menores al 3% en el caso de los algoritmos del estado del arte, y menor al 2% en el algoritmo propuesto.

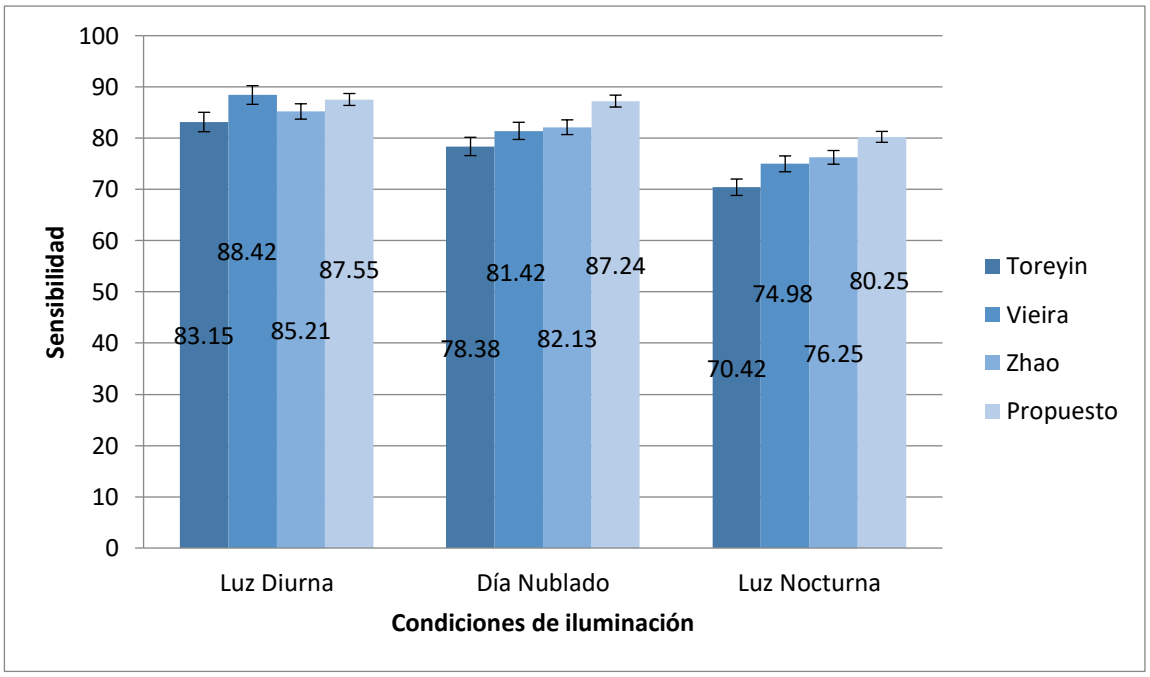

Figura 6.17: Gráfico comparativo de la sensibilidad obtenida por los algoritmos analizados

En cuanto a la Especificidad, la Figura 6.18 muestra los resultados obtenidos por los algoritmos analizados. De manera similar a la sensibilidad, se muestra la variación presente entre los distintos porcentajes obtenidos por las herramientas, que fue menor para las ejecuciones en luz nocturna y día nublado, con menos del 3% para cada algoritmo y un valor menor al 4% para las ejecuciones en luz diurna, es decir, el error representado en la gráfica, es menor al 4% en iluminación diurna y 3% para los otros casos, esto implica que los resultados de la ejecución en diferentes videos con las mismas condiciones no presentan variaciones mayores a los porcentajes descritos. La disminución del efecto de iluminación incrementa el porcentaje de no detecciones, mejorando los valores de especificidad. La disminución del porcentaje de falsos positivos resulta en una menor variación de la especificidad en condiciones de iluminación reducida (día nublado y luz nocturna).

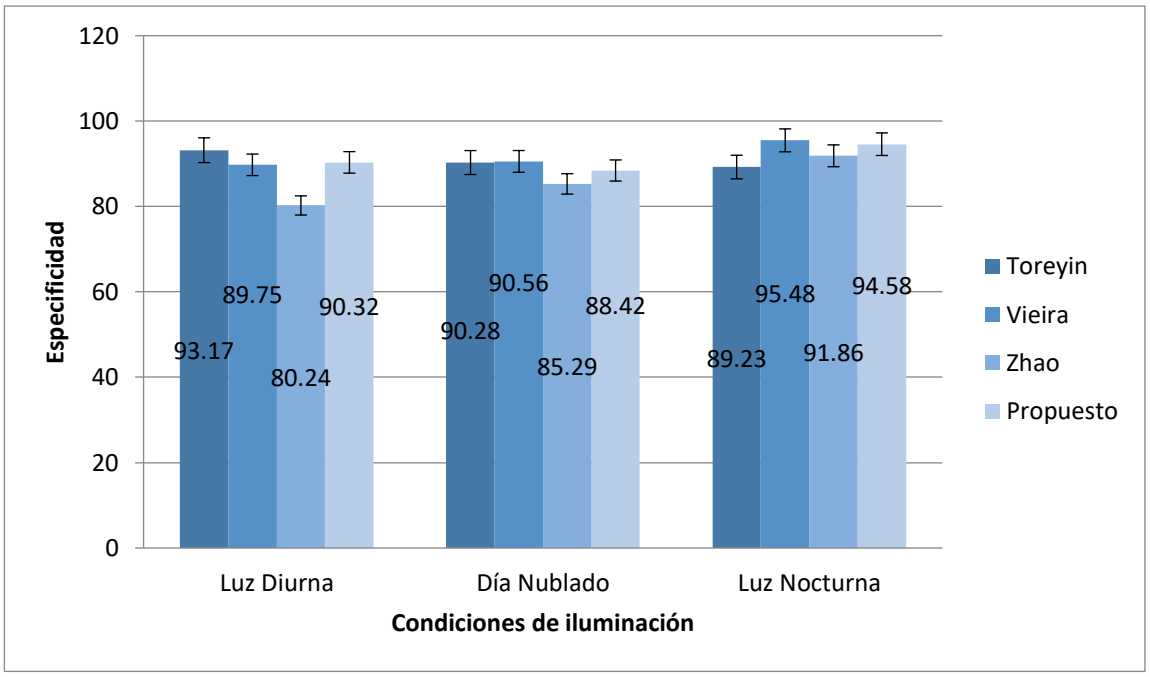

Figura 6.18: Gráfico de comparativo de la Especificidad obtenida por los algoritmos analizados

.

Tabla 6.10: Resultados de sensibilidad obtenidos por los algoritmos replicados del estado del arte para los casos de prueba establecidos

Criterio	Caso de prueba	Video de prueba	(Töreyin et al. 2005)	(Vieira et al. 2016)	(Zhao et al. 2015)	Algoritmo Propuesto
Sensibilidad	Luz diurna	Video 1	80.95	86.74	83.88	86.89
		Video 2	82.98	88.98	84.89	87.20
		Video 3	85.52	89.54	86.86	88.56
		Media	**83.15**	**88.42**	**85.21**	**87.55**
	Día nublado	Video 1	75.51	78.43	79.12	84.04
		Video 2	79.69	82.78	83.50	88.70
		Video 3	79.94	83.05	83.77	88.98
		Media	**78.38**	**81.42**	**82.13**	**87.24**
	Luz Nocturna	Video 1	69.37	76.92	77.19	80.89
		Video 2	71.71	75.29	76.57	81.29
		Video 3	70.18	72.73	74.99	80.67
		Media	**70.42**	**74.98**	**76.25**	**80.95**

Tabla 6.11: Resultados de especificidad obtenidos por los algoritmos replicados del estado del arte para los casos de prueba establecidos

Criterio	Caso de prueba	Video de prueba	(Töreyin et al. 2005)	(Vieira et al. 2016)	(Zhao et al. 2015)	Algoritmo Propuesto
Especificidad	Luz diurna	Video 1	88.80	85.54	76.47	86.08
		Video 2	96.51	92.97	83.12	93.56
		Video 3	94.20	90.74	81.13	91.32
		Media	**93.17**	**89.75**	**80.24**	**90.32**
	Día nublado	Video 1	87.29	87.57	82.47	85.50
		Video 2	91.51	91.79	85.45	89.62
		Video 3	92.04	92.32	87.95	90.14
		Media	**90.28**	**90.56**	**85.29**	**88.42**
	Luz Nocturna	Video 1	88.05	94.22	89.65	93.33
		Video 2	92.40	97.80	94.09	96.88
		Video 3	87.24	94.42	91.84	93.53
		Media	**89.23**	**95.48**	**91.86**	**94.58**

La sensibilidad resultante presenta una variación entre las condiciones de luz diurna y día nublado de 6.02% en el algoritmo de (Töreyin et al. 2005), de 7.95% en el algoritmo de (Vieira et al. 2016), y de 3.52% en el algoritmo de (Zhao et al. 2015). Estos valores contrastan con la variación de 0.96% del algoritmo propuesto.
Bajo condiciones de Luz Nocturna, el algoritmo de (Töreyin et al. 2005), obtuvo una variación de 15.66%; el de (Vieira et al. 2016) una variación de 14.77, y el de (Zhao et al. 2015), del 10.58%. Por otro lado, la herramienta propuesta presentó una variación de 7.28%.

