ESTUDIO DE PERFILES DE COMPORTAMIENTO HUMANO EN SECUENCIAS DE VIDEO CON APLICACIONES DE SEGURIDAD

CARLOS EDUARDO GUEVARA NICHOY
JUAN DAVID VARGAS ALVARADO

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
Bogotá D.C.
2012

ESTUDIO DE PERFILES DE COMPORTAMIENTO HUMANO EN SECUENCIAS DE VIDEO CON APLICACIONES DE SEGURIDAD

CARLOS EDUARDO GUEVARA NICHOY
JUAN DAVID VARGAS ALVARADO

Trabajo de grado presentado como requisito parcial para optar al título de Ingeniero Electrónico

Director
Ing. CÉSAR LEONARDO NIÑO BARRERA PhD.

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
Bogotá D.C.
2012

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA ELECTRÓNICA

RECTOR MAGNÍFICO	**P. JOAQUÍN EMILIO SÁNCHEZ GARCÍA S.J.**
DECANO ACADÉMICO	**Ing. LUIS DAVID PRIETO MARTÍNEZ PhD.**
DECANO DEL MEDIO UNIVERSITARIO	**P. SERGIO BERNAL S.J.**
DIRECTOR DE CARRERA	**Ing. JAIRO ALBERTO HURTADO LONDOÑO PhD.**
DIRECTOR DEL PROYECTO	**Ing. CÉSAR LEONARDO NIÑO BARRERA PhD.**

ARTICULO 23 DE LA RESOLUCIÓN No. 13 DE JUNIO DE 1946

"La universidad no se hace responsable de los conceptos emitidos por sus alumnos en sus proyectos de grado.

Sólo velará porque no se publique nada contrario al dogma y la moral católica y porque los trabajos no contengan ataques o polémicas puramente personales. Antes bien, que se vea en ellos el anhelo de buscar la verdad y la justicia".

Quiero agradecer infinitamente a mis padres, Piedad Alicia Nichoy Martínez y Luis Eduardo Guevara Obando ya que gracias a su esfuerzo, ejemplo, apoyo, consejo, educación y amor termino satisfactoriamente esta etapa de mi vida, y de la misma manera les dedico este trabajo ya que todo se lo debo a ellos. A Ángela Sofía Esparza Albornoz, mi consentida hermosa, quiero dedicarle especialmente este trabajo ya que estuvo pendiente y contribuyo con su ayuda y consejo al desarrollo de éste, me resta agradecerle por estar apoyándome incondicionalmente en toda mi vida, llenándola de luz y de amor, te amo demasiado mi linda!

Asimismo agradezco al ingeniero Cesar Niño por su ayuda y acompañamiento incondicional durante el desarrollo de este trabajo y a mi amigo y compañero de trabajo de grado Juan David Vargas Alvarado por el compromiso con éste, para conseguir juntos los resultados deseados. También dedico este logro a toda mi familia y amigos que siempre estuvieron me apoyaron durante toda la carrera y a una persona en especial, a Daniel Alejandro Nichoy Bastidas que siempre estará guiando mi vida desde el cielo.

Carlos Eduardo Guevara Nichoy

Este trabajo está dedicado a las personas más importantes en mi vida, Hernán Vargas Méndez, Sonia Alvarado Osorio y Luis Hernán Vargas Alvarado quienes siempre con una sonrisa dibujada en el rostro me muestran que todo estará bien a pesar de lo adversa e injusta que pueda ser la vida y me han impulsado a poner la cara a retos tan grandes como este trabajo de grado para llevarlos a buen puerto. A Diana Carolina Quintero porque se convirtió en un apoyo incondicional en mi vida.

De igual manera agradezco a mi gran amigo Carlos Eduardo Guevara Nichoy por su incansable apoyo e interés en este proyecto en el que nos embarcamos y porque quedan muy buenos recuerdos y anécdotas para la vida. Infinitos agradecimientos al Ing. César Leonardo Niño por sus enseñanzas, por su buena manera para guiarnos para obtener los mejores resultados, sin él sería impensable llevar a cabo este trabajo de grado.

Juan David Vargas Alvarado

Tabla de contenido

1. INTRODUCCIÓN....1

2. MARCO TEÓRICO....1

3. ESPECIFICACIONES....4

4. DESARROLLOS....6

4.1 CARGA DE VIDEO Y CONFIGURACIÓN DE PARÁMETROS....8

4.2 SEGMENTACIÓN DE FONDO....9

4.2.1 Principio del algoritmo....9

4.2.2 Análisis de parámetros....10

4.3.3 Proceso....11

4.3 DETECCIÓN DE PUNTOS DE INTERÉS....13

4.3.1 Principio del algoritmo....13

4.3.2 Proceso....14

4.4 CONSTRUCCIÓN DE ROIS....14

4.4.1 Principio del algoritmo k-medias....15

4.4.2 Proceso....16

4.5 SEGUIMIENTO Y PREDICCIÓN....17

4.5.1 Seguimiento....17

4.5.2 Predicción....22

4.6 DECISIÓN....24

5. ANÁLISIS DE RESULTADOS....25

5.1 CASO DE ANÁLISIS....25

5.2 DESEMPEÑO DE ATRIBUTOS....37

5.3 ANÁLISIS POR GRUPOS DE *FRAMES* Y DESEMPEÑO DEL ALGORITMO EN INTERIORES Y EXTERIORES....38

6. CONCLUSIONES....46

7. BIBLIOGRAFÍA....48

8. ANEXOS....49

ANEXO A: ALGORITMOS....49

1. Detección de Puntos de Interés....49

2. Seguimiento....52

ANEXO B: TABLAS DE RESULTADOS....54

ANEXO C: CÓDIGO....57

1. Función mixog (Sustracción de Fondo)....57

2. Función corner (Detección de Puntos de Interés)....60

3. Función SSDXCORR (Función de Tracking)....68

4. Interfaz Gráfica del algoritmo....69

9. LISTA DE FIGURAS....81

1. INTRODUCCIÓN

Según confirmó Fenalco, las pérdidas debidas al robo de mercancía en los grandes supermercados ascendieron a más de 166 mil millones de pesos al finalizar el año 2008. Este fenómeno viene en crecimiento y se hace evidente al destacar los resultados obtenidos en el Censo Nacional de Mermas [1], donde se pone en evidencia que durante el 2009 las principales cadenas de comercio reportaron pérdidas por 420.119 millones de pesos, donde el robo continúa siendo la principal causa de las pérdidas en los supermercados. El 40 por ciento de las pérdidas obedeció a desperdicios, averías en las mercancías, y vencimientos, 8 por ciento a errores administrativos, en tanto que 41 por ciento tuvo como origen los robos, o sea 172.249 millones de pesos, del cual un 19 por ciento corresponde a los cometidos por funcionarios o empleados de los establecimientos y 22 por ciento por los clientes o visitantes a los almacenes y supermercados.

Gracias a la masificación en cuanto a implementación de sistemas de seguridad y vigilancia como los CCTV (*Closed Circuit Television*) en grandes superficies y a la incapacidad de procesar toda esta información por parte de operadores humanos, surge la idea de desarrollar y ahondar en la investigación de una serie de algoritmos que permitan realizar un monitoreo del comportamiento de las actividades de clientes y visitantes a los diversos lugares de interés. Factores como la potencia de procesamiento de las computadoras actuales, la disponibilidad de amplia información, y la facilidad de acceso a este tipo de tecnología hacen viable la posibilidad de hacer un seguimiento efectivo a visitantes en estos establecimientos.

El propósito de éste algoritmo es el de mejorar los niveles de seguridad en grandes superficies, buscando disminuir cifras de hurto y de violencia a través de la construcción de perfiles de comportamiento a partir de regiones de interés para la detección de anomalías que constituye el objetivo general de éste trabajo.

Los registros de video presentan múltiples escenarios y diversas situaciones, podemos encontrar una persona sola o un grupo de personas, acciones como caminar o simplemente ver vitrinas dentro de un centro comercial, hasta acciones como una caída o enfrentamientos entre personas, asimismo, los escenarios son diversos, se pueden encontrar registros de video al interior de una oficina, dentro de un centro comercial, en una bodega o incluso en exteriores como una calle o un parqueadero. La clasificación de los videos se hace entonces teniendo en cuenta los escenarios y las personas que intervienen. De acuerdo a los escenarios los registros se clasificaron en videos interiores (*indoor*) y exteriores (*outdoor*) y de acuerdo a las personas que intervienen en el video se clasifica como una persona o un grupo de personas.

La integración de las múltiples técnicas formuladas y desarrolladas en [2], [3], [4] y [5] como la segmentación de fondo, la detección de puntos de interés, el agrupamiento en regiones de interés y el seguimiento de objetos permitirán construir perfiles de comportamiento humano para su posterior estudio, análisis y caracterización, arrojando como resultado final la construcción de perfiles de comportamiento humano a partir de regiones de interés para la detección de anomalías consiguiendo así el objetivo general propuesto.

Para cada una de las técnicas se profundizará en fundamentaciones teóricas con sus respectivas referencias bibliográficas, presentando primero una idea general para luego terminar con la descripción de la herramienta utilizada para que el lector encuentre en el informe todo lo respectivo al desarrollo del programa. Después de contar con las bases teóricas se mostrará la implementación, los desarrollos y resultados de cada una de las etapas que conciernen a este proyecto, para evaluar el cumplimiento y la consecución de los objetivos propuestos.

2. MARCO TEÓRICO

La construcción de perfiles de comportamiento ha sido estudiada desde diversos enfoques en [6], [7], [8] y [9]. Para lograr la construcción de perfiles de comportamiento a partir de una secuencia de video, se deben poner en consideración diferentes técnicas y herramientas que permitan la consecución del objetivo propuesto. El desarrollo de un perfil de comportamiento incluye discriminar qué

características (posición, velocidad, aceleración, valor del pixel en RGB y tono de gris, *kurtosis*, asimetría, etc.) son relevantes y cuáles no lo son. Es importante tener presente que las características son funciones que tienen como variable el tiempo.

Se tendrán en cuenta conceptos importantes durante el desarrollo del trabajo como la segmentación de fondo [10], [2], [11], [12], la detección de esquinas [13], [14], [15], [16], la generación de regiones de interés [17], [4], [18], [19], [20], *tracking* [21], [5], predicción [22], etc., que serán las herramientas que posibilitaran poder definir perfiles de comportamiento con el fin de detectar anomalías dentro de la secuencia de video. Muchas de éstas técnicas ya han sido estudiadas y desarrolladas por otros autores.

A continuación se hará una recopilación del estado del arte de los conceptos mencionados anteriormente con el fin de proporcionar bases teóricas sólidas al desarrollo del trabajo.

La sustracción o segmentación de fondo es una técnica comúnmente usada para segmentar objetos de interés en una escena para ser implementado en aplicaciones, como en el caso particular, de vigilancia, y es tal vez, la parte del proceso de generación de perfiles más importante de este proyecto, ya que de este depende la fidelidad de los puntos de interés sobre el objeto en movimiento y la generación de las ROIs para realizar un seguimiento eficaz en el tiempo.

La técnica de segmentación de fondo ha sido tratada por X. Zhang, J. Jiang, Z. Liang, C. Liu en [2], y Y. Wang y B. Yuan en [11], donde realizan segmentación de piel para detección de rostros, gran aporte puede brindar este tipo de segmentación, sin embargo recopilar únicamente esta información resulta limitada ya que se perderían características diferentes a la cara, por lo que para el caso de nuestra segmentación se tendrán en cuenta diferentes métodos como *frame difference*, correlación espacial y mezcla de Gaussianas.

La mezcla de Gaussianas ofrece múltiples ventajas como robustez, oclusión parcial y estabilidad, además de contar con elevada precisión, sin embargo, el costo computacional de la implementación de esta técnica es de igual manera elevado. En [12] se realiza el análisis de los parámetros que componen el modelo de mezcla de Gaussianas.

No menos importante, la identificación de puntos de interés sobre el objeto segmentado jugará un papel trascendental durante la generación de perfiles de comportamiento, este proceso se puede llevar a cabo por medio de diversas técnicas [13] como Harris [14], Noble [23], SUSAN [15] y CSS [16].

La localización de puntos de interés dentro de una escena es de vital importancia y es ampliamente usada en aplicaciones de procesamiento de imágenes y visión por computador para caracterizar los objeto de interés.

Harris realiza un análisis acerca de los auto-vectores de una matriz compuesta por derivadas de la intensidad lumínica. La matriz de auto-correlación se estima por las derivadas de primer orden. Los vectores propios de la matriz son las direcciones de máximo y mínimo cambio, por lo que se puede deducir según los valores propios la existencia de un punto uniforme, punto de borde o puntos de esquinas dependiendo del valor que tomen.

Noble define una nueva medida de selección de esquina como función de la matriz de auto-correlación en el detector de esquinas de Harris y Stephens, eliminando un parámetro que dependía de la información de la imagen y agregando una constante (ε) independiente que es mucho más fácil de seleccionar.

El detector de esquinas SUSAN es el principio implementado por Smith y Brady y usa una máscara circular que bordea cada pixel con el fin de verificar la existencia de esquinas. Buscando el enfoque de rapidez, se busca evitar el uso de derivadas para la obtención de las características bidimensionales. El elemento primordial de SUSAN es el filtro no lineal para la detección de características, y lo que hace es calcular el área de la imagen que tiene el mismo nivel de gris que el centro de la máscara circular. El filtro acumula el número de puntos dentro del círculo con un nivel de gris similar al del centro.