En el primer caso, la mayor variación, presentada por el algoritmo de (Vieira et al. 2016), debe su magnitud tanto al entrenamiento como a la falta de un complemento para el análisis en espacio RGB. Éste algoritmo posee una baja tolerancia a los diferentes escenarios de iluminación. El algoritmo de (Zhao et al. 2015) reduce la variación al reemplazar el análisis de color con el de textura dinámica; sin embargo, no emplear características estáticas ocasiona que se encuentre con interferencia de otros objetos en movimiento. La variación entre Luz Diurna y Nocturna se incrementa debido a la mayor cantidad de falsos negativos ocasionados por la pérdida de variación entre la intensidad de los pixeles de humo y los de otros objetos. La combinación de características estáticas y dinámicas con el entrenamiento de AdaBoost empleado por la herramienta propuesta, sin el requerimiento de entrenar el análisis de color permiten reducir la variación en los mismos casos de prueba. Cabe resaltar que a la necesidad de un adecuado entrenamiento para AdaBoost se mantiene en la herramienta propuesta, y es pieza clave para mejorar los resultados obtenidos por las otras herramientas.

La Figura 6.19 Muestra la gráfica de barras de la variación de sensibilidad entre los distintos algoritmos bajo diferentes condiciones de iluminación.

La especificidad también presenta variaciones entre las distintas condiciones de iluminación. Bajo condiciones de día nublado, el algoritmo de Toreyin et al. Presentó un decremento de 3.22%, mientras que el algoritmo de Vieira et al. obtuvo un incremento de 1.11% y el algoritmo de Zhao et al, de 6.25%. La herramienta propuesta por otro lado, obtuvo un decremento de 2.11%.

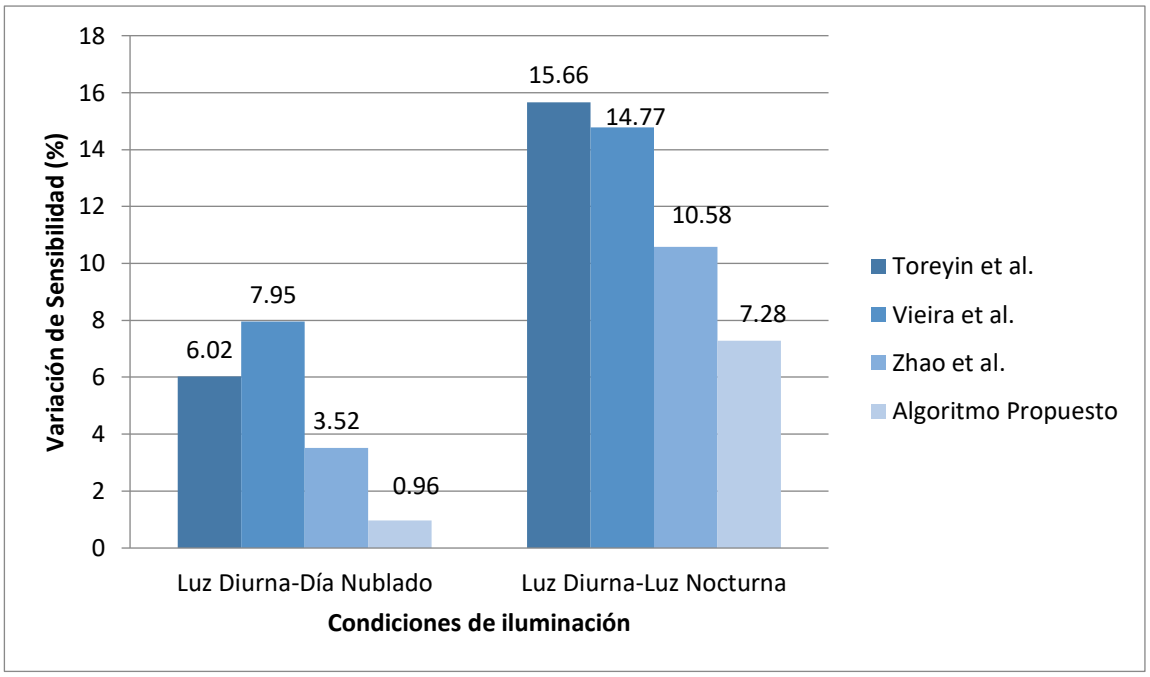

Figura 6.19: Gráfica comparativa de la variación de sensibilidad de cada algoritmo entre distintas condiciones de iluminación

En cuanto a las condiciones de Luz Nocturna, el algoritmo de Toreyin et al, resultó en una variación de 4.30% con respecto a la luz diurna; el algoritmo de Vieira et al, obtuvo un incremento de 6.66%. El algoritmo de Zhao et al, presentó la mayor variación con incremento de 15%. El algoritmo propuesto se incrementó en 3.87%.

En cuanto a la variación de especificidad entre los videos de Luz Diurna y los de Día Nublado, el algoritmo de Vieira reduce la cantidad de detecciones debido nuevamente a su alta dependencia al entrenamiento del análisis en RGB, y puesto que se evaluaron los dos estados con más iluminación, las diferencias entre el entrenamiento de un escenario y el otro son menores a las requeridas para luz nocturna. El algoritmo de (Töreyin et al. 2005), regula las variaciones generadas por sus etapas de análisis dinámico empleando el análisis de color en YUV, a diferencia del algoritmo de (Zhao et al. 2015) que presenta la mayor variación debido a que ninguna de sus etapas descarta zonas de la imagen que no poseen el color adecuado, es decir, la textura dinámica es detectada en algunos casos dentro del cielo nocturno, generando falsos positivos que afectan la métrica analizada.

La herramienta propuesta, por otro lado, hace uso del análisis de características dinámicas, descartando regiones de pixeles que no están en movimiento, y de características estáticas, que permiten ajustar los resultados de las etapas dinámicas y filtrar los pixeles de una mejor manera. Esto plantea una mejor solución que mantiene porcentajes similares en los criterios de exactitud analizados pero con una menor variación en distintos escenarios.

La Figura 6.20 muestra la gráfica comparativa de especificidad entre los algoritmos contrastados para los distintos casos de iluminación.

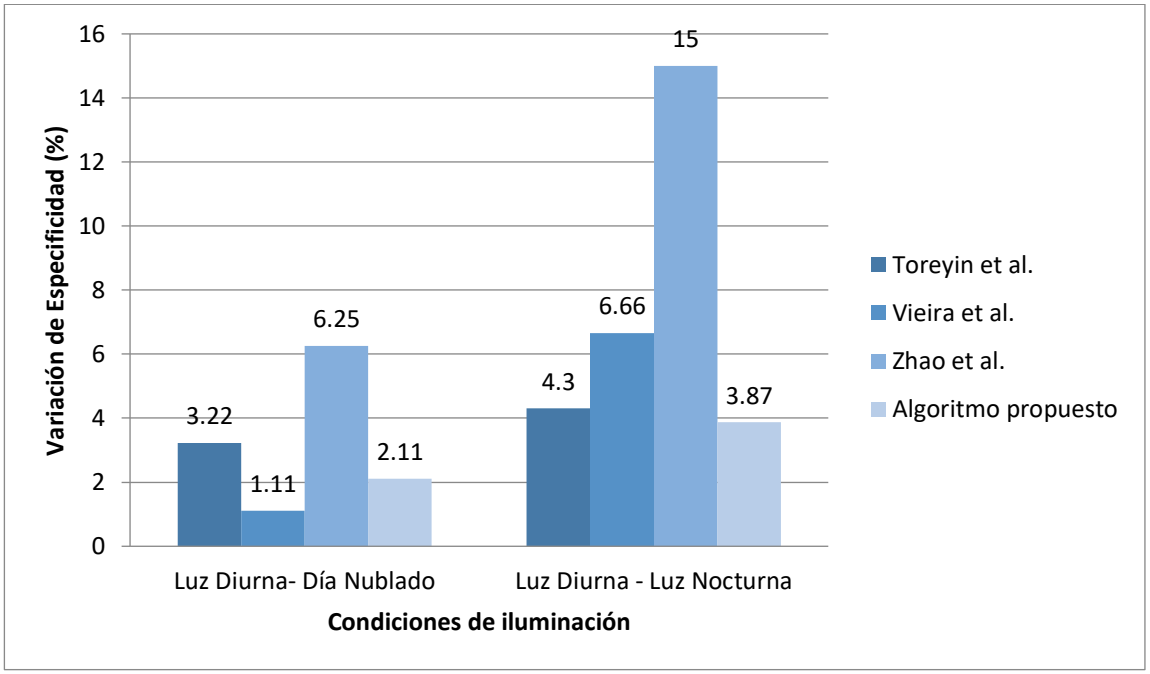

Figura 6.20: Gráfica comparativa de la variación de especificidad de cada algoritmo entre distintas condiciones de iluminación

Adicionalmente, se analizó el tiempo de ejecución de los algoritmos bajo condiciones de iluminación diurna, dado que es el caso de prueba en el que se encuentran la mayor cantidad de detecciones tanto de pixeles con humo como de regiones con movimiento que no son descartables por la etapa de detección de dicha característica. De manera similar al ejercicio de complejidad temporal descrito en la sección 5.1, se realizó la

ejecución de las pruebas con cantidades de cuadros establecidas. La tabla 6.11 muestra los resultados obtenidos.

Tabla 6.12: Tiempos de ejecución de los algoritmos analizados en pruebas con diferentes cantidades de cuadros de entrada.

Algoritmo	**200 cuadros**	**400 cuadros**	**600 cuadros**	**800 cuadros**	**1000 cuadros**	**1200 cuadros**	**1400 cuadros**
Toreyin et al.	34.59	83.97	112.48	156.56	184.35	221.63	254.24
Vieira et al.	259.84	618.18	906.65	941.24	987.65	992.48	1007.26
Zhao et al.	98.14	234.05	339.71	365.93	382.72	408.13	427.41
Algoritmo propuesto	30.21	72.54	100.32	125.03	172.42	217.25	239.46

La Figura 6.21 muestra un gráfico comparativo de los resultados obtenidos. La tendencia general de los algoritmos de (Vieira et al. 2016) y (Zhao et al. 2015) es a una complejidad proporcional logarítmica, con tiempos de ejecución más altos para el primero, mientras que el algoritmo de (Töreyin et al. 2005) y el algoritmo propuesto muestran una tendencia a una complejidad lineal, con un menor tiempo de ejecución en la herramienta propuesta. El ajuste previo de la resolución en los cuadros procesados, así como el uso de la etapa de detección de movimiento como filtro inicial para identificar las regiones de interés, permite que el algoritmo propuesto presente el menor tiempo de ejecución; cabe señalar que estos tiempos de ejecución no contemplan el pre-procesamiento realizado a los cuadros. La cantidad de operaciones de comparación requeridas por la etapa de análisis de color en el algoritmo de (Vieira et al. 2016) ocasiona los mayores tiempos de ejecución, sin embargo, delimitar los distintos tipos de humo a detectar, puede reducirlas, y disminuir el tiempo de ejecución del algoritmo. El algoritmo de (Zhao et al. 2015), presentó una complejidad y un tiempo de ejecución mayor al algoritmo propuesto debido principalmente a que cada etapa trabaja con la imagen original en lugar de filtrar las regiones de interés. El algoritmo de (Töreyin et al.

2005) muestra un comportamiento similar al algoritmo propuesto, con tiempos de ejecución más altos; la posibilidad de filtrar cada etapa con los resultados de la anterior, permite mantener una complejidad con tendencia al comportamiento lineal.

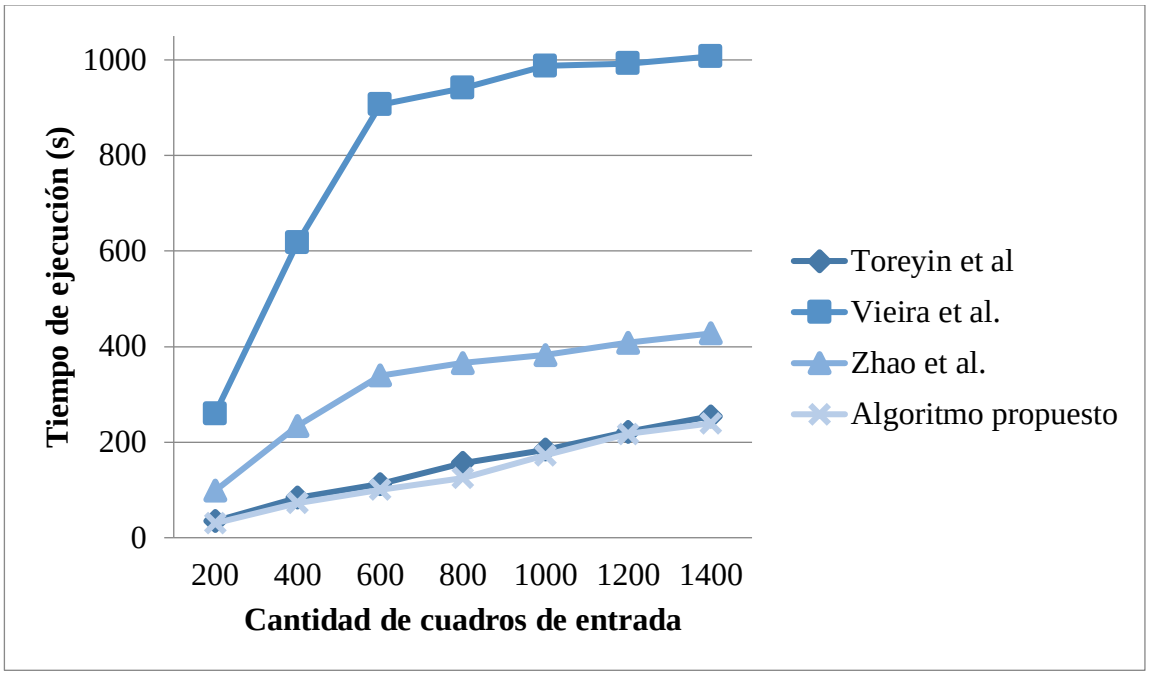

Figura 6.21: Gráfico comparativo de los tiempos de ejecución obtenidos por los algoritmos analizados en pruebas con cantidades diferentes de cuadros de entrada en condiciones de iluminación diurna

La adaptación del algoritmo propuesto a diferentes condiciones de iluminación se evaluó tanto en sensibilidad, donde se obtuvieron los menores porcentajes de variación entre las distintas condiciones de iluminación, como en especificidad dónde se presentó el menor porcentaje de variación en condiciones de iluminación nocturna, mientras que el algoritmo de Vieira et al presentó la menor variación en condiciones de día nublado.

De acuerdo con lo presentado en esta sección, la herramienta propuesta cumple con la tolerancia a cambios de iluminación ambiental que se propuso en la hipótesis, esto teniendo en cuenta que tanto la sensibilidad y la especificidad mantienen porcentajes que superan la misma propuesta para los tres casos de iluminación evaluados.

Como se mencionó previamente, se han detectado también distintas oportunidades de mejora para la propuesta, iniciando por buscar un mejor ajuste de los parámetros constantes que permita mantener la estabilidad lograda por el algoritmo y buscar la adaptación de otras herramientas que permitan mejorar la exactitud del algoritmo y reducir las posibles falsas alarmas u omisiones de alertas.