Finalmente, CSS brinda la capacidad de detectar características de curvaturas tanto gruesas como finas a un costo computacional bajo. Adicionalmente, CSS realiza una detección de esquinas en la imagen a multi-escala y cuenta con un umbral adaptativo.

En el trabajo desarrollado por X. Chen He y N. H. C. Yung, [3], se desarrolla el detector de curvaturas CSS. El método propuesto se basa en la generación de un mapa de contornos del cual se computa la curvatura absoluta a baja escala para retener la mayor cantidad posible de esquinas, ya sean falsas o verdaderas. Se asume que todas las esquinas verdaderas están incluidas en el grupo de las esquinas candidatas junto con las esquinas falsas. Esta suposición es verdadera en la medida en que el mapa de contornos sea extraído usando un umbral bajo y la escala usada sea de igual manera baja.

Dado que un local máximo puede representar esquinas verdaderas como circulares y también ruido, Chen y Yung proponen dos criterios para eliminar el ruido y las esquinas circulares. Para conseguir este cometido, primero se comparan las esquinas candidatas usando un umbral adaptativo local que es calculado automáticamente en lugar de solo hacer uso de un umbral global para remover las esquinas circulares. Lo que se hace posteriormente es que los ángulos restantes de las esquinas candidatas se evalúan con el fin de eliminar cualquier tipo de esquina falsa debido al ruido de cuantización y detalles triviales. La evaluación se basa en una región dinámica de soporte, la cual varía de una esquina a otra según la esquina candidata contigua. Finalmente introducen el método para que los extremos de las curvaturas sean tratados como tal.

Existen diversos criterios para realizar el agrupamiento en regiones contenedoras de puntos de interés (ROIs), algunos de ellos propuestos en [17], [4], [18], [19], [20]. Serán estas ventanas los objetos de predilección para caracterizarlos de acuerdo a su comportamiento en el tiempo, según su posición, velocidad, aceleración, valor del pixel en RGB, valor de *kurtosis* y asimetría.

El *Clustering* o agrupamiento hace referencia a la clasificación sin supervisión de un conjunto de datos en grupos o *clusters* [17], los datos pertenecientes a cada *cluster* son similares o comparten características. El *clustering* es utilizado en diversos contextos como el análisis de patrones, la toma de decisiones, el aprendizaje artificial, la recuperación de la información, la segmentación de imágenes, la clasificación de patrones, etc., a esto se debe su acogida en las diversas comunidades de investigación.

Como un ejemplo de *clustering* [4] tenemos la recuperación de la información que puede evidenciarse cuando se realiza una búsqueda en la internet, como hay millones de páginas web relacionadas con la información que estamos buscando, se puede utilizar el *clustering* para reducir el tamaño de los resultados mostrándolos por grupos de menor dimensión; si nuestra búsqueda fuese “libro”, los grupos que se pueden realizar serían género, idioma, tipo, etc. donde a cada uno se asignan las páginas correspondientes. Es posible encontrar grupos mucho más pequeños dando lugar a una estructura jerárquica. La estructura jerárquica es uno de los dos tipos de agrupamiento y consiste en un conjunto de agrupaciones anidadas que se organizan como un árbol, cada uno de los *clusters* es la unión de *subclusters*, todos los objetos componen un *cluster* que se asimila a la raíz del árbol. La siguiente estructura de agrupamiento es la estructura particional, donde simplemente los datos son asignados en grupos que no se traslapan entre sí, por lo que cada objeto sólo pertenecerá a un único grupo.

Dentro de los diversos algoritmos existentes para realizar *clustering* tenemos las redes neuronales, clasificadores de Bayes, *k-means* y *k-Nearest Neighbor*; de los mencionados, *k-means* es un algoritmo que cuenta con una estructura de agrupamiento particional.

El algoritmo *k-means* [18] es el algoritmo más popular a la hora de realizar agrupamiento, fue propuesto por *McQueen* en 1967. La velocidad computacional del algoritmo es su principal atractivo y a esto debe su popularidad, por lo anterior fue pensada su aplicación para éste desarrollo.

Con las ROIs construidas, se necesita obtener datos y funciones de interés, para esto es necesario ver qué ocurre con cada región durante un periodo de tiempo implementando un seguimiento o *tracking*. Existen diferentes métodos para realizar *tracking* pero debido a la necesidad de optimizar y reducir el costo computacional del algoritmo se decidió implementar el algoritmo planteado en [5].

Este algoritmo propone una nueva forma para realizar *tracking* mediante la estimación de movimiento de traslación de la imagen usando suma de diferencias de cuadrados. El modelo propuesto es muy sencillo y es utilizado en procesamiento de video por su bajo costo computacional.

En [5], la estimación de movimiento consiste en la definición de vectores de movimiento que indican los desplazamientos en las coordenadas del *frame* y la búsqueda de un vector óptimo que produzca el mínimo error. Por otro lado, la correlación cruzada es ampliamente usada en el *matching* de imágenes desplazadas ya que posee grandes ventajas como independencia de la iluminación y sombras además de que permite realizar estimación de grandes desplazamientos de la región. El proceso descrito en [5] indica que para un bloque g que pertenece al *frame* actual, se realiza una búsqueda en el *frame* de referencia sobre un área búsqueda f y con una ventana de búsqueda de tamaño fijo. El objetivo de la búsqueda es encontrar el bloque que mejor se ajuste y cuyo error sea mínimo. Para este algoritmo esto se consigue minimizando la suma de diferencia de cuadrados.

Para la evaluación de los resultados existen diferentes definiciones y métodos aceptados, en la mayoría se busca definir términos como exactitud, eficiencia o efectividad de un clasificador a la hora de realizar una prueba diagnóstica. En general no existe un acuerdo para la definición de éstos términos por lo que encontrar un método adecuado resulta en ocasiones ambiguo. El método propuesto en [24] y [25] es uno de los más comunes y sobre el que se tiene mayor información, éste método es conocido como curvas ROC (*Receiver Operating Characteristics*) y es utilizado en el análisis de señales para diferenciar las tasas de aciertos y de falsas alarmas generadas por un clasificador. Otra de las aplicaciones de este método es la evaluación de pruebas realizadas en medicina para discriminar entre pacientes sanos y enfermos. El método de curvas ROC define dos términos importantes que son la sensibilidad y la especificidad, el primero de ellos hace referencia a la probabilidad de obtener un resultado positivo de la prueba cuando éste debe ser así, es decir que mide la capacidad de aciertos, por otro lado la especificidad es entendida como la probabilidad de obtener un resultado negativo cuando éste debe ser asi, es decir que mide la capacidad de descarte.

3. ESPECIFICACIONES

Durante el desarrollo del trabajo de grado se hizo necesario ensayar diferentes técnicas y herramientas para la consecución de los diversos objetivos propuestos. En el proceso de experimentación, además de las herramientas necesarias, se logró determinar otras especificaciones del proyecto que no habían sido contempladas anteriormente, como el formato de video, resolución y tasa de *frames*, estas nuevas especificaciones contribuyeron a la delimitación del alcance del trabajo.

La rutina desarrollada se compone de diversos algoritmos que en conjunto tendrán como resultado la adquisición de los datos de interés y el posterior estudio. Algunos de los algoritmos usados fueron desarrollados por otros autores, cada uno de estos cuenta con una sustentación teórica que ha sido estudiada por diversas comunidades científicas, los algoritmos cuentan con su respectiva licencia y su uso está dirigido a la investigación y desarrollo de nuevo *software*. Para otras etapas se utilizaron las funciones propias de los *Toolbox* de MATLAB.

La secuencia de video es la entrada al programa y debe cumplir con las siguientes especificaciones. Formato del video: .avi, resolución del video: 720 x 480, tasa de *frames*: 30 *frames*/segundo. Estas especificaciones se establecieron después de realizar múltiples pruebas y se establecieron con el fin de mejorar la velocidad de procesamiento del programa, la resolución efectiva y obtener buenos resultados finales.

La primera etapa consiste en detectar en la secuencia de video lo que corresponde al fondo del video y separarlo, es decir realizar una sustracción de fondo; para esto se utilizó un algoritmo basado en Mezcla de Gaussianas [12] que emplea 5 *frames* para entregar después de éstos una imagen binaria con los objetos de interés que no hacen parte del fondo. Seguidamente, mediante operaciones morfológicas realizadas mediante el *Image Toolbox* de MATLAB [26], se identifica y se selecciona el objeto con mayor área asumiendo que es la persona o personas de interés.

Teniendo el objeto más grande diferenciado del fondo se devuelve las propiedades de color para determinar en esa imagen los puntos de interés usando el algoritmo de detección de esquinas CSS (*Curvature Scale Space*) [3] que arroja como resultado una matriz de N filas y 2 columnas, que corresponde a las coordenadas que tiene cada punto de interés sobre la imagen. Cada fila es un punto de interés.

Una vez se tienen los puntos de interés, el siguiente paso es agruparlos. El número de *clusters* va desde 3 hasta 5. Este número se determinó empíricamente ensayando en diversas secuencias de video. Para realizar los *clusters* se usa la función *kmeans* de MATLAB que se basa en el algoritmo de *k-medias* [17], [4], [18] y [19]. Cada *cluster* se encierra en el rectángulo más pequeño que contendrá todos los puntos de interés correspondientes a cada *cluster*, esta ventana se denomina ROI (*Region of Interest*).

Para la siguiente fase se encuentra un algoritmo que es el encargado de realizar la ubicación de cada ROI en cada uno de los *frames* de la secuencia de video e implementa una suma de diferencia de cuadrados usando correlación cruzada [5]. Durante el seguimiento de cada una de las ROIs se captura y almacena información de desplazamiento en X e Y del centro de la ROI y momentos de alto orden como varianza, *kurtosis* y asimetría (*skewness*).

Como etapa final se realiza el estudio y clasificación de la información obtenida de la secuencia de video para decidir qué segmento de información presenta anomalías o comportamientos extraños. Se realiza una estimación de la señal mediante un predictor lineal que utiliza 5 *frames* para predecir el comportamiento de los 5 siguientes y se compara con los resultados obtenidos del tracking. Para la discriminación de comportamientos se utiliza la suma de las señales de error para cada atributo, seguido de un umbral adaptativo.

Los pasos mostrados anteriormente se llevarán a cabo durante una ventana de 10 *frames*. Una vez haya transcurrido la ventana, se desplaza 5 *frames* quedando traslapada con la anterior. Este proceso se repite hasta el final de la secuencia de video con el fin de actualizar la información ya que ésta cambia durante el video y se pretende garantizar que sea reciente y acorde a lo que esta ocurriendo en la escena analizada.

La información anterior y las operaciones de cada una de las fases del programa se resumen en la Tabla I.

Tabla I. Fases de sustracción, detección puntos de interés, construcción de ROIs, *tracking*, estimación y clasificación y las herramientas utilizadas en cada fase del programa.

Fases del programa	Herramienta utilizada
Sustracción de fondo	**Método: Mezcla de Gaussianas**. Se realiza en los 5 primeros *frames* de la ventana. Recibe como entrada los 5 primeros *frames* de la ventana y arroja como salida el objeto de interés recortado y con sus propiedades de color
Detección de puntos de interés	**Método: CSS.** Se realiza en el quinto *frame* de la ventana. Recibe como entrada la imagen del objeto segmentado y arroja como salida una matriz de N filas y 2 columnas correspondientes a los N puntos de interés encontrados.
Construcción de ROIs	**Método: K-means.** Se realiza en el quinto *frame* de la ventana. Recibe la matriz de puntos de interés y arroja como salida las coordenadas de las ROIs.
Seguimiento	**Método: Block Matching.** Se realiza durante toda la ventana. Tiene como referencia las ROIs construidas, almacena información de los atributos.
Predicción	**Método: Predictor lineal.** Se realiza en los últimos 5 *frames* de la ventana. Recibe como entrada los datos de los 5 primeros *frames* de la ventana y entrega la predicción de los 5 últimos.
Decisión	**Método: Umbral adaptativo.** Se realiza en los últimos *frames* de la ventana. Recibe como entrada los datos de los atributos y arroja como salida si el *frame* se clasifica como normal o anómalo.

En la Figura 1 se muestra el diagrama de bloques que representa el algoritmo desarrollado, los procesos se realizan en una ventana de 10 *frames* y como se puede observar, vuelven a iniciar apenas culmina la etapa de decisión entre comportamientos normales o anómalos. Las entradas y salidas de estos bloques se encuentran en la Tabla 1. Cada uno de los bloques se explicará a fondo en la sección de Desarrollos.

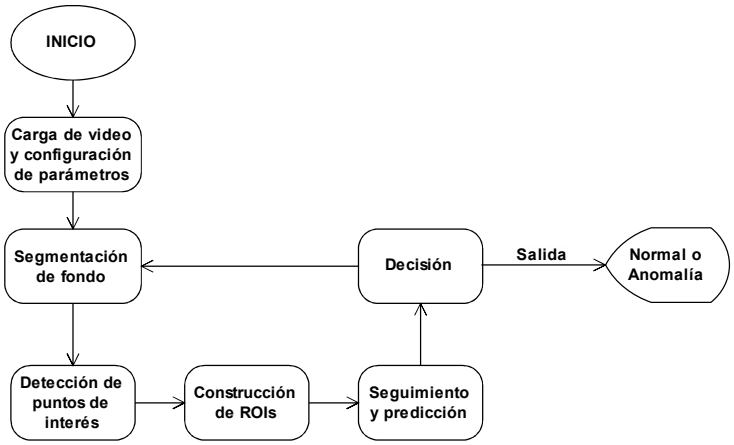


Fig. 1. Diagrama de bloques del algoritmo desarrollado.