7. CONCLUSIONES

Las condiciones de iluminación generan diversos problemas cuando se intentan detectar objetos dentro de imágenes mediante herramientas de Visión Artificial. En el caso del humo, al no tener bordes definidos, los cambios en las condiciones atmosféricas y de iluminación complican aún más su identificación debido a la amplia variedad de combinaciones de intensidades que puede contener el humo en los canales de color.

Un cambio en la iluminación que influye sobre una escena ocasiona una variación en la luminancia relativa que está representada por el valor de intensidad capturado por una cámara. La detección de incendios debe permitir el monitoreo de escenas bajo diferentes condiciones climáticas y reducir en lo posible la aparición de falsos positivos en el proceso de análisis de escenas.

El trabajo presente se enfocó en proponer una herramienta con tolerancia a los cambios de iluminación; es decir, a diferentes grados de luminancia relativa. Se emplearon etapas para detección de características estáticas y dinámicas que permiten adaptar su funcionamiento a la información contenida en la secuencia de imágenes.

A partir del algoritmo propuesto, se realizó la evaluación para tres casos de iluminación, Luz Diurna como referencia de trabajo, Luz Nocturna como estado bajo de iluminación y Día Nublado como punto de iluminación intermedia. La exactitud de la herramienta, determinada por los criterios de sensibilidad y especificidad, mostró una menor variación respecto a la condición de luz diurna.

En contraste con tres algoritmos representativos del estado del arte que fueron replicados, la sensibilidad de la herramienta propuesta presentó el menor porcentaje de variación, con un 0.96% para la condición de Día Nublado, que representa una reducción del 72.7% en la variación con respecto al menor valor obtenido por los algoritmos replicados. En condiciones de Luz nocturna, se obtuvo una variación de 7.28%, que representa una reducción del 31.19% con respecto a la variación más baja obtenida por las herramientas replicadas. Aunado a lo anterior, el algoritmo propuesto conservó un porcentaje de sensibilidad superior al 80%, que se estableció como objetivo para las detecciones correctas de la propuesta.

La especificidad presentó variaciones del 2.11% y 3.87% para Día Nublado y Luz Nocturna respectivamente, mientras que para Luz Nocturna esto representa una reducción del 10% con respecto al porcentaje de variación más bajo de las herramientas replicadas. Para Día Nublado se obtuvo un incremento del 90% con respecto al valor más bajo replicado. Sin embargo, cabe resaltar que la herramienta que obtuvo el valor más bajo (la propuesta por Vieira et al. con 1.11%) presenta un comportamiento altamente dependiente en el entrenamiento de la herramienta RGB, la cual debe almacenar todas las combinaciones de color posibles. La Especificidad resultante supero el 88% para las tres condiciones evaluadas, cumpliendo con el objetivo de mantener el 85% de no-detecciones correctas.

A partir de lo anterior, es posible confirmar el cumplimiento de la hipótesis establecida, al conservar porcentajes de detecciones y no-detecciones superiores a los establecidos en la misma, y reducir en la mayoría de los casos la variación existente en el funcionamiento bajo diferentes condiciones de iluminación.

Adicionalmente, el tiempo de ejecución del algoritmo propuesto, sin contemplar el pre-procesamiento de las imágenes, mantuvo el menor tiempo de ejecución, con una complejidad temporal con tendencia lineal $O(n)$; es importante considerar que de cara a una posible implementación del algoritmo en un sistema real, existen distintas alternativas para mejorar el comportamiento y el tiempo de ejecución de la herramienta propuesta. Considerar el uso de procesamiento paralelo, o delimitar el tipo de humo a detectar para disminuir la cantidad de entrenamiento y operaciones necesarias de comparación, son las principales propuestas identificadas.

Por otro lado, cabe resaltar que la implementación de la herramienta propuesta fue realizada de manera experimental. Buscar una implementación real de un sistema para monitoreo y detección de incendios basado en la presente propuesta requiere la adaptación de las herramientas a un lenguaje de programación apropiado, así como de las adecuaciones que permitan reducir el tiempo de ejecución necesario para realizar las detecciones.

REFERENCIAS

Agrawal, D.A. y Mishra, P., 2014. Smoke Detection using Local Binary Pattern. International Journal of Current Engineering and Technology, Vol. 4(6), pp.4052–4056.

Besbes, O. y Benazza-Benyahia, A., 2016. A novel video-based smoke detection method based on color invariants. 2016 IEEE International Conference on Acoustics, Speech and Signal Processing, pp.1911–1915.

Brovko, N., Bogush, R. y Ablameyko, S., 2013. Smoke detection algorithm for intelligent video surveillance system. Computer Science Journal of Moldova, Vol. 21(1), pp.142–156.

Çetin, A.E. Dimitropoulos, K., Gouverneur, B., Grammalidis, N., Günay, O., Habiboglu, Y., Hakan, Toreyin, B.U. y Verstockt, S., 2013. Video fire detection – Review. Digital Signal Processing, Vol. 23, pp.1827–1843.

Çetin, A.E., Merci, B., Gunay, O., Toreyin, B.U. y Verstockt, S., 2016. Methods and Techniques for Fire Detection: Signal, Image and Video Processing Perspectives 1st ed., Londres: Elsevier B.V.

Cetin, E. (Bilkent U., 2003. Computer Vision Based Fire Detection Software. Available at: http://signal.ee.bilkent.edu.tr/VisiFire/.

Chen, H., Zhang, X., Hong, P., Hu, H. y Yin, X., 2016. Recognition of the Temperature Condition of a Rotary Kiln Using Dynamic Features of a Series of Blurry Flame Images. IEEE Trans. On Industrial Informatics, Vol. 12(1), pp.148–157.

Collins, R.T., Lipton, A.J. y Kanade, T., 2000. Introduction to the special section on video surveillance., VSAM Final Report.

Deldjoo, Y., Nazary, F. y Fotouhi, A.M., 2015. A Novel Fuzzy-Based Smoke Detection System Using Dynamic and Static Smoke Features. 23rd Iranian Conference on Electrical Engineering, pp.729–733.

Dimitropoulos, K., Barmpoutis, P. y Grammalidis, N., 2016. Higher Order Linear

Dynamical Systems for Smoke Detection in Video Surveillance Applications. IEEE Transactions on Circuits and Systems for Video Technology, pp.1–13.

Dukuzumuremyi, J.P., Zou, B. y Hanyurwimfura, D., 2014. A Novel Algorithm for Fire / Smoke Detection based on Computer- Vision. International Journal of Hybrid Information Technology, Vol. 7(3), pp.143–154.

Francis, A.B., 2016. ANFIS based Color Constancy Algorithms., 2016 International Conference on Next Generation Intelligent Systems.

Genovese, A., Labati, R.D., Piuri, V., y Scotti, F., 2011. Wildfire smoke detection using computational intelligence techniques. IEEE Transactions on System, Man, and Cybernetics, Vol. 43(4), pp.1–6.

Gottuck, D.T. y Dinaburg, J., 2012. Fire Detection in Warehouse Facilities, Baltimore: Hughes Associates, Inc.

Günay, O., 2015. Video processing algorithms for wildfire surveillance. Bilkent University.

Heras, D., 2017. Fruit image classifier based on artificial intelligence., Revista Killkana Tecnica, Vol. 1(2), pp.21–30.

Jinlan, L., Lin, W., Ruliang, Z., Chengquan, H. y Yan, R., 2016. A Method of Fire And Smoke Detection Based on Surendra Background And Gray Bitmap Plane Algorithm. 8th International Conference on Information Technology in Medicine and Education. pp. 1–5.

Kaur, H. y Sharma, S., 2016. A Comparative Review of Various Illumination Estimation Based Color Constancy Techniques., Int. Conf. on Communication and Signal Processing, pp.486–490.

Kim, D. y Wang, Y.F., 2009. Smoke detection in video., 2009 WRI World Congress on Computer Science and Information Engineering, CSIE 2009, Vol. 5, pp.759–763.

Kim, H., Ryu, D. y Park, J., 2014. Smoke Detection Using GMM and Adaboost., International Journal of Computer and Communication Engineering, Vol. 3(2),

pp.123–126.

Ko, B., Park, J. y Nam, J., 2013. Spatiotemporal bag-of-features for early wild fire smoke detection. Image and Vision Computing, Vol. 31(10), pp.786–795.

Labati, R.D., Genovese, A., Piuri, V. y Scotti, F., 2013. Wildfire Smoke Detection using Computational Intelligence Techniques Enhanced with Synthetic Smoke Plume Generation. IEEE Transactions on Sys, pp.1–11.

Luo, S., Yan, C., Wu, K. y Zheng, J., 2015. Smoke detection based on condensed image. Fire Safety Journal, Vol. 75, pp.23–35.

Borregón-Domènech, D.M., 2012. Sistema de detección de incendios forestales utilizando técnicas de procesado de imagen. Universidad Politecnica de Cataluña.

Muñoz, C., Acevedo, P., Salvo, S., Fagalde, G. y Vargas, F., 2007. Detección de incendios forestales utilizando imágenes NOAA / 16-LAC en la Región de La Araucanía, Chile. Bosque, Vol. 28(2), pp.119–128.

Nieto-González, J.L., Granda-Gutierrez, E.E., Pérez-Martínez, J.A. y Flores-Fuentes A.A., 2017. Evaluación de algoritmos para detección de humo mediante visión artificial bajo criterios de desempeño unificados. 39 Congreso Int. De Ing. Electronica.

Pandey, S. y Singh, A., 2014. Smoke and Fire Detection. International Journal of Engineering and Technical Research, Vol. 2(7), pp.26–29.

Pritam, D. y Dewan, J.H., 2017. Detection of fire using image processing techniques with LUV color space., 2nd Int. Conf. for Convergence in Technology. pp. 1158–1162.

Shuai, L., Bo, W., Ranran, D., Zhiqiang, Z. y Sun, L., 2016. A novel smoke detection algorithm based on Fast Self-tuning background subtraction. 28th Chinese Control and Decision Conference, pp.3539–3543.

Tek, F.B., Dempster, A.G. y Kale, İ., 2016. Adaptive Gray World-Based Color Normalization of Thin Blood Film Images. Available at:

http://arxiv.org/abs/1607.04032.

Töreyin, B.U., Yigithan, D. y Çetin, A.E., 2005. Wavelet Based Real-time Smoke Detection in Video. Proceedings of the European Signal Processings Conference, p.4.

Vala, M.H.J. y Baxi, A., 2013. A review on Otsu image segmentation algorithm., International Journal of Advanced Research in Computer Engineering and Technology, Vol. 2(2), pp.387–389.

Vieira, D.A.G., Santos A.L., Yehia, H.C., Lisboa, A.C. y Nascimento C.A.M., 2016. Smoke Detection in Environmental Regions by Means of Computer Vision., Combinations of Intelligent Methods and Applications, pp.135–151.

Wang, R., 2012. AdaBoost for Feature Selection, Classification and Its Relation with SVM, A Review. Physics Procedia, Vol. 25, pp.800–807.

Wong, A.K.K. y Fong, N.K., 2014. Study of pool fire heat release rate using video fire detection., Int. High Performance Buildings Conf. pp.1–15.

Wu, S., Yuan, F., Yang, Y., Fang Z. y Fang, Y, 2015. Real-time image smoke detection using staircase searching-based dual threshold AdaBoost and dynamic analysis. IET Image Processing, Vol. 9(10), pp.849–856.

Zhao, Y., Zhou, Z. y Xu, M., 2015. Forest Fire Smoke Video Detection Using Spatiotemporal and Dynamic Texture Features. Journal of Electrical and Computer Engineering, 2015, pp.1–7.

ANEXOS

Anexo 1

Código de implementación de la réplica del algoritmo de (Toreyin et al. 2005)

```
clear;
clc;
%Background estimation Method
t1=clock;
for frame=1:2
tau=0.07;
tau2=0.1;
a=0.1;
Result=0;
CarpetaImagenes=fullfile('D:\MACSCO\Proyecto',datestr(now,'yy-mm-dd'));
CarpetaSalida =
fullfile('D:\MACSCO\Proyecto',datestr(now,'yy'));%Designacion de
carpeta para cuadros
if ~exist(CarpetaSalida,'dir')
   mkdir(CarpetaSalida);%Creacion de carpeta de salida; si no existe
end
LabelRegion=1;
imagen=im2double(imread(fullfile(CarpetaImagenes,sprintf('%3.3d.png',fr
ame))));
imagenYCbCr=rgb2ycbcr(imagen);
imagenY=imagenYCbCr(:,:,1);
imagenU=imagenYCbCr(:,:,2);
imagenV=imagenYCbCr(:,:,3);
ImagenCMBinaria=imagenY*0;
[M,N] = size(imagenY);
if frame==1
    background=imagenY;
    ImagenCMBinaria=imagenY*0;
elseif frame==2
    for k=1:M
        for l=1:N
            if abs(background(k,l)-imagenY(k,l))>tau
                %pixel con movimiento
                backgroundNmas1(k,l)=background(k,l);
                ImagenCMBinaria(k,l)=1;
            else
                backgroundNmas1(k,l)=a*background(k,l)+(1-
a)*imagenY(k,l);
                ImagenCMBinaria(k,l)=0;
            end
        end
    end
else
    background=backgroundNmas1;
    for k=1:M
        for l=1:N
            if abs(background(k,l)-imagenY(k,l))>tau &&
abs(background(k,l)-imagenY(k,l))<tau2
                %pixel con movimiento
```

```
                backgroundNmas1(k,l)=background(k,l);
                ImagenCMBinaria(k,l)=1;
            else
                backgroundNmas1(k,l)=a*background(k,l)+(1-
a)*imagenY(k,l);
                ImagenCMBinaria(k,l)=0;
            end
        end
    end
end
ImagenCCA = bwlabel(ImagenCMBinaria);
%%Paso ii
imagenEW=imagenY;
imagenVDW=imagenY;
thres1=0.08;
thres2=0.1;
T1=0.0155;
T2=0.0115;
[cA1,cH1,cV1,cD1] = dwt2(imagenY,'haar');
sx=size(imagenY);
LH = idwt2([],cV1,[],[],'haar',sx)
HL = idwt2([],cH1,[],[],'haar',sx);
HH = idwt2([],cD1,[],[],'haar',sx);

[cA2,cH2,cV2,cD2] = dwt2(background,'haar');
LHb = idwt2([],cV2,[],[],'haar',sx);
HLb = idwt2([],cH2,[],[],'haar',sx);
HHb = idwt2([],cD2,[],[],'haar',sx);

for x=1:M
    for y=1:N
        Wn(x,y)=abs(LH(x,y))^2+abs(HL(x,y))^2+abs(HH(x,y))^2;
        Wbn(x,y)=abs(LHb(x,y))^2+abs(HLb(x,y))^2+abs(HHb(x,y))^2;
    end
end
bloquesv=M/8;
bloquesh=N/8;
for l1=0:bloquesv-1
    for l2=0:bloquesh-1
        energiaBloque=0;
        energiaBloqueb=0;
        for x=1:8
            for y=1:8
                energiaBloque=energiaBloque+Wn(x+l1*8,y+l2*8);
                energiaBloqueb=energiaBloqueb+Wbn(x+l1*8,y+l2*8);
            end
        end
        for x=1:8
            for y=1:8
                if energiaBloqueb-energiaBloque>thres1&&energiaBloqueb-
energiaBloque<thres2
                    imagenEW(x+l1*8,y+l2*8)=1;
                else
                    imagenEW(x+l1*8,y+l2*8)=0;
                end
            end
```

```
        end
    end
end
frame;
for x=1:M
    for y=1:N
        if T1*Wbn(x,y)>Wn(x,y) && Wn(x,y)>T2*Wbn(x,y)
            imagenVDW(x,y)=1;
        else
            imagenVDW(x,y)=0;
        end
    end
end
NombreArchivo = sprintf('%3.3d.png',frame); %nombre del archivo
FullName = fullfile(CarpetaSalida, NombreArchivo);
imwrite (imagenEW, FullName, 'png'); %creación del archivo
imshow(imagenEW);
memory;
end
t2=clock;
t=etime(t2,t1)
```