4. DESARROLLOS

Se lograron múltiples desarrollos a nivel de software con este trabajo, desarrollos como la sustracción de fondo, la detección de esquinas, la generación de regiones de interés y el *tracking*, eran algoritmos que ya estaban implementados, sin embargo, se presentaron desafíos de gran magnitud por el hecho de que eran algoritmos totalmente independientes por lo que fue necesario adecuarlos para que en conjunto se pudiese obtener el resultado deseado.

No obstante, ante la necesidad de ser pertinentes con la elección de los parámetros se propusieron los atributos por los cuáles se tomaba la decisión acerca de la presencia o no de anomalía en la secuencia de video. Varias fueron sometidas a análisis para llegar a la conclusión de cuáles eran los parámetros y atributos que mejor definían la anomalía; atributos como el tono promedio en las regiones de interés, la posición del centro de la regiones de interés en los ejes coordenados *x* e *y*, así como la *kurtosis* de las mismas, su asimetría y varianza fueron tenidos en cuenta. Finalmente se descartó el parámetro de tono promedio debido a que aportaba información casi nula acerca de los eventos que tenían presencia en la secuencia de video.

El algoritmo implementado utiliza el diagrama de bloques que se muestra en la Figura 1, como ya se dijo, es un proceso repetitivo, todos los procesos a excepción de la carga y configuración de parámetros son periódicos, se realizan durante una ventana cuyo tamaño es de 10 *frames* con el fin que el algoritmo sea adaptativo, ya que la información se actualiza continuamente. En cada ventana se realizan los procesos indicados en la Figura 2 seguidos de una predicción y una toma de decisión. Estos procesos no se realizan en todos los *frames* de la ventana ya que para la predicción se necesitan datos almacenados para así lanzar una predicción de los siguientes. El movimiento de la ventana permite la actualización de las regiones, ésta se mueve una cantidad fija de 5 *frames* quedando traslapada con la anterior, haciendo que los datos obtenidos sean funciones continuas y no existan saltos de datos entre ventanas.

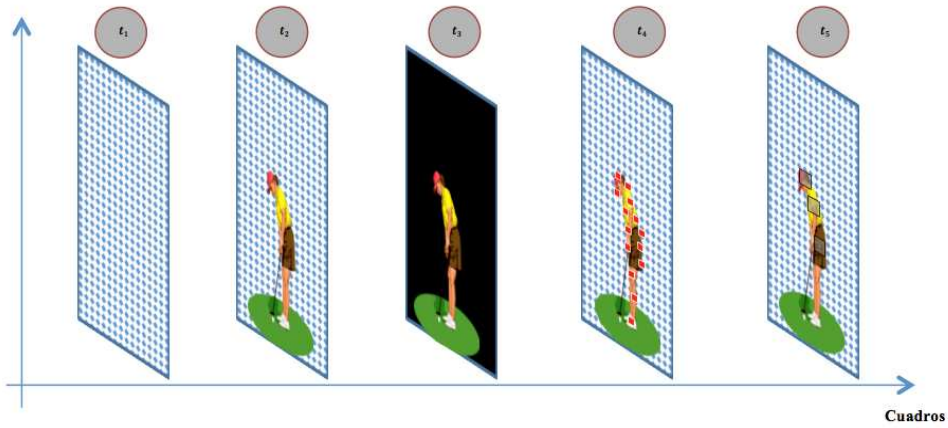


Fig. 2. Descripción de proceso de construcción de perfiles de comportamiento. t_1: Estimación de fondo. t_2: Detección de objeto de interés. t_3: Eliminación de fondo. t_4: Detección de puntos de interés. t_5: Agrupamiento de puntos de interés en ROI's para posterior seguimiento.

El algoritmo no necesita secuencias de entrenamiento, por lo que es independiente del video con que se esté trabajando; ésta es una gran ventaja ya que la aplicación es en seguridad, donde las situaciones y escenarios son muy diversos y no se limita a unos cuantos escenarios y comportamientos posibles; el aprendizaje se da en los *frames* previos a la predicción y la toma de decisión. En las secciones siguientes se explicara a fondo los bloques del diagrama de la Figura 1 y el proceso descrito anteriormente.

Además del desarrollo de software se hizo necesaria la elaboración de una base de datos con videos propios debido a la dificultad para obtener videos de situaciones reales, ya que son especialmente cuidados por las empresas y sus departamentos de seguridad. Se elaboraron un total de 40 videos. En estas secuencias se recrearon comportamientos de un sólo sujeto como saltos, caídas, desmayos, flexiones, sentadillas, lanzamiento de golpes y huidas inesperadas para diferentes escenarios y se llevaron a cabo tanto en interiores como en exteriores y con diferentes resoluciones efectivas (distancia del sujeto hacia la cámara). Igualmente se recrearon acciones para dos sujetos, especialmente peleas, con las mismas características descritas anteriormente. Se mantuvo la cámara fija y la iluminación controlada aunque en algunos videos se generaron sombras que pueden producir errores en los resultados obtenidos. Los videos elaborados son de corta duración debido a que la tasa de *frames* por segundo es de 30, por lo que hay un gran número de imágenes por procesar. La duración va desde dos hasta doce segundos como máxima. En algunas secuencias de video se puede observar que existe oclusión parcial.

Cada video cuenta con su respectivo *ground truth*, archivo que describe detalladamente los comportamientos encontrados en cada video; serán éstos los que posteriormente serán utilizados para la evaluación de desempeño del algoritmo al comparar los resultados obtenidos de cada video con su respectivo *ground truth*. Es importante resaltar que el *ground truth* es elaborado de manera subjetiva, sin embargo, se hace de la manera más objetiva posible y corresponde con lo que se aleja del contexto de la secuencia.

Debido a la importancia de la medición de lo sucedido en una secuencia de video y con el fin de medir el desempeño del código desarrollado, se generaron herramientas que permitieran confirmar el buen funcionamiento del mismo, cerciorándose que los resultados eran convincentes y acordes a lo que se presenciaba en el video.

Se generó mediante MATLAB un video de control para mostrar y evaluar el desempeño del algoritmo usado para realizar la sustracción de fondo en cada *frame*. El proceso de mezcla de Gaussianas se realiza cada 5 *frames*, el primero de ellos siempre será totalmente blanco ya que es el *frame* que se establece como fondo inicial, los *frames* restantes muestran lo que realmente es de nuestro interés que es todo lo que está en movimiento en el video, es decir, lo que no hace parte del fondo. El video de control se completa al terminar todo el proceso y es almacenado en una estructura de nombre MOG. El paso siguiente es el de creación del video mediante comando de MATLAB.

Otro video de control importante es el que contiene los *frames* que se utilizan para encontrar puntos de interés, proceso que se da después de realizar mezcla de gaussianas y de la selección del objeto de predilección, a éste se devuelven propiedades de color y se buscan los puntos de interés. En el video, los puntos están marcados con color, cada color indica a que región de interés pertenece dicho punto. Igual que el video anterior, éste se completa al terminar todo el proceso y es almacenado en una estructura de nombre CSS.

Para la evaluación del algoritmo de *tracking* nuevamente se genera un video. La región que es objeto de seguimiento se encierra en un rectángulo que indica donde se encuentra, es decir, que para considerar como óptimo un algoritmo de *tracking,* las regiones deberán aparecer sobre el/los sujeto/os en movimiento. El video que corresponde al *tracking* de las ROIs mantiene los mismos colores usados para identificar la ROI del video de puntos de interés. Los videos se almacenan en estructuras de nombres V y VT, el primero es un *tracking* auxiliar de la región identificada en movimiento y el último corresponde al *tracking* de cada una de las regiones de interés.

4.1 Carga de video y configuración de parámetros.

Éste es el bloque inicial del algoritmo y realiza diferentes tareas para garantizar el funcionamiento óptimo del mismo. Arranca con almacenar en un espacio de memoria la secuencia de video que ha sido seleccionada por el usuario, seguido de la obtención de información de la misma. La información extraída de la secuencia es la de duración del video y número de *frames*. Ésta información es utilizada para definir el tamaño de la ventana.

Igualmente éste bloque define el número de ROIs y el factor de escala para el umbral a partir de la información ingresada por el usuario, además establece el orden del predictor lineal que es utilizado.

Por último se crean todos los espacios de memoria necesarios para almacenar y procesar toda la información recogida durante el proceso, estos espacios de memoria están en función del número de *frames* y del número de ROIs.

En el código que se encuentra en el ANEXO CÓDIGO las siguientes variables corresponden con las tareas desarrolladas por el bloque.

- *Frames*, almacena el número de *frames* de la secuencia.
- Duración, almacena la duración de la secuencia.
- Tasa, almacena el número de *frames* por segundo de la secuencia, obtenido a partir de la duración y el número de *frames*.
- mínimo y máximo, usadas para definir el tamaño la ventana.
- l y h, usadas para limitar el proceso de sustracción de fondo.
- k, define el número de ROIs con las que se va a trabajar.
- ROIS, espacio de memoria que almacena las coordenadas de las ROIs.
- orden, corresponde al orden del predictor lineal.
- Desplazamientov, Desplazamientoh, Varianza, *Kurtosis* y Skewness, arreglos usados para almacenar los datos obtenidos de la secuencia y la predicción, éstos están en función del número de ROIs y del número de *frames*. Cada arreglo es de dimensión: $Frames \times 3 \times k$, cada matriz de las k corresponde con una ROI, cada fila de cada matriz es usada para almacenar diferentes tipos de información, la primera almacena los datos de los últimos cinco *frames* de la ventana, la segunda almacena los datos obtenidos de la predicción realizada y la tercera fila almacena los datos de los primeros cinco *frames* de la ventana.
- edv, edh, ev, ek y es, arreglos utilizados para almacenar el error entre las señales de la secuencia y de la predicción, están en función del número de ROIs y del número de *frames*. Cada arreglo es de dimensión: $Frames \times 1 \times k$

- sdv, sdh, sv, sk y ss, vectores utilizados para almacenar la suma de los atributos comunes de las las ROIs, están en función del número de *frames*.

4.2 Segmentación de fondo

Con todos los parámetros listos, el siguiente paso del algoritmo es la realización de la sustracción de fondo para detectar así el sujeto en movimiento. La selección del algoritmo para este punto se realizó teniendo en cuenta diversos parámetros, pero el más relevante fue el de robustez, es así como se eligió un algoritmo que realiza sustracción de fondo usando un modelo de Mezcla de Gaussianas [12].

En el algoritmo implementado, se presenta el modelo de Mezcla de Gaussianas para sustracción de fondo en escenas estáticas, donde el dispositivo de captura de imágenes se encuentra fijo. El modelo tiene diversos parámetros que pueden ser ajustados para producir diferentes resultados de acuerdo a los escenarios y comportamientos registrados.

4.2.1 Principio del algoritmo

A partir de [12] se inicia la implementación con la definición de parámetros:

X_t: variable que representa el pixel actual en el cuadro actual I_t.
C: Número de distribuciones.
t: representación del tiempo (índice de cuadro).
$w_{i,t}$: es el estimado del peso de la i-ésima Gaussiana en la mezcla en el tiempo t.
η: función de densidad espectral de la Gaussiana.
$\mu_{i,t}$: valor medio de la i-ésima Gaussiana en la mezcla en el tiempo t.
$\Sigma_{i,t}$: matriz de covarianza de la i-ésima Gaussiana en la mezcla en el tiempo t.

Las anteriores funciones se combinan para brindar una función de densidad combinada.

La construcción del modelo se realiza mediante la computación de las probabilidades de cada pixel de color.

En general el modelo de mezcla de Gaussianas se formula como:

$$P(X_t) = \sum_{i=1}^{C} w_{i,t}\eta\left(X_t; \mu_{i,t}, \Sigma_{i,t}\right)$$

Donde,

$$\sum_{i=1}^{C} w_{i,t} = 1$$

Media de una mezcla es igual a:

$$\mu_t = \sum_{i=1}^{C} w_{i,t}\mu_{i,t}$$

Esto es la suma ponderada de las medias de las densidades de los componentes.

Se dice que X_t coincide con el parámetro i si X_t está dentro del valor 2.5 de la desviación estándar de esa distribución, es posible que se presenten múltiples coincidencias. Los parámetros del i-ésimo componentes se actualizan así:

$$w_{i,t} = (1-\alpha)w_{i,t-1} + \alpha$$
$$\mu_{i,t} = (1-\rho)\mu_{i,t-1} + \rho X_t$$

$$\sigma^2{}_{i,t} = (1-\rho)\sigma^2{}_{i,t-1} + \rho\left(X_t - \mu_{i,t}\right)^T + \left(X_t - \mu_{i,t}\right)$$

Dónde:

$$\rho = \alpha P\left(X_t|_{\mu_{i,t-1},\Sigma_{i,t-1}}\right)$$

α: parámetro de aprendizaje predefinido
$\sigma^2{}_{i,t}$: varianza de la i-ésima Gaussiana en la mezcla en el tiempo t
μ_t: es la media del pixel en el tiempo t
Los parámetros de todas las distribuciones que no tienen coincidencias permanecen sin cambiar, esto es:

$$\mu_{i,t} = \mu_{i,t-1}$$
$$\sigma^2{}_{i,t} = \sigma^2{}_{i,t-1}$$

Sin embargo, los pesos $w_{i,t}$ tienen que ser ajustados mediante:

$$w_{i,t} = (1-\alpha)w_{i,t-1}$$

Si X_t no coincide con ninguna de las C distribuciones, entonces la distribución menos probable se remplaza por una distribución donde el valor actual actúa como el valor medio, y la varianza se escoge para que sea alta y el peso a-priori sea bajo.

El problema de estimación de fondo se resuelve especificando las distribuciones Gaussianas que tienen la evidencia más influyente y la menor varianza.

Como los objetos en movimiento tienen mayores varianzas que un pixel perteneciente al fondo y para representar los procesos de fondo, primeros las Gaussianas se ordenan de acuerdo al valor de $w_{i,t}/\left\|\Sigma_{i,t}\right\|$ de manera decreciente.

La distribución de fondo con menor varianza es la que se posiciona en primer lugar mediante la aplicación de un umbral, donde:

$$B = argmin_b\left(\frac{\sum_{i=1}^{b} w_{i,t}}{\sum_{i=1}^{C} w_{i,t}} > T\right)$$

Todos los pixeles X_t que no coincidan con ninguna de los componentes son catalogados como pixeles pertenecientes a un objeto en movimiento, es este caso, el objeto de interés.

4.2.2 Análisis de parámetros

En [19] se realiza el análisis de los parámetros que componen el modelo de mezcla de Gaussianas, donde menciona que para valores pequeños de *T* (relación entre distribución de *background* y *foreground*) como 0.1 funciona adecuadamente cuando no se cubre toda la distribución de fondo. Valores elevados de T, como 0.9 es adecuado para situaciones donde la distribución del objeto de interés se fusiona con la distribución del fondo.

El número de componentes en el modelo de mezcla de Gaussianas C es dependiente del ambiente donde se va a poner a trabajar. Para espacios interiores poco complejos un valor de 2 componentes es suficiente. Para escenas complejas como exteriores, el número de componentes puede ser de 3 a 5. Para el caso particular, se usan 3 componentes, ya que se pueden presentar escenas con algún grado de complejidad dado que el algoritmo está orientado a la aplicación en grandes superficies y lugares frecuentados por más de una persona.

Otros parámetros son ρ y α, que se refieren a la tasa de aprendizaje. α es el parámetro de aprendizaje predefinido, y ρ es usado como un segundo filtro que es calculado, sin embargo, el no tener en cuenta este último parámetro no afecta el desempeño del algoritmo y por el contrario, se libera al algoritmo

de una carga computacional adicional que es desfavorable para el rendimiento, por lo que no se tuvo en cuenta. Los valores típicos para α oscilan entre 0.01 y 0.5. Observando los resultados obtenidos en [12] y tras la realización de pruebas, para éste parámetro se eligió un valor de $\alpha = 0.01$.

La asignación inicial de los parámetros se realizó teniendo en cuenta videos en diferentes contextos que fueron evaluados en este proyecto: un sujeto en un ambiente interior complejo, un sujeto en un ambiente exterior complejo, dos sujetos en un ambiente interior sencillo y dos sujetos en un ambiente exterior sencillo; y buscando obtener la mayor eficiencia al momento de hacer la discriminación entre fondo y objeto de interés.

4.3.3 Proceso

Como se mencionó anteriormente, varias tareas de la aplicación se realizan periódicamente permitiendo así la actualización de la información y tener un algoritmo adaptativo. El algoritmo de sustracción de fondo debe tener entonces, una duración fija dentro de la ventana de 10 *frames*, en nuestro caso se seleccionó de 5 *frames*, donde el primero de ellos se elige como fondo inicial de la secuencia del video y a partir de este, se realiza el proceso de sustracción en los *frames* siguientes. Se elige el último *frame* de los cinco como resultado del proceso, en este punto se tiene una imagen binaria, donde en negro se encuentran los pixeles que no corresponden al fondo.

La imagen obtenida presenta ruido resultado de cambios en la iluminación o por sombras de los sujetos, en este punto se utiliza una operación morfológica conocida como *close* [26], que consiste en realizar una dilatación seguida de una erosión. Ésta operación permite rellenar huecos de los objetos sin distorsionarlos, pero no tiene la capacidad de eliminar ruido, al rellenar los objetos y hacerlos más grandes, permite una fácil extracción del objeto de interés, que corresponde al más grande encontrado en dicho *frame* de la secuencia de video y que se recorta usando las coordenadas de su *BoundingBox*. Después de realizar este corte se procede a devolver a esa imagen los colores que tenía originalmente. La imagen con el objeto más grande recortado del *frame* y que cuenta con todas sus propiedades de color es la salida de este bloque.

Para realizar este proceso se utiliza la función ***[cout,auxiliar,G,COR]=mixog(source,l,h,G)*** [27] que se encuentra implementada en el ANEXO CÓDIGO, ésta tiene como parámetros de entrada la secuencia de video (*source*) y los limites inferior y superior para delimitar el proceso de sustracción de fondo (*l y h*, límite inferior y superior respectivamente). Las salidas de esta función son las coordenadas de los puntos de interés (*cout*), una imagen con el objeto de interés recortado y el resto de la imagen en negro (*auxiliar*), una estructura que contiene los *frames* que han sido procesados y que son utilizados para la construcción del video de evaluación (*G*) y las coordenadas y dimensiones del *BoundingBox* que contiene el objeto de interés (*COR*). Con ésta función se puede obtener los puntos de interés ya que la función que realiza este procedimiento se encuentra incluida en *mixog;* se hizo así ya que la detección de puntos se hace con el último *frame* usado para la sustracción de fondo, se aprovechó entonces para realizar la detección de puntos de interés inmediatamente después de que termine éste proceso.

Con los parámetros ajustados, a continuación se mostrarán algunas de las secuencias generadas por este proceso en diferentes videos que presentan ambientes y situaciones diversas, para así dar claridad y complementar la explicación del proceso llevado a cabo por este bloque.

La primera secuencia que se va a mostrar es la de correspondiente al inicio del video C6.avi, que se encuentra en la Figura 3, donde interviene un sujeto en la escena. Esta secuencia se desarrolla en un ambiente exterior.

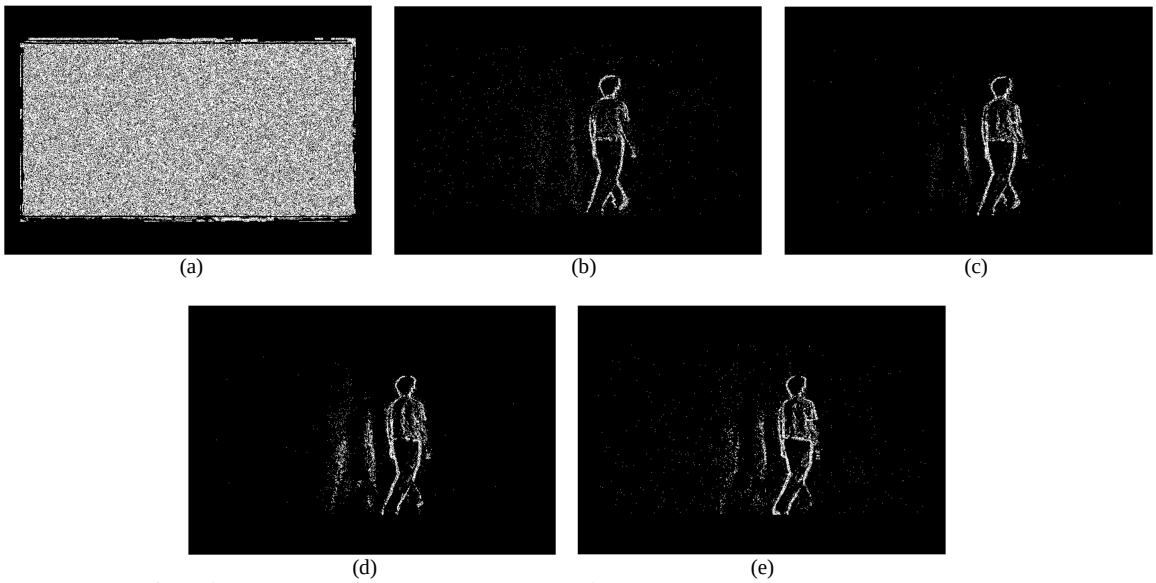


Fig. 3. Sustracción de fondo para el video C6.avi, se indican los *frames* objeto de este proceso. En puntos blancos se muestra lo que no pertenece al fondo: (a) Sustracción de fondo para el *frame* 1, (b) Sustracción de fondo para el *frame* 2, (c) Sustracción de fondo para el *frame* 3, (d) Sustracción de fondo para el *frame* 4, (e) Sustracción de fondo para el *frame* 5.

En la Figura 3 se encuentran los cinco *frames* que pertenecen al proceso de sustracción de fondo, como se explicó anteriormente, el primer *frame* del proceso se elige como fondo inicial, éste siempre va a estar en blanco, indicando así el inicio del proceso. En los *frames* siguientes se lleva a cabo el proceso de sustracción de fondo, el sujeto en movimiento se marca en blanco, en la Figura 3 (d) y la Figura 3 (e) se puede ver que la sombra de sujeto se marca igualmente, indicando así que no pertenece al fondo. La Figura 3 (e) es la imagen seleccionada para realizar la operación morfológica *close*, para después realizar el recorte del objeto más grande y finalmente devolver las propiedades de color. Es ésta imagen la salida del proceso que será usada posteriormente por el bloque de detección de puntos de interés.

En la Figura 4 se muestra el proceso de sustracción de fondo para la secuencia de video C19.avi, en ésta hay dos sujetos que participan en la escena, el video se desarrolla igualmente en un ambiente exterior.

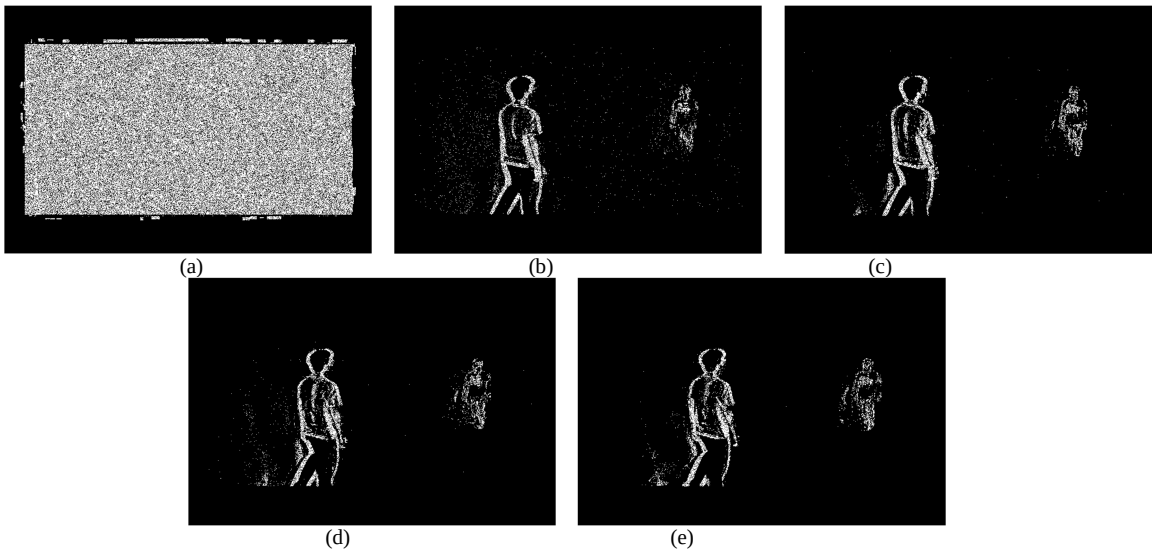


Fig. 4. Sustracción de fondo para el video C19.avi, se indican los *frames* objeto de este proceso. En puntos blancos se muestra lo que no pertenece al fondo: (a) Sustracción de fondo para el *frame* 56, (b) Sustracción de fondo para el *frame* 57, (c) Sustracción de fondo para el *frame* 58, (d) Sustracción de fondo para el *frame* 59, (e) Sustracción de fondo para el *frame* 60.

La Figura 4 contiene los cinco *frames* de la sustracción de fondo, pero a diferencia de la Figura 3, el proceso inicia en un *frame* diferente al primero, mostrando así que el proceso es repetitivo durante todo el video. El proceso inicia con el *frame* 56 como se muestra en la Figura 4 (a), los *frames* siguientes muestran en blanco a los dos sujetos que intervienen en la escena, también se puede

observar que en algunos *frames* se marcan nuevamente sobras de los sujetos, la Figura 4 (e) nuevamente pasa por el proceso de *close*, seguido de la extracción del objeto más grande y finalmente el retorno de las propiedades de color, para obtener así la imagen de entrada al bloque de detección de puntos de interés.

4.3 Detección de puntos de interés.

La localización de puntos de interés dentro de una escena es de gran importancia y es ampliamente usada en aplicaciones de procesamiento de imágenes y visión por computador.