Anexo 2

Código de implementación de la réplica del algoritmo de (Vieira et al. 2016)

```
clear
clc
z=1;
ImRegionesP(:,:)=0;
numP=0;
t1=clock;
for frame=1:550

    Result=0;
    %Background removal
    alfa=0.09;
    tau=0.1;
    CarpetaImagenes=fullfile('D:\MACSCO\Proyecto',datestr(now,'yy-mm-
dd'));

imagenRGB=imread(fullfile(CarpetaImagenes,sprintf('%3.3d_N.png',frame))
);
    imagen=im2double(imagenRGB);
    imagenGris=rgb2gray(imagen);
    Ai=imagenGris*0;
    [M,N] = size(imagenGris);
    if frame==1
        Bi=imagenGris;
    end
    for x=1:M
        for y=1:N
            if abs(imagenGris(x,y)-Bi(x,y))>tau
                Ai(x,y)=1;
            else
```

```
                Ai(x,y)=0;
            end
            if frame==2
                Bi(x,y)=imagenGris(x,y);
            elseif frame>2
                Bi(x,y)=(1-alfa)*Bi(x,y)+alfa*imagenGris(x,y);
            end
        end
    end
    %%
    %Clasificacion de color
    ColorM = csvread('colores2.csv');
    [colorMSize,rgb]=size(ColorM);
    for x=1:M
        for y=1:N
          if Ai(x,y)==1&&abs(imagenRGB(x,y,1)-
imagenRGB(x,y,2))<=10&&abs(imagenRGB(x,y,1)-
imagenRGB(x,y,3))<=10&&abs(imagenRGB(x,y,2)-imagenRGB(x,y,3))<=10
                Ai2(x,y)=1;
            else
                Ai2(x,y)=0;
            end
        end
    end

    %%
    %Spatial/temporal Persistence
   % if frame==1
        Ai3=Ai2*0;
   % end
    fps=15;
    totalframes=30;
    if frame==1
        [ImRegiones,num] = bwlabel(Ai2,8);
    elseif frame>1
        ImRegionesP=ImRegiones;
        numP=num;
        [ImRegiones,num] = bwlabel(Ai2,8);
        for region=1:num
            [r, c] = find(bwlabel(ImRegiones)==region);
            [rP, cP] = find(bwlabel(ImRegionesP)==region);
            ImRegion = bwselect(ImRegiones,c,r,8);
            ImRegionP = bwselect(ImRegionesP,c,r,8);
            ImR=ImRegion.*ImRegionP;
            pixeles=0;
            for k=1:M
                for l=1:N
                    if ImR(k,l)==1
                      pixeles=pixeles+1;
                    end
                end
            end
            if pixeles>8
                Ai3=Ai3+ImR;
            end
        end
        for i=1:M
```

```
            for j=1:N
                if Ai3(i,j)==1
                    Result=1;
                end
            end
        end
        %imshow(Ai3);
    end

   % imshow(Ai3)
    %%
    %Guardar imagen resultado
    if Result==1
        CarpetaSalida =
fullfile('D:\MACSCO\Proyecto\Resultados\Vieira',datestr(now,'yy-mm-
dd'));%Designacion de carpeta para cuadros
        if ~exist(CarpetaSalida,'dir')
            mkdir(CarpetaSalida);%Creacion de carpeta de salida; si no
existe
        end
        NombreArchivo = sprintf('%3.3d.bmp',frame); %nombre del archivo
        FullName = fullfile(CarpetaSalida, NombreArchivo);
        imwrite (imagenRGB, FullName, 'bmp'); %creación del archivo
    end
    %t2=clock;
%t(frame)=etime(t2,t1)
memory
end
t2=clock;
t=etime(t2,t1)
```

Anexo 3

Código de implementación de la réplica del algoritmo de (Zhao et al. 2015)

```
clear;
clc;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Paso
i%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Background estimation Method
t1=clock;

for frame=1:N
%t1=clock;
tau=0.07;
tau2=0.1;
a=0.1;
Result=0;
CarpetaImagenes=fullfile('D:\MACSCO\Proyecto',datestr(now,'yy-mm-dd'));
CarpetaSalida =
fullfile('D:\MACSCO\Proyecto',datestr(now,'yy'));%Designacion de
carpeta para cuadros
if ~exist(CarpetaSalida,'dir')
   mkdir(CarpetaSalida);%Creacion de carpeta de salida; si no existe
```

```
end
imagen=im2double(imread(fullfile(CarpetaImagenes,sprintf('%3.3d.png',fr
ame))));
imagenGris=rgb2gray(imagen);
[M,N] = size(imagenGris);
if frame==1
    background=imagenGris;
    ImagenBinaria=imagenGris*0;
elseif frame==2
    for k=1:M
        for l=1:N
            if abs(background(k,l)-imagenGris(k,l))>tau
                %pixel con movimiento
                backgroundNmas1(k,l)=background(k,l);
                ImagenBinaria(k,l)=1;
            else
                backgroundNmas1(k,l)=a*background(k,l)+(1-
a)*imagenY(k,l);
                ImagenBinaria(k,l)=0;
            end
        end
    end
else
    background=backgroundNmas1;
    for k=1:M
        for l=1:N
            if abs(background(k,l)-imagenGris(k,l))>tau &&
abs(background(k,l)-imagenGris(k,l))<tau2
                %pixel con movimiento
                backgroundNmas1(k,l)=background(k,l);
                ImagenBinaria(k,l)=1;
            else
                backgroundNmas1(k,l)=a*background(k,l)+(1-
a)*imagenGris(k,l);
                ImagenBinaria(k,l)=0;
            end
        end
    end
end
%%
%Consistencia espacio-temporal
imagenBinaria2=imagenGris;
thres1=0.08;
thres2=0.1;
[cA1,cH1,cV1,cD1] = dwt2(imagenY,'haar');
sx=size(imagenGris);
LH = idwt2([],cV1,[],[],'haar',sx)
HL = idwt2([],cH1,[],[],'haar',sx);
HH = idwt2([],cD1,[],[],'haar',sx);

[cA2,cH2,cV2,cD2] = dwt2(background,'haar');
LHb = idwt2([],cV2,[],[],'haar',sx);
HLb = idwt2([],cH2,[],[],'haar',sx);
HHb = idwt2([],cD2,[],[],'haar',sx);

for x=1:M
```

```
    for y=1:N
        Wn(x,y)=abs(LH(x,y))^2+abs(HL(x,y))^2+abs(HH(x,y))^2;
        Wbn(x,y)=abs(LHb(x,y))^2+abs(HLb(x,y))^2+abs(HHb(x,y))^2;
    end
end
bloquesv=M/8;
bloquesh=N/8;
for l1=0:bloquesv-1
    for l2=0:bloquesh-1
        energiaBloque=0;
        energiaBloqueb=0;
        for x=1:8
            for y=1:8
                energiaBloque=energiaBloque+Wn(x+l1*8,y+l2*8);
                energiaBloqueb=energiaBloqueb+Wbn(x+l1*8,y+l2*8);
            end
        end
        for x=1:8
            for y=1:8
                if energiaBloqueb-energiaBloque>thres1&&energiaBloqueb-
energiaBloque<thres2
                    imagenBinaria2(x+l1*8,y+l2*8)=1;
                else
                    imagenBinaria2(x+l1*8,y+l2*8)=0;
                end
            end
        end
    end
end

%%
%Detección de direccion del movimiento
for i=0:320
    for j=0:240
        if imagenBinaria(i,j)==1 && r==0
            s1=i;
            s2=j;
        elseif imagenBinaria(i,j)==1 && r==1
            s3=i;
            s4=j;
        end
    end
    subIm=im(s1,s2,s3,s4,:);
    m00=sum(subIM);
    for i=0:s3
        m01=sum(i*subIM);
    end
    for i=0:s4
        m10=sum(i*subIM);
    end
    cx=m10/m00;
    cy=m01/m00;
    ang=acos(cx/sqrt(pow(cx,2)+pow(cy,2)));
    if ang>0
        mapaBinario3=1;
    else
        mapaBinario3=0;
```

```
    end
end
%%
%AdaBoost
Pesos = csvread('VectEntrenamiento.csv');
if sum(imagenBinaria)>64
    c1=1;
else
    c1=0;
end
if sum(imagenBinaria2)>64
    c2=1;
else
    c2=0;
end
if sum(imagenBinaria3)>64
    c3=1;
else
    c3=0;
end
if sum(imagenBinaria4)>64
    c4=1;
else
    c4=0;
end
SC=c1*Pesos(0)+c2*Pesos(1)+c3*Pesos(2)+c4*Pesos(3);
if SC>0
    deteccion=1;
else
    deteccion=0;
end
end
```