Con la discriminación entre objeto de interés y fondo, el siguiente paso es realizar la detección de rasgos distintivos dentro de la imagen recibida del bloque anterior; no tiene sentido realizar la detección de puntos en toda la imagen ya que generaría puntos que no son de importancia para el análisis del comportamiento del sujeto. Para la consecución de este objetivo se estudiaron diversos mecanismos y métodos para detección de puntos de interés.

Entre los métodos que se estudiaron se encuentran el detector de esquinas de Harris y Stephens, el detector de esquinas de Noble, SUSAN y CSS.

CSS, es el método apropiado para implementar en el trabajo no sólo por su capacidad de detectar características de curvaturas tanto gruesas como finas, sino también porque lo hace a baja carga computacional. Adicionalmente, CSS realiza una detección de esquinas en la imagen a multi-escala y cuenta con un umbral adaptativo.

En el trabajo desarrollado por X. Chen He y N. H. C. Yung [3], se desarrolla el detector de curvaturas CSS. El método propuesto se basa en la generación de un mapa de contornos del cual se computa la curvatura absoluta a baja escala para retener la mayor cantidad posible de esquinas, ya sean falsas o verdaderas. Todos los máximos locales de la función de curvatura absoluta se establecen como candidatas a esquinas. Se asume que las esquinas verdaderas están todas incluidas en el grupo de las esquinas candidatas junto con las esquinas falsas. Esta suposición es verdadera en la medida en que el mapa de contornos sea extraído usando un umbral bajo y la escala usada sea baja de igual manera.

Dado que un local máximo puede representar esquinas verdaderas como circulares y también ruido, Chen y Yung proponen dos criterios para eliminar el ruido y las esquinas circulares. Para conseguir este cometido, primero se comparan las esquinas candidatas usando un umbral adaptativo local que es calculado automáticamente en lugar de sólo hacer uso de un umbral global para remover las esquinas circulares. Lo que se hace posteriormente es que los ángulos restantes de las esquinas candidatas se evalúan con el fin de eliminar cualquier tipo de esquina falsa debido al ruido de cuantización y detalles triviales. La evaluación se basa en una región dinámica de soporte, la cual varía de una esquina a otra de acuerdo a la esquina candidata contigua. Finalmente introducen el método para que los extremos de las curvaturas sean tratados como tal.

4.3.1 Principio del algoritmo

La filosofía del método propuesto en [3], es utilizar las propiedades globales y locales de las curvaturas e influenciar dichas propiedades al momento de hacer la extracción de las esquinas.

El método se fundamenta en primera instancia en detectar los bordes usando el detector de bordes de Canny para obtener un mapa de bordes binario. Se extraen seguidamente los contornos de los bordes del mapa que se había generado, se realiza el llenado de espacios cuando se llegue a puntos de fin de los bordes y se prosigue con la extracción si algún punto de fin está en cercanía con otro punto de fin, o, se marca dicho punto como esquina de unión en *T* si el punto de fin está conectado a un contorno de borde, pero no a otro punto de fin.

Posterior a la extracción de contornos, se calcula la curvatura en una escala baja fija para cada contorno para almacenar las esquinas verdaderas y considerar los máximos locales de la curvatura absoluta como candidatas a esquinas.

Se calcula un umbral adaptativo de acuerdo a la curvatura media dentro de la región de soporte. Las esquinas circulares se eliminan comparándolas con la curvatura de las esquinas candidatas con el umbral adaptativo.

De acuerdo con una región de soporte dinámica recalculada, se evalúan los ángulos de las esquinas candidatas restantes para eliminar cualquier esquinas falsa.

Finalmente, los puntos de fin de los contornos abiertos no se marcan como esquinas hasta que no consideren estar demasiado cerca una esquina de otra.

Para consultar el desarrollo matemático completo favor remitirse al ANEXO A.

4.3.2 Proceso

El proceso de detección de esquinas recibe el *frame* con tonalidad de pixeles original donde sólo se destaca la región que se sustrajo del fondo, y la transforma a una imagen de intensidad. A ésta se le aplica el detector de esquinas de Canny y se obtiene como resultado un mapa binario de esquinas. El paso siguiente es extraer el contorno a partir del mapa y rellenar los huecos que puedan existir en el. Para cada contorno se calcula la curvatura a baja escala para quedarse con todas las esquinas verdaderas. Todos los máximos locales de curvatura se consideran como candidatos a esquinas. Posteriormente se eliminan todas las esquinas redondeadas y todas las falsas esquinas originadas por ruido y, los puntos finales de una línea también son añadidos como esquinas sólo si éstos no están cerca de esquinas superiores detectadas.

Este proceso se lleva a cabo con en el mismo *frame* donde termina el proceso de mezcla de Gaussianas. Se recibe la imagen del objeto más grande recortado de la imagen original y con todas sus propiedades de color; esta imagen es ingresada a la función ***[cout,marked_img]=corner(I,C,T_angle,sig,H,L,Endpiont,Gap_size)*** [28] que se encuentra implementada en el ANEXO CÓDIGO, recibe como entrada la imagen (*I*) ya sea en color o en tono de gris, la relación entre el eje mayor y menor de una elipse (*C*), el ángulo obtuso máximo que una esquina o punto puede tener cuando es detectada como verdadera (*T_angle*), la desviación estándar de un filtro usado para calcular la curvatura (*sig*), los limites inferior y superior del detector de esquinas de *Canny* (*H,L*), una bandera para controlar le decisión de añadir los puntos extremos de una curva como esquinas (*Endpoint*) y un parámetro usado para llenar los vacíos en los contornos (*Gap_size*); como resultado se obtiene una matriz de 2 columnas y N filas (*cout*), cada fila corresponde a un punto de interés y en cada columna se almacena las coordenadas x e y que tiene el punto de interés dentro del *frame*, adicionalmente devuelve una imagen con los puntos de interés marcados (*marked_img*). Por lo anterior, es la matriz de puntos de interés la única salida de este bloque y la entrada al siguiente de construcción de regiones de interés.

4.4 Construcción de ROIs

Este bloque se encaraga de construir las regiones de predilección a partir de la matriz de puntos de interés recibida del bloque anterior. La idea en este punto es la de realizar grupos de puntos teniendo en cuenta características comunes.

El *Clustering* o agrupamiento hace referencia a la clasificación sin supervisión de un conjunto de datos en grupos o *clusters* [17], los datos pertenecientes a cada *cluster* son similares o comparten características. El *clustering* es utilizado en diversos contextos como el análisis de patrones, la toma de decisiones, el aprendizaje artificial, la recuperación de la información, la segmentación de imágenes, la clasificación de patrones, etc.

Como un ejemplo de *clustering* [4], tenemos la recuperación de la información que puede evidenciarse cuando se realiza una búsqueda en la internet mediante un motor de búsqueda.

La estructura jerárquica es uno de los dos tipos de agrupamiento, consiste en un conjunto de agrupaciones anidadas que se organizan como un árbol, cada uno de los *clusters* es la unión de *subclusters*, todos los objetos componen un *cluster* que se asimila a la raíz del árbol. La siguiente estructura de agrupamiento es la estructura particional, donde simplemente los datos son asignados en grupos que no se traslapan entre sí, por lo que cada objeto sólo pertenecerá a un único grupo.

Dentro de los diversos algoritmos existentes para realizar *clustering* tenemos las redes neuronales, clasificadores de Bayes, *k-medias* y *k-Nearest Neighbor*; de los anteriores se eligió el algoritmo *k-means* que cuenta con una estructura de agrupamiento particional. A continuación se describirá el algoritmo utilizado.

4.4.1 Principio del algoritmo k-medias

El algoritmo *k-means* [18] es el algoritmo más popular a la hora de realizar agrupamiento, fue propuesto por *McQueen* en 1967, el asunto de agrupar un conjunto de datos en un número k de grupos o *clusters* tiene diversas aplicaciones como en biología, genética, criminología, estadística, etc. La velocidad computacional del algoritmo es su principal atractivo y a esto debe su popularidad.

El algoritmo requiere de unos valores iniciales que son fundamentales para obtener una solución óptima, la solución óptima se encuentra en función del conjunto de datos, de los grupos y de la distribución del conjunto de datos. La elección de diferentes valores iniciales, conduce a obtener resultados diferentes que no siempre serán los óptimos. De acuerdo al escenario donde se utilice el algoritmo existen diversas formas de condiciones iniciales, la selección de condiciones iniciales arbitrarias es la forma general de inicialización.

El resultado final de cada uno de los k grupos, corresponde a los puntos que minimizan la distancia al centro del k-ésimo grupo. La medida de distancia generalmente es la Euclidiana, pero no resulta raro encontrar implementaciones haciendo uso de la distancia Mahalanobis.

El algoritmo tiene cuatro etapas fundamentales según [18] y [19]; se parte de un arreglo X que cuenta con n puntos y que se quieren clasificar en k grupos distintos, para esto se siguen los pasos que se encuentran a continuación:

1) Se asignan condiciones iniciales a los centros de los k grupos, la elección de los centros se realiza arbitrariamente, los centros se definen como $C_1, C_2, C_3, \dots, C_k$.

2) En la segunda etapa del algoritmo se realizan i iteraciones y se asigna cada X_i, donde $1 \leq i \leq n$, al grupo j, donde $1 \leq j \leq k$ y se escoge teniendo en cuenta que la distancia entre X_i y C_j sea la mínima de las distancias de X_i con los demás centros de *cluster*.

$$\|X_i - C_j\| = \min \|X_i - C_m\|; \quad 1 \leq m \leq k, \text{ donde } j \neq m$$

El operador $\|\cdot\|$ corresponde a la norma euclídea, y se define como la distancia entre dos puntos A y B, en general para un espacio euclídeo de p dimensiones se tiene que

$$\|\overrightarrow{AB}\| = \sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2 + \cdots + (b_p - a_p)^2}$$

En este punto pueden suceder dos situaciones que no son consideradas por el algoritmo; la primera es que ningún punto se asigne a un centro de *cluster* por lo que el *cluster* desaparece y hay un número menor de *clusters*; la otra situación que se puede manifestar en esta etapa

corresponde a aquella donde el punto es igual de cercano a varios centros de *cluster*, para este caso, la asignación se realiza arbitrariamente.

3) La fase siguiente corresponde a actualizar los centros de los *clusters*, encontrando la media o el centro de masa de los puntos que han sido asignados a cada *cluster*, esto es, para cada C_j se hace,

$$C_j = \frac{1}{|C_j|} \sum_{X \in C_j} X$$

Donde $|C_j|$ es el número de elementos que fueron asignados al centro C_j. De ésta fase proviene el nombre del algoritmo ya que se realiza la media de los puntos para calcular el nuevo centro de *cluster*.

4) El último paso consiste en repetir los pasos dos y tres iterativamente hasta que el centro de *cluster* converja o no tenga cambios significativos, cuando esto suceda, el arreglo X queda asignado a cada uno de los centros de *cluster* $C_1, C_2, C_3, \dots, C_k$.

4.4.2 Proceso

La función *kmeans* de MATLAB implementa el algoritmo *k-medias* descrito anteriormente y entrega un vector de n elementos que indica la pertenencia de cada punto a cada uno de los k *clusters*. La función permite además seleccionar el tipo de distancia que se desee utilizar y permite ingresar el número de iteraciones para el algoritmo. Tiene dos fases bien diferenciadas, la primera es la asignación en grupo, donde un conjunto de datos se asigna a cada centro, y en la segunda se hace la asignación individual, donde cada punto se asigna individualmente a cada centro.

La función *kmeans* es la siguiente ***[IDX,Cnt] = kmeans(cout,k)***; tiene dos entradas, una es la matriz de puntos de interés (*cout*) que es la salida del bloque de detección de puntos de interés y la otra entrada es el número de *clusters* o regiones de interés (ROIs) que se quieren construir (*k*), como se dijo en las especificaciones, éste número va de 3 a 5; las salidas de la función son un vector que indica la pertenencia de un punto de interés a uno de los *clusters* (*IDX*) y una matriz de dos columnas y *k* filas que indican las coordenadas del centro de cada uno de los *clusters* (*Cnt*).

La construcción de las regiones de interés se realiza teniendo en cuenta la función de pertenencia de los puntos de interés. Todos los puntos que pertenecen a un mismo *cluster* se encierran en un rectángulo que los contiene a todos quedando así conformada la región de interés (ROI). Cada región de interés se identifica con un color.

En Figura 5 y Figura 6 se muestran los grupos de puntos de interés que corresponden con el proceso de sustracción de fondo mostrado en Figura 3 y Figura 4 respectivamente; cada grupo de puntos se identifica con un color diferente. En estas figuras también se puede apreciar el resultado de la operación de recorte del objeto más grande que corresponde al sujeto en movimiento en ambos casos. En Figura 6 se muestra que cuando hay más de un sujeto y se encuentran separados, el algoritmo escoge uno de los dos sujetos y en ese realiza la detección de puntos de interés durante ese proceso.

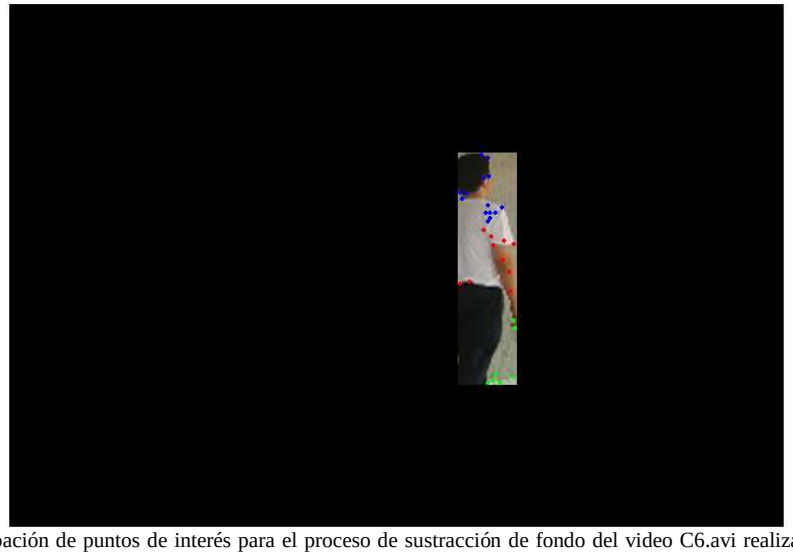


Fig. 5. *Clusters* o agrupación de puntos de interés para el proceso de sustracción de fondo del video C6.avi realizado en el *frame* 5. Cada agrupación corresponde una región de interés.

Fig. 6. *Clusters* o agrupación de puntos de interés para el proceso de sustracción de fondo del video C19.avi realizado en el *frame* 60. Cada agrupación corresponde una región de interés.

4.5 Seguimiento y Predicción

4.5.1 Seguimiento

Con las ROIs construidas, se necesita obtener datos y funciones de interés de sus atributos como son su posición del centro en los ejes coordenados *x* e *y*, *kurtosis*, *skewness* (asimetría) y varianza de las mismas, para esto es necesario ver qué ocurre con cada región durante un periodo de tiempo. Es necesario entonces, implementar un seguimiento o *tracking*, existen diferentes métodos para realizar esta tarea pero debido a la necesidad de optimizar y reducir el costo computacional del algoritmo se decidió implementar el algoritmo planteado en [5].

4.5.1.1 Principio del algoritmo

[17] propone una nueva forma para realizar *tracking* mediante la estimación de movimiento de traslación de la imagen usando suma de diferencias de cuadrados (SSD). El modelo propuesto es muy sencillo y es ampliamente utilizado en procesamiento de video.

En [17] la estimación de movimiento consiste en la definición de vectores de movimiento que indica los desplazamientos en las coordenadas del *frame* y la búsqueda de un vector óptimo que produzca el mínimo error. Por otra parte, la correlación cruzada (XCORR) es usada en el *matching* de imágenes

desplazadas ya que posee grandes ventajas como, independencia de la iluminación y sombras, y permite realizar la estimación de grandes desplazamientos de la región.

El *frame* actual se divide en bloques cuadrados de $B \times B$ y se asume que cada uno de los bloques presenta movimiento de traslación por lo que es posible calcular un vector de movimiento, que describa el desplazamiento en las dos coordenadas. Para un bloque g que pertenece al *frame* actual, se realiza una búsqueda en el *frame* de referencia sobre un área de búsqueda f y con una ventana de búsqueda de tamaño fijo. El objetivo de la búsqueda es encontrar el bloque que mejor se ajuste y cuyo error sea mínimo. Para este algoritmo esto se consigue minimizando la suma de diferencia de cuadrados.

Para consultar el desarrollo matemático y la base teórica de este procedimiento, consultar el ANEXO A.

4.5.1.2 Proceso

Cómo se dijo anteriormente el algoritmo es adaptativo y para esto se hace necesario que los procesos sean periódicos permitiendo así la actualización de la información. Ya se ha explicado que los cinco primeros se usan para realizar la sustracción de fondo y el último *frame* de estos se usa para la detección de puntos de interés y construcción de las regiones de interés a partir de dichos puntos. También se habló de los atributos de las ROIs y que son éstos los que generan la información durante el tiempo, es aquí donde se ve la necesidad de realizar un seguimiento a la región de interés en la línea de tiempo.

El tamaño de la ventana de diez *frames* se definió realizando pruebas de *tracking;* como este algoritmo utiliza correlación se debe definir hasta qué punto el algoritmo resulta óptimo, evitando equivocaciones o resultados negativos. De las pruebas se obtuvo que el algoritmo presenta resultados confiables durante cinco *frames* (avanzando en la línea de tiempo y retrocediendo en la línea de tiempo) del *frame* que se toma como referencia, es decir, que si las regiones de interés se encuentran en el *frame* K, los resultados del tracking a estas regiones son confiables desde $K - 5$ hasta $K + 5$.

Para realizar el seguimiento se utilizó el algoritmo descrito anteriormente que se encuentra implementado en la función ***[S,c,r,w,h] = SSDXCORR(f,t)*** [29] del ANEXO CÓDIGO, tiene dos entradas que son la imagen que se quiere buscar o imagen de referencia (*t*) que para nuestro caso es la región de interés y la imagen donde se quiere encontrar esa imagen de referencia (*f*) que corresponde con el nuevo *frame,* y tiene como salidas las coordenadas del centro de la región de interés (*c* y *r*) y el ancho y alto de la región (*w* y *h*), las coordenadas del centro y de las dimensiones son usadas para la ubicación de la ROI en el *frame* que está siendo analizado para la extracción de sus atributos.

Para garantizar la confiabilidad de los resultados del *tracking* se decidió implementar un seguimiento previo (*pre-tracking*) al de las regiones de interés, se resolvió así al analizar los resultados de pruebas realizadas que mostraban que si las regiones de interés son muy pequeñas y el fondo es complejo se producían errores en el seguimiento generando así datos erróneos de los atributos de las mismas. Con el *pre-tracking* se tiene una región de referencia más grande lo que reduce el margen de que el algoritmo falle a la hora de buscarla dentro del *frame*. El *pre-tracking* toma como referencia el objeto más grande recortado y con sus propiedades de color, como se dijo, se obtiene en el quinto *frame* de la ventana. El *pre-tracking* va desde el primer *frame* hasta el último de la ventana. El *tracking* de las regiones de interés se da dentro del resultado arrojado por el *pre-tracking* para cada *frame*, es decir que la región de interés ya no se busca en todo en *frame* sino que queda limitada a una región menor.

Una vez localizada la región de interés en cada uno de los *frames* se procede a obtener sus atributos. Éstos se van a almacenar en dos vectores diferentes que sirven para guardar toda la información ya que la ventana al moverse queda traslapada con la anterior y si sólo se tuviera un vector para almacenar la información se sobrescribiría la información y por ende se perdería. Los datos de los atributos de las ROI de los cinco primeros *frames* de la ventana se almacenan en la tercera fila de cada una de las matrices de atributos correspondientes a cada ROI, éstos serán usados posteriormente para realizar la

predicción. Los datos de los últimos cinco *frames* de la ventana se almacenan en la primera fila de cada una de las matrices de atributos correspondientes a cada ROI.

A continuación se presentan secuencias donde se puede observar el *pre-tracking* y *tracking* correspondiente a las ventanas que se han venido mostrando anteriormente. En la Figura 7 y en la Figura 9 se observa el *pre-tracking* realizado en los videos C6.avi y C19.avi respectivamente, donde en rojo se encuentra encerrado el resultado. En la Figura 8 y en la Figura 10 se observa el *tracking* de las regiones de interés de las mismas secuencias; cada rectángulo indica una región diferente y como se dijo, conservan los mismos colores que se había elegido en el bloque de construcción de regiones de interés. En las secuencias se puede observar el cambio el cambio de tamaño de las diferentes regiones que se da cuando se recalculan las regiones de interés.

Fig. 7. *Pre-tracking* realizado en una ventana de 10 *frames* para el video C6.avi: (a) *Pre-tracking frame* 1, (b) *Pre-tracking frame* 2, (c) *Pre-tracking frame* 3, (d) *Pre-tracking frame* 4, (e) *Pre-tracking frame* 5, (f) *Pre-tracking frame* 6, (g) *Pre-tracking frame* 7, (h) *Pre-tracking frame* 8, (i) *Pre-tracking frame* 9, (j) *Pre-tracking frame* 10.

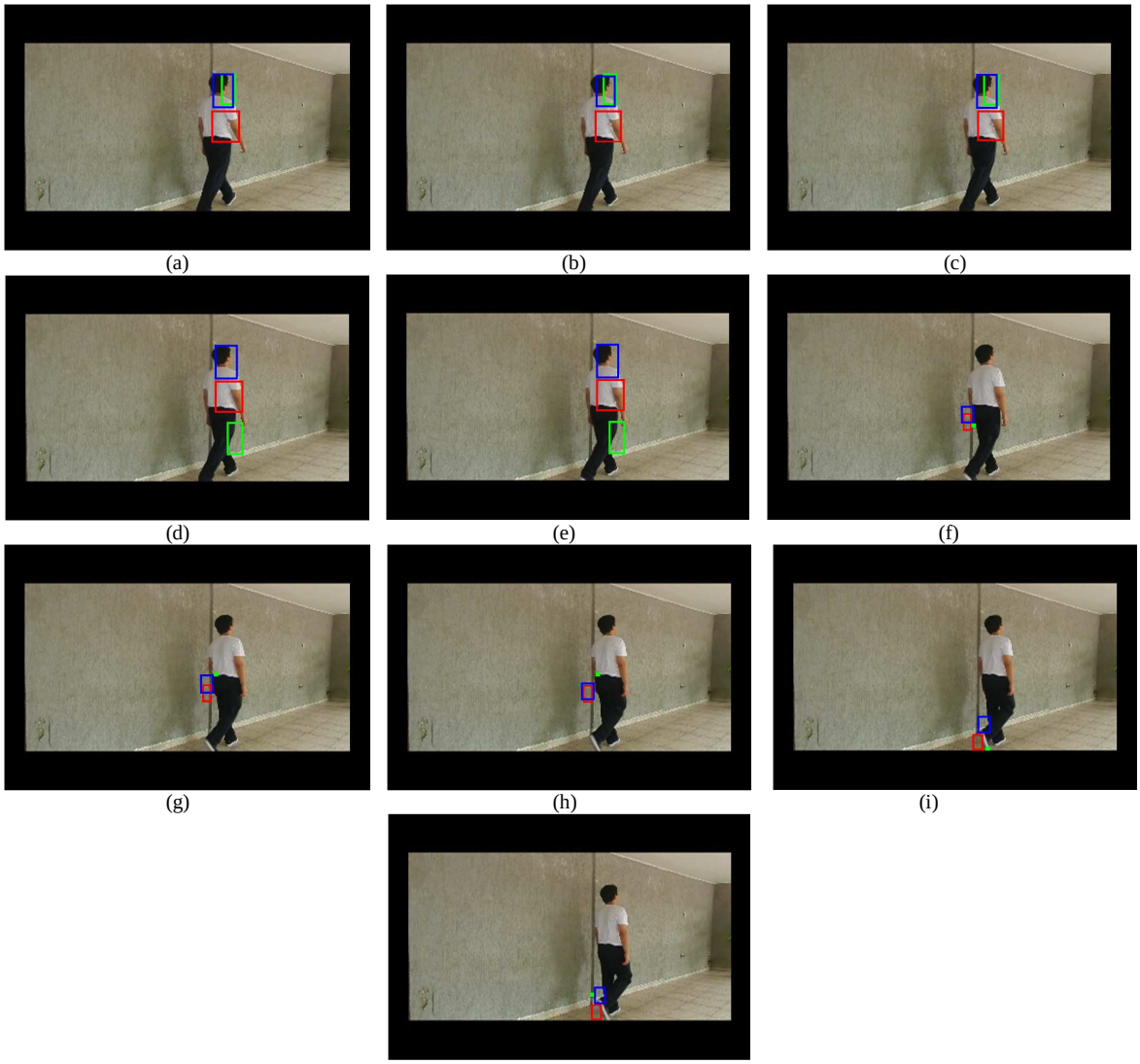
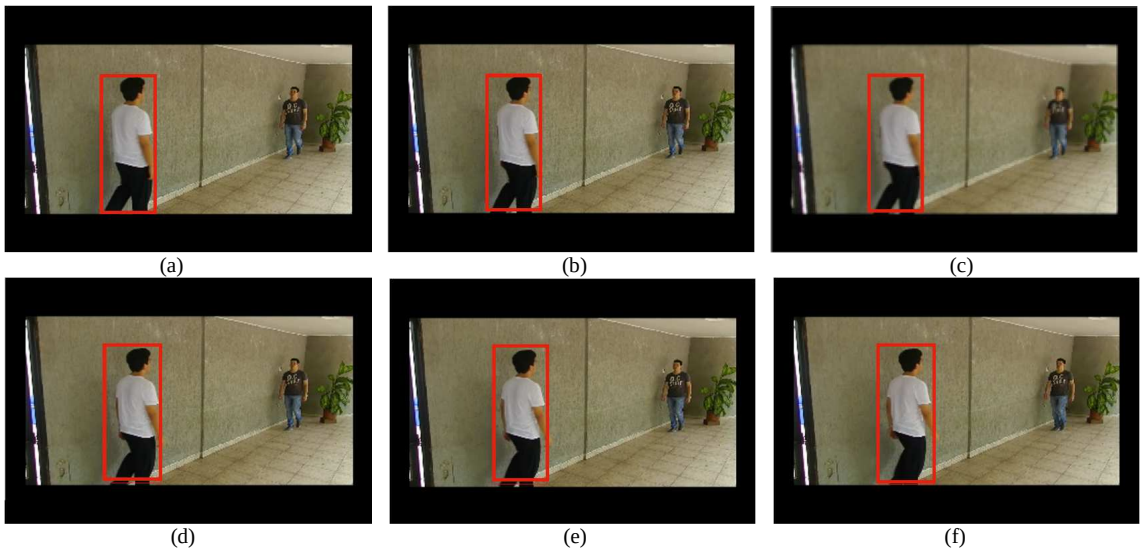
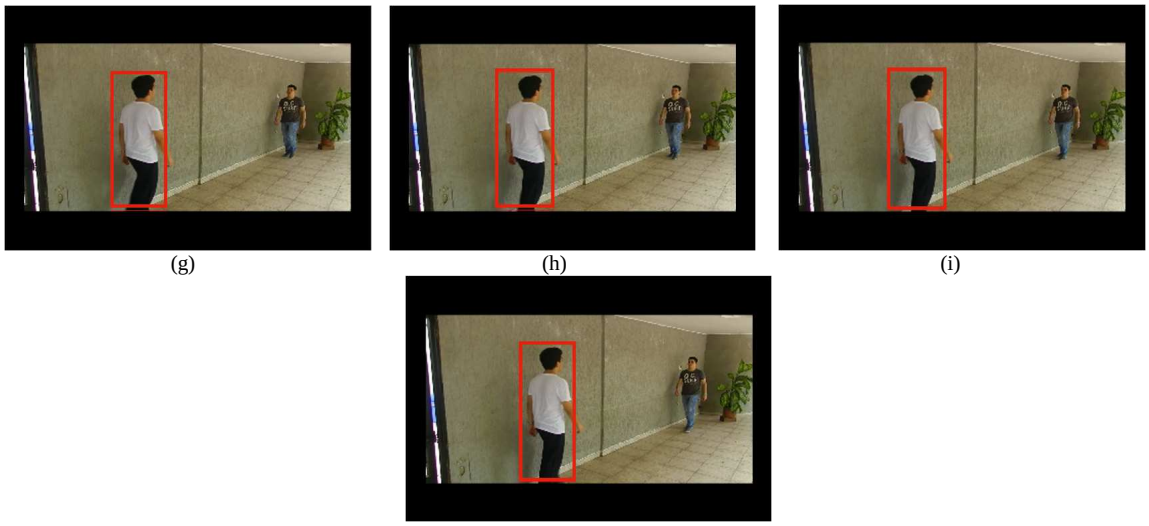