Anexo 4

Código correspondiente a la etapa de pre-procesamiento

```
video = VideoReader('D:\MACSCO\Proyecto\IMG_1144.mov');%Lectura de
archivo de video
CarpetaSalida = fullfile('D:\MACSCO\Proyecto',datestr(now,'yy-mm-
dd'));%Designacion de carpeta para cuadros
if ~exist(CarpetaSalida,'dir')
   mkdir(CarpetaSalida);%Creacion de carpeta de salida; si no existe
end
duracion = video.Duration;%Duracion del video
FRate = video.FrameRate;%Cuadros por segundo del video
Cuadros = duracion*FRate;
MuestraActual = 1;%Muestra inicial
ind=1;
%imshow(ImagenCMBinaria)
for i = 1:Cuadros %Ciclo for para generacion de muestras
    Cuadro = read(video,MuestraActual); %Lectura de cuadros
    Cuadro = imresize(Cuadro,[240 320]);%Cambio de resolución
    if ind==1
        Ir=MAX(Cuadro(:,:,1));
```

```
        Ig=MAX(Cuadro(:,:,2));
        Ib=MAX(Cuadro(:,:,3));
        Cuadro=Cuadro/MAX(Ir,Ig,Ib);
    elseif i==1
        med=mean(im2gray(Cuadro));
    else
        med2=mean(im2gray(Cuadro));
        dif=med-med2;
        Cuadro=Cuadro+dif;
    end
    NombreArchivo = sprintf('%3.3d.png',i); %nombre del archivo
    FullName = fullfile(CarpetaSalida, NombreArchivo);
    imwrite (Cuadro, FullName, 'png'); %creación del archivo
    MuestraActual = MuestraActual + 1; %avance del contador de muestras
end
```

Anexo 5

Código de la etapa de detección de movimiento

```
for frame=1:N
    Result=0;
    %Background removal
    alfa=0.09;
    tau=0;
    imagenRGB=Cuadro;
    imread(fullfile(CarpetaImagenes,sprintf('%3.3d_N.png',frame)));
    imagen=im2double(imagenRGB);
    imagenGris=rgb2gray(imagen);
    for j=0:255
        p(j)=conteo(imagenGris,j)/sum(imagenGris);
    end
    for j=0:255
         for k=0:j
             s1=s1+p(k);
             m1=m1+k*p(k);
         end
         for k=j:255
             s2=s2+p(k);
             m2=m2+k*p(k);
         end
         sig2=s1*s2*pow(m1-m2,2);
         if sig2>tau
             tau=sig2;
         end
    end
    Ai=imagenGris*0;
    [M,N] = size(imagenGris);
    if frame==1
        Bi=imagenGris;
    end
    for x=1:M
        for y=1:N
            if abs(imagenGris(x,y)-Bi(x,y))>tau
```

```
                Ai(x,y)=1;
            else
                Ai(x,y)=0;
            end
            if frame==2
                Bi(x,y)=imagenGris(x,y);
            elseif frame>2
                Bi(x,y)=(1-alfa)*Bi(x,y)+alfa*imagenGris(x,y);
            end
        end
    end
end
```

Anexo 6

Código de la etapa de análisis en espacio de Wavelets

```
MapaBinario=imagenGris;
[cA1,cH1,cV1,cD1] = dwt2(imagenY,'db4');
sx=size(imagenY);
LH = idwt2([],cV1,[],[],'db4',sx)
HL = idwt2([],cH1,[],[],'db4',sx);
HH = idwt2([],cD1,[],[],'db4',sx);

[cA2,cH2,cV2,cD2] = dwt2(background,'db4');
LHb = idwt2([],cV2,[],[],'db4',sx);
HLb = idwt2([],cH2,[],[],'db4',sx);
HHb = idwt2([],cD2,[],[],'db4',sx);

for x=1:M
    for y=1:N
        Wn(x,y)=abs(LH(x,y))^2+abs(HL(x,y))^2+abs(HH(x,y))^2;
        Wbn(x,y)=abs(LHb(x,y))^2+abs(HLb(x,y))^2+abs(HHb(x,y))^2;
    end
end
thres1=otsu(Wn,1);
thres2=otsu(Wbn,2);
bloquesv=M/8;
bloquesh=N/8;
for l1=0:bloquesv-1
    for l2=0:bloquesh-1
        energiaBloque=0;
        energiaBloqueb=0;
        for x=1:8
            for y=1:8
                energiaBloque=energiaBloque+Wn(x+l1*8,y+l2*8);
                energiaBloqueb=energiaBloqueb+Wbn(x+l1*8,y+l2*8);
            end
        end
        for x=1:8
            for y=1:8
                if energiaBloqueb-energiaBloque>thres1&&energiaBloqueb-
energiaBloque<thres2
                    MapaBinario(x+l1*8,y+l2*8)=1;
```

```
                else
                    MapaBinario(x+l1*8,y+l2*8)=0;
                end
            end
        end
    end
end
```

Anexo 7

Código de la etapa de detección de dirección del movimiento

```
for i=0:320
    for j=0:240
        if MapaBinaria(i,j)==1 && r==0
            s1=i;
            s2=j;
        elseif MapaBinaria(i,j)==1 && r==1
            s3=i;
            s4=j;
        end
    end
    subIm=im(s1,s2,s3,s4,:);
    m00=sum(subIM);
    for i=0:s3
        m01=sum(i*subIM);
    end
    for i=0:s4
        m10=sum(i*subIM);
    end
    cx=m10/m00;
    cy=m01/m00;
    ang=acos(cx/sqrt(pow(cx,2)+pow(cy,2)));
    if ang>0
        mapaBinario3=1;
    else
        mapaBinario3=0;
    end
end
```

Anexo 8

Código de la etapa de análisis de color

```
[colorMSize,rgb]=size(ColorM);
    for x=1:M
        for y=1:N
            if Ai(x,y)==1&&abs(imagenRGB(x,y,1)-
imagenRGB(x,y,2))<=10&&abs(imagenRGB(x,y,1)-
imagenRGB(x,y,3))<=10&&abs(imagenRGB(x,y,2)-imagenRGB(x,y,3))<=10
                Ai2(x,y)=1;
            else
```

```
                Ai2(x,y)=0;
            end
        end
    end
    %%imagenYCbCr=rgb2ycbcr(imagen);
imagenR=imagen(:,:,1);
imagenG=imagen(:,:,2);
imagenB=imagen(:,:,3);
imagenBinaria=imagenR;
[M,N] = size(imagenR);
for k=1:M
        for l=1:N
            if abs(imagenR(k,l)-imagenG(k,l))<=Th && abs(imagenG(k,l)-
imagenB(k,l))<=Th && abs(imagenR(k,l)-imagenB(k,l)<=Th)
                imagenBinaria(k,l)=255;
            else
                imagenBinaria(k,l)=0;
            end
        end
end
```

Anexo 9

Código de la etapa de toma de decisiones mediante el algoritmo AdaBoost

```
Pesos = csvread('VectEntrenamiento.csv');
if sum(imagenBinaria)>64
    c1=1;
else
    c1=0;
end
if sum(imagenBinaria2)>64
    c2=1;
else
    c2=0;
end
if sum(imagenBinaria3)>64
    c3=1;
else
    c3=0;
end
if sum(imagenBinaria4)>64
    c4=1;
else
    c4=0;
end
SC=c1*Pesos(0)+c2*Pesos(1)+c3*Pesos(2)+c4*Pesos(3);
if SC>0
    deteccion=1;
else
    deteccion=0;
end
```