Fig. 8. *Tracking* realizado a las regiones de interés en una ventana de 10 *frames* para el video C6.avi: (a) Regiones de interés *frame* 1, (b) Regiones de interés *frame* 2, (c) Regiones de interés *frame* 3, (d) Regiones de interés *frame* 4, (e) Regiones de interés *frame* 5, (f) Regiones de interés *frame* 6, (g) Regiones de interés *frame* 7, (h) Regiones de interés *frame* 8, (i) Regiones de interés *frame* 9, (j) Regiones de interés *frame* 10.

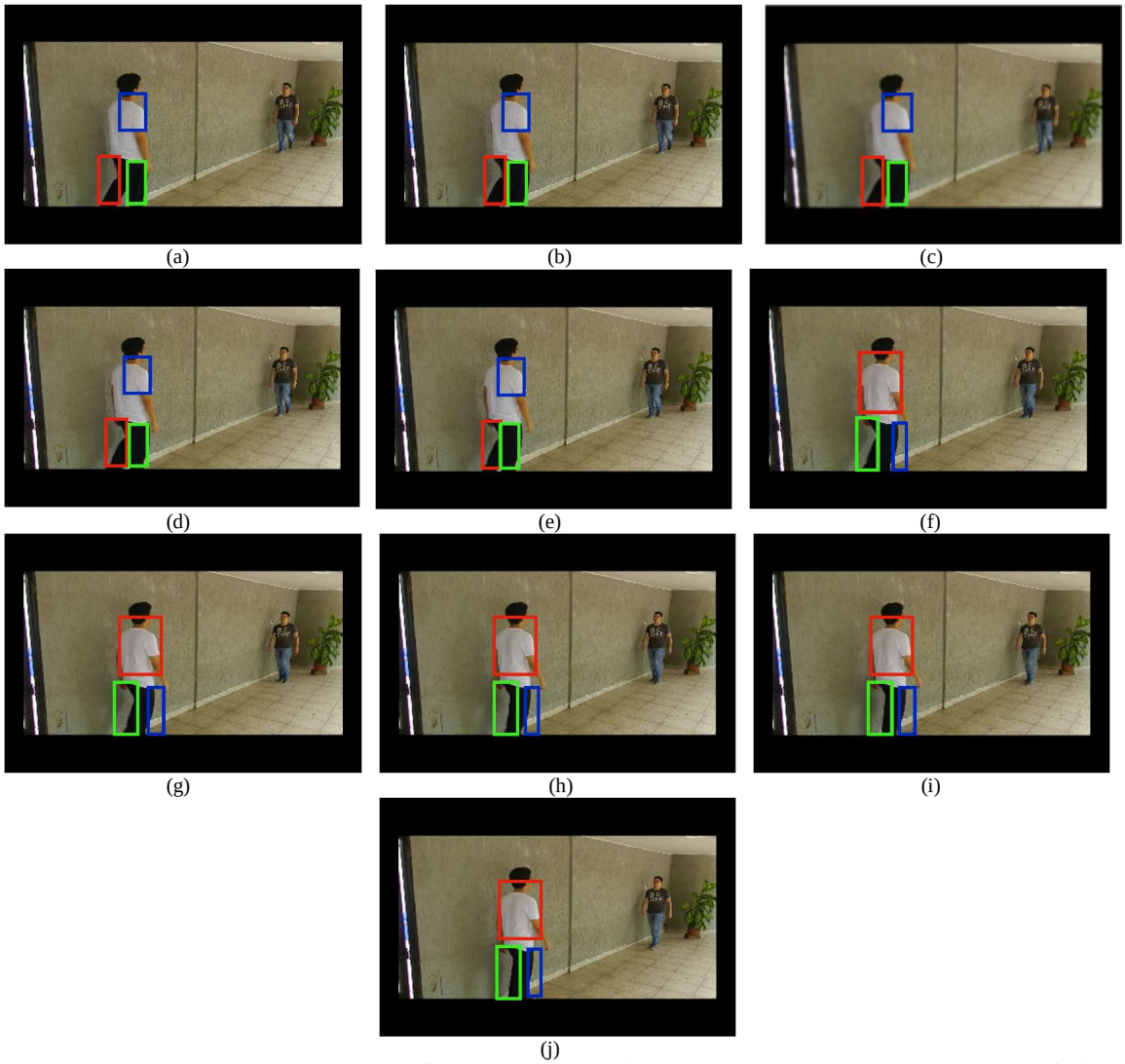




(g) (h) (i)

(j)

Fig. 9. *Pre-tracking* realizado en una ventana de 10 *frames* para el video C19.avi: (a) *Pre-tracking frame* 56, (b) *Pre-tracking frame* 57, (c) *Pre-tracking frame* 58, (d) *Pre-tracking frame* 59, (e) *Pre-tracking frame* 60, (f) *Pre-tracking frame* 61, (g) *Pre-tracking frame* 62, (h) *Pre-tracking frame* 63, (i) *Pre-tracking frame* 64, (j) *Pre-tracking frame* 65.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j)

Fig. 10. *Tracking* realizado a las regiones de interés en una ventana de 10 *frames* para el video C19.avi: (a) Regiones de interés *frame* 56, (b) Regiones de interés *frame* 57, (c) Regiones de interés *frame* 58, (d) Regiones de interés *frame* 59, (e) Regiones de interés *frame* 60, (f) Regiones de interés *frame* 61, (g) Regiones de interés *frame* 62, (h) Regiones de interés *frame* 63, (i) Regiones de interés *frame* 64, (j) Regiones de interés *frame* 65.

4.5.2 Predicción

El proceso de predicción también se desarrolla en este bloque, su función es la de realizar la predicción del comportamiento de datos siguientes a partir de una serie de datos de entrada. El propósito de tener una señal resultado de una predicción es el de compararla con una señal de datos y establecer posibles diferencias entre las dos. Para realizar la predicción se utiliza un predictor lineal hacia adelante y su principio se explica en la siguiente sección.

4.5.2.1 Principio del algoritmo

En [22] se muestra un predictor lineal, éste se compone de un filtro con *M tabs*, cada uno con pesos $w_1, w_2, \dots, w_M$ y con entradas $u(n-1), u(n-2), \dots, u(n-M)$, como se puede observar en la Figura 11. El objetivo de este filtro es el de obtener una muestra futura a partir de un número de muestras pasadas, es decir que es un sistema causal. El orden *M* del predictor lineal viene dado por el número de elementos de retardo necesarios para almacenar el conjunto de muestras usadas para hacer la predicción.

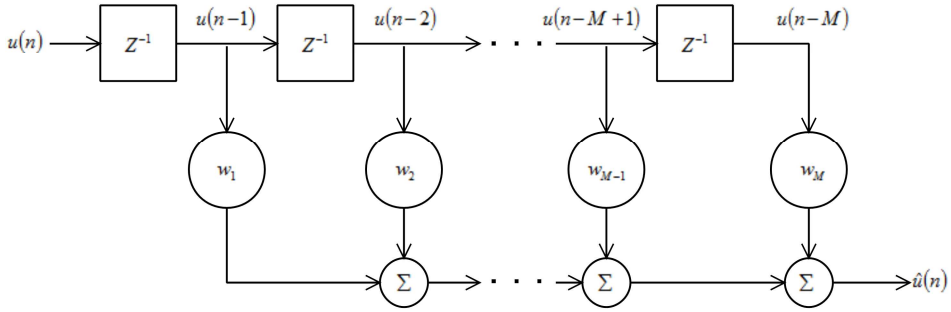


Fig. 11. Filtro usado para implementar un predictor lineal hacia delante utilizando la segunda forma traspuesta.

La salida del sistema $\hat{u}(n)$ está dada por la siguiente expresión

$$\hat{u}(n) = \sum_{k=1}^{M} w_k u(n-k)$$

Por último el error de predicción viene dado por la diferencia entre la muestra de entrada y su valor predicho, es decir

$$e_M = u(n) - \hat{u}(n)$$

4.5.2.2 Proceso

Para el proceso de predicción se utilizan funciones ya implementadas en MATLAB. La primera que se utiliza es ***coef = lpc(x,p)***, que calcula los coeficientes de un predictor lineal hacia delante; recibe dos entradas que son los datos anteriores al dato que se quiere predecir (*x*) y el orden del predictor lineal (*p*) y arroja como resultado un vector que contiene los coeficientes (*coef*) para la implementación de un filtro que corresponde con un predictor lineal. La segunda función se usa para la implementación del predictor, ésta es la función ***y = filter(b,a,x)***, ésta función implementa un filtro usando la segunda forma traspuesta (Direct Form II Transposed) de una ecuación de diferencias estándar. La función recibe como entrada la secuencia de datos, los coeficientes de promedio móvil y los coeficientes auto-regresión y arroja como resultado la señal o dato estimado.

Las secuencias que son usadas en estas funciones corresponden con los datos almacenados en los 5 *frames* anteriores del *frame* que se quiere calcular el dato, para esto se calculan los coeficientes y se ingresan al filtro para calcular el dato en el *frame* siguiente, es decir que si se quiere calcular un dato determinado en el *frame* 6 se deben enviar los datos de los 5 *frames* anteriores. Este proceso se lleva a cabo con todos los atributos de cada región de interés. Como se dijo anteriormente, hay una fila en

cada matriz de datos donde se almacenan los resultados de la predicción; como es evidente los cinco primeros *frames* de un video no almacenan datos de predicción. En este punto tenemos dos vectores de datos que a partir del quinto *frame* de cada video almacenan información y son este par los que se van a comparar para ver las diferencias existentes, dependiendo de su amplitud nos permiten identificar anomalías en los atributos.

Para relacionar las dos señales, la de datos y la de predicción, se realiza la diferencia de cada punto y el resultado obtenido se eleva al cuadrado para evitar valores negativos, obteniendo así una función de error o diferencia para cada atributo de cada región de interés. En la Figura 12 se puede observar un atributo para una región de interés del video C6.avi, se muestran los datos de Desplazamiento Horizontal, en Figura 12 (a), la señal resultado de la predicción en Figura 12 (b) y en la Figura 12 (c) se observa la señal de error correspondiente. En esta figura se puede observar que la señal de predicción sigue acertadamente a la señal de datos salvo algunas diferencias de amplitud.