Anexo 10

Código de la interfaz gráfica de usuario.

```
function varargout = Deteccion_Humo(varargin)
% DETECCION_HUMO MATLAB code for Deteccion_Humo.fig
%      DETECCION_HUMO, by itself, creates a new DETECCION_HUMO or
raises the existing
%      singleton*.
%
%      H = DETECCION_HUMO returns the handle to a new DETECCION_HUMO or
the handle to
%      the existing singleton*.
%
%      DETECCION_HUMO('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in DETECCION_HUMO.M with the given input
arguments.
%
%      DETECCION_HUMO('Property','Value',...) creates a new
DETECCION_HUMO or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before Deteccion_Humo_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Deteccion_Humo_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Deteccion_Humo

% Last Modified by GUIDE v2.5 26-Sep-2018 19:08:16

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Deteccion_Humo_OpeningFcn, ...
                   'gui_OutputFcn',  @Deteccion_Humo_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
```

```
end
% End initialization code - DO NOT EDIT


% --- Executes just before Deteccion_Humo is made visible.
function Deteccion_Humo_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Deteccion_Humo (see VARARGIN)

% Choose default command line output for Deteccion_Humo
handles.output = hObject;
imagen= imread('D:\MACSCO\Proyecto\17-11-09\862.png');
imshow(imagen,'Parent',handles.axes1);
imagen= imread('D:\MACSCO\Proyecto\17_e1\862.png');
imshow(imagen,'Parent',handles.axes2);
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Deteccion_Humo wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Deteccion_Humo_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[handles.archivo, handles.ruta]=uigetfile;
guidata(hObject,handles);
if handles.archivo
    set(handles.text4,'String',handles.archivo);
    set(handles.popupmenu3,'Enable','on');
    set(handles.pushbutton2,'Enable','on');
end


% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
str = get(hObject, 'String');
val = get(hObject, 'Value');
switch str{val};
    case 'Prestablecido'
        handles.tipo = 1;
    case 'Nuevo'
        handles.tipo = 2;
end
guidata(hObject, handles)
% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu2
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu2


% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in popupmenu3.
function popupmenu3_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
str = get(hObject, 'String');
val = get(hObject, 'Value');
switch str{val};
    case 'Video con humo'
        handles.etiqueta = 1;
    case 'Video sin humo'
        handles.etiqueta = 2;
end
guidata(hObject, handles)
% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu3
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu3


% --- Executes during object creation, after setting all properties.
function popupmenu3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu3 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
handles.etiqueta=1;
guidata(hObject, handles);
% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.pushbutton1,'Enable','off');
set(handles.popupmenu3,'Enable','off');
set(handles.pushbutton2,'Enable','off');
set(handles.pushbutton3,'Enable','on');
handles.video = VideoReader(strcat(handles.ruta,'\',handles.archivo));
handles.res1 = handles.video.Width;
handles.res2 = handles.video.Height;
handles.duracion = handles.video.Duration;
handles.FRate = handles.video.FrameRate;
handles.Cuadros = handles.duracion*handles.FRate;
guidata(hObject,handles);
set(handles.text10,'String',strcat(int2str(handles.res1),' x
',int2str(handles.res2)));
set(handles.text11,'String',round(handles.FRate));
minutos=floor(handles.duracion/60);
if minutos<10
    durMin=strcat('0',int2str(minutos));
else
    durMin=int2str(minutos);
end
segundos=handles.duracion-(minutos*60);
if segundos<10
    durSeg=strcat('0',int2str(segundos));
else
    durSeg=int2str(segundos);
end
set(handles.text12,'String',strcat(durMin,':',durSeg));
%MuestraActual = 1;
%direccion=strcat('C:\Smoke\temp\',handles.archivo);
[direc,nomb,extencion]=fileparts(handles.archivo);
handles.CarpetaTemp= fullfile('C:\Smoke\temp',datestr(now,'yy-mm-
dd'),nomb);%Designacion de carpeta para cuadros
 if ~exist(handles.CarpetaTemp,'dir')
     mkdir(handles.CarpetaTemp);%Creacion de carpeta de salida; si no
existe
 end
 for i=1:handles.Cuadros
     drawnow;
```

```
     if get(handles.pushbutton3,'UserData')==1
          set(handles.pushbutton1,'Enable','on');
          set(handles.popupmenu3,'Enable','on');
          set(handles.pushbutton2,'Enable','on');
          set(handles.pushbutton3,'Enable','off');
          set(handles.popupmenu4,'Enable','on');
          set(handles.pushbutton5,'Enable','on');
          handles.Cuadros=i;
          set(handles.pushbutton3,'UserData',0);

set(handles.text15,'String',get(handles.pushbutton3,'UserData'));
          break;
     else
          Cuadro = read(handles.video,i); %Lectura de cuadros
          Frame=Preprocesamiento(Cuadro);
          imshow(Frame,'Parent',handles.axes1);
          if handles.etiqueta==1
              NombreArchivo = sprintf('%3.3d_S.jpg',i); %nombre del
archivo
          else
              NombreArchivo = sprintf('%3.3d_N.jpg',i); %nombre del
archivo
          end
          FullName = fullfile(handles.CarpetaTemp, NombreArchivo);
          imwrite (Frame, FullName, 'jpg');
          set(handles.text13,'String',i);

set(handles.text14,'String',strcat(int2str(floor((i/handles.Cuadros)*10
0)),'%'));
          pause(0.00000001);
     end
 end
 set(handles.pushbutton1,'Enable','on');
 set(handles.popupmenu3,'Enable','on');
 set(handles.pushbutton2,'Enable','on');
 set(handles.pushbutton3,'Enable','off');
 set(handles.popupmenu4,'Enable','on');
 set(handles.pushbutton5,'Enable','on');

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.pushbutton3,'UserData',1);
set(handles.text15,'String',get(handles.pushbutton3,'UserData'));
%guidata(hObject,handles);



% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over
pushbutton3.
function pushbutton3_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
set(handles.pushbutton3,'UserData',1);
set(handles.text15,'String',get(handles.pushbutton3,'UserData'));


% --- Executes on selection change in popupmenu4.
function popupmenu4_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu4
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu4


% --- Executes during object creation, after setting all properties.
function popupmenu4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
for i=1:handles.Cuadros
    CarpetaImagenes = handles.CarpetaTemp;
    if handles.etiqueta==1

Cuadro=im2double(imread(fullfile(CarpetaImagenes,sprintf('%3.3d_S.jpg',
i))));
    else

Cuadro=im2double(imread(fullfile(CarpetaImagenes,sprintf('%3.3d_N.jpg',
i))));
    end
    ImagenMov=movimiento(Cuadro,i);
    ImagenDir=direccion(ImagenMov,i);
    ImagenWav=AnalisisWavelet(ImagenMov,i);
    ImagenRGB=AnalisisRGB(ImagenMov,i);
    ImagenYCbCr=AnalisisYCbCr(ImagenMov,i);

ImagenSalida=AdaBoostMB(ImagenMov,ImagenDir,ImagenWav,ImagenRGB,ImagenY
CbCr,i);
```

```
    switch handles.salida
        case 1:
           imshow(ImagenMov,'Parent',handles.axes2);
           break;
        case 2:
            imshow(ImagenDir,'Parent',handles.axes2);
            break;
        case 3:
            imshow(ImagenWav,'Parent',handles.axes2);
            break;
        case 4:
            imshow(ImagenRGB,'Parent',handles.axes2);
            break;
        case 5:
            imshow(ImagenYCbCr,'Parent',handles.axes2);
            break;
        case 6:
            imshow(ImagenSalida,'Parent',handles.axes2);
            break;
    end

Deteccion=AdaBoost(ImagenMov,ImagenDir,ImagenWav,ImagenRGB,ImagenYCbCr,
i);
    if Deteccion==1
        handles.permanencia=handles.permanencia+1;
    else
        handles.permanencia=0;
    end
    if handles.permanencia==30
        set(handles.text25,'String','Encendido');
        set(handles.pushbutton9,'Enable','on');
    else
        set(handles.text25,'String','Apagado');
        set(handles.pushbutton7,'Enable','off');
    end
end


% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.pushbutton6,'UserData',1);


% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.permanencia=0;
set(handles.text25,'String','Apagado');
set(handles.pushbutton7,'Enable','off');
```