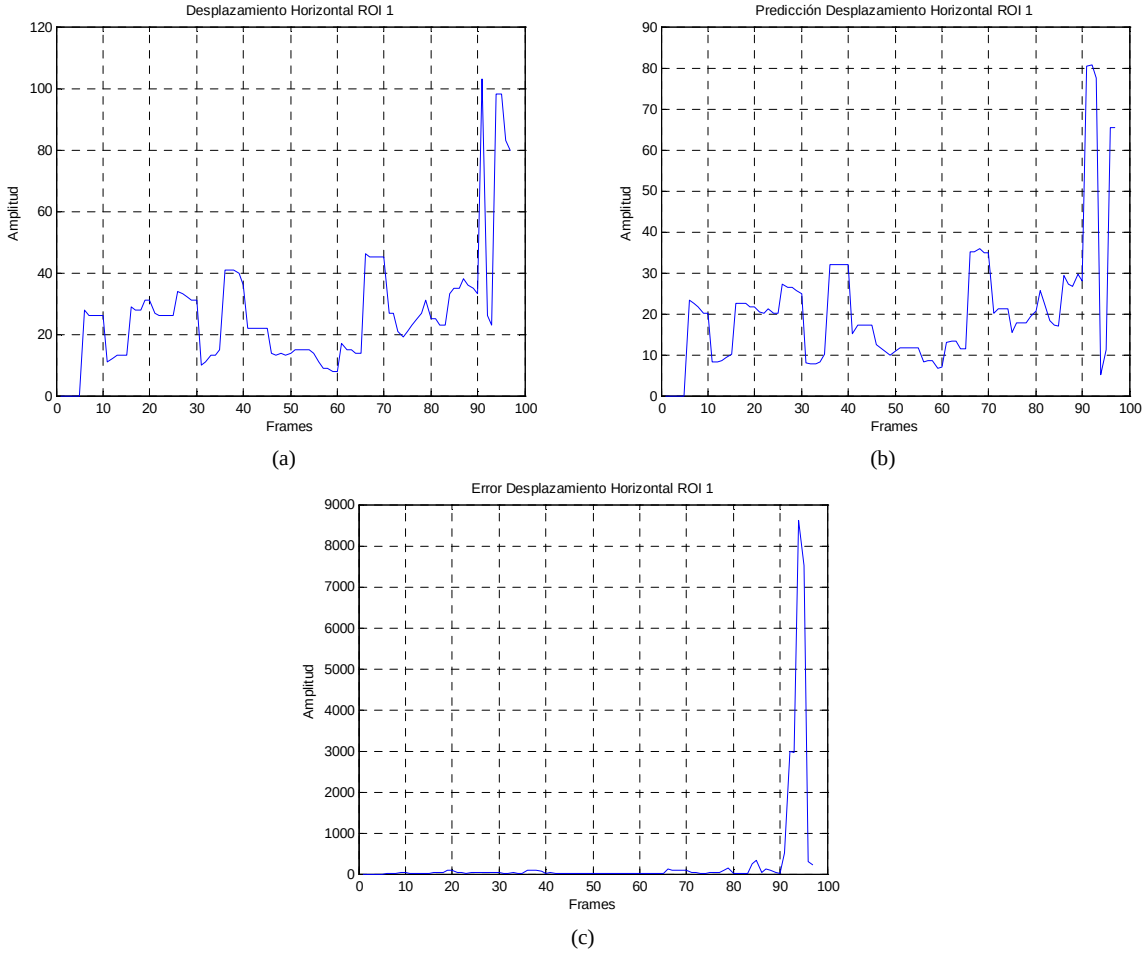


Fig. 12. Datos de Desplazamiento Horizontal ROI 1: (a) Datos obtenidos de seguimiento, (b) Datos obtenidos de predicción, (c) Señal de error

La última parte del proceso es relacionar atributos utilizando sus señales de error; de cada ROI se tienen los mismos atributos, ahora para relacionarlos se realiza una suma de los atributos, es decir, se promedian las señales de error salvo por un factor de escala de desplazamiento, varianza, *skewness* y *kurtosis* de las ROIs, obteniendo finalmente cinco señales correspondientes a cada uno de los atributos, siguiendo el proceso descrito estas señales relacionan los atributos en las diferentes ROIs y a su vez las señales de datos y predicción. En la Figura 13 (a), (b) y (c) se muestra las señales de error del atributo Desplazamiento Horizontal en las regiones de interés (tres regiones de interés por defecto) y en la Figura 13 (d), se muestra el resultado de la suma de éstas.

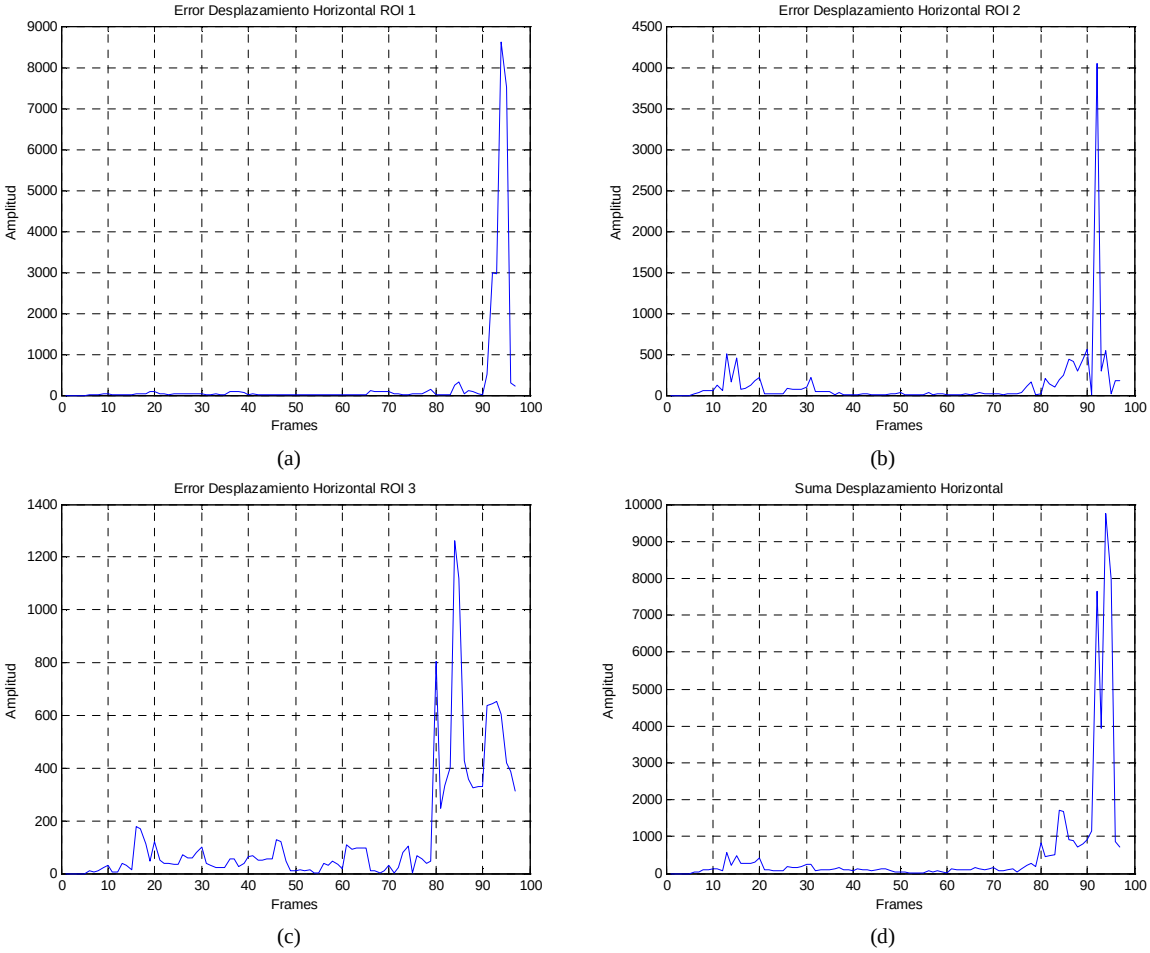


Fig. 13. Datos de Desplazamiento Horizontal: (a) Señal de error ROI 1, (b) Señal de error ROI 2, (c) Señal de error ROI 3, (d) Suma de atributos

4.6 Decisión

El último paso del proceso es el de decisión, es aquí donde se discrimina entre comportamientos anómalos y normales. La discriminación debe hacerse durante todo el video y cada *frame* es evaluado para determinar su comportamiento. El proceso de decisión involucra varias tareas que se encuentran dentro del circuito de decisión indicado en la Figura 14**.**

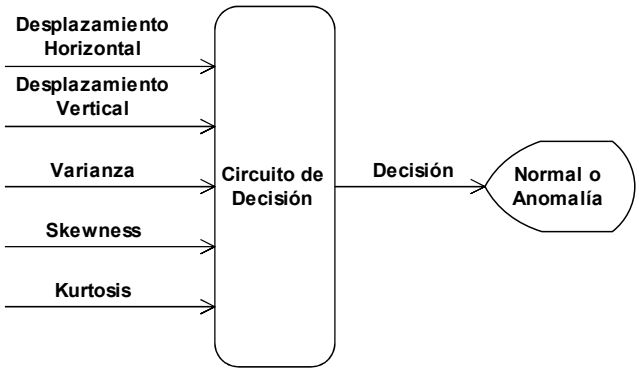


Fig. 14. Circuito de decisión para detección de anomalías; recibe como entradas la suma de todos los atributos y arroja la decisión acerca de la existencia o no de anomalía en el *frame.*

El circuito recibe como entradas las señales de suma de cada uno de los atributos y arroja como única salida la decisión de si el *frame* analizado presenta o no. Para determinar anomalías en cada atributo se decidió utilizar un umbral y un factor de escala, el uso del umbral es muy sencillo, para cada atributo del *frame* en análisis se revisa si el dato supera o no el umbral, si lo supera el dato presenta anomalía,

el factor de escala se utiliza para ajustar el umbral, su valor por defecto es la unidad, y al aumentarlo el algoritmo se vuelve más selectivo como se verá en la sección de análisis de resultados.

Ahora bien, como la información se renueva mediante la actualización de las regiones de interés, si se selecciona un umbral fijo los resultados no serán confiables o el algoritmo se va a limitar a un grupo de secuencias de video que cumplan con ciertas características.

Se decidió entonces utilizar un umbral que se actualice periódicamente al igual que todos los procesos, éste se realiza en la misma ventana de 10 *frames*; cuando se llega al último *frame* de la ventana y conociendo que los datos guardan relación durante 5 *frames* se calcula el máximo de los últimos 5 *frames* donde se incluye el último y éste este valor se asigna como umbral. El umbral es válido durante los 5 *frames* siguientes donde al llegar al último se actualiza nuevamente. El umbral se calcula únicamente sobre las señales de suma de atributos.

Para decidir si el *frame* tiene un comportamiento anómalo se tienen en cuenta los resultados de si el dato supera el atributo, se tienen cinco atributos, si mínimo tres de ellos presentan anomalía se dice que el *frame* tiene un comportamiento anómalo, de lo contrario se clasifica como normal.

5. ANÁLISIS DE RESULTADOS

Para realizar el análisis del trabajo desarrollado se hizo necesario la generación de diversos videos donde se mostraran actividades poco usuales en el comportamiento humano como saltos, caídas, peleas entre sujetos y movimientos erráticos que permitieran discriminar de forma adecuada entre un movimiento normal y uno anómalo.

La toma de los videos se hizo en dos tipos de ambientes diferentes: exteriores e interiores. Para el caso de los videos realizados en interiores, se realizaron en casas de familia con iluminación artificial controlada donde el dispositivo de captura se encuentra fijo en un trípode cerca al sujeto debido al limitado espacio. Se realizaron dos tipos tomas, tanto perpendicular como diagonal de frente al/los sujeto/tos. Para el caso de los videos en exteriores se realizaron a las afueras y en las calles teniendo iluminación natural y de la misma manera el dispositivo de captura fijo en un trípode. Éstas tomas fueron más amplias y se realizaron capturas diagonales de frente y de espaldas al/los sujeto/tos

El método propuesto se probó con videos que recreaban ambientes diversos, tanto en interiores como en exteriores, con ángulos de cámara diferentes y con presencia de hasta dos sujetos en la escena.

Atributos como el tono promedio, varianza, *kurtosis*, asimetría o *skewness*, velocidad, aceleración, posición horizontal y vertical del centro de las regiones de interés fueron analizados y estudiados para determinar qué atributos eran los que mayor información aportaban con el fin de detectar anomalías en la secuencia de *frames.* El desempeño de cada atributo dependía de los movimientos que se registraron en los videos de prueba, la calidad de los mismos y el ambiente en el que se desarrollaba la escena. Los atributos que mejor desempeño obtuvieron fueron los de asimetría o skewness, *kurtosis*, varianza que son medidas estadísticas diferentes por ser de orden distinto y posición tanto vertical como horizontal del centro de las regiones de interés que describen el movimiento del sujeto/s en la escena.

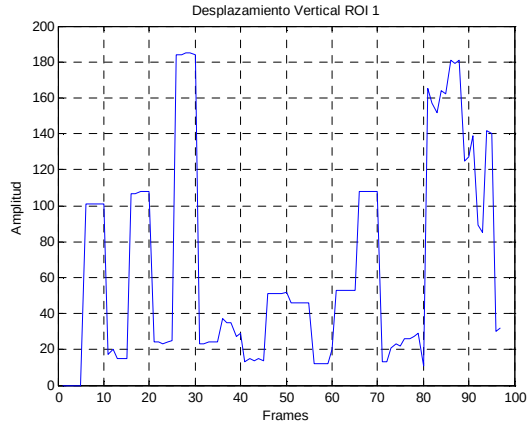
5.1 Caso de Análisis

Nombre video: C6.avi, Tasa de *Frames*: 30 *frames*/s, Longitud de ventana: 10 *frames*., Número de, ROIs: 3, Orden del predictor: 2, Factor de umbral: 1,0.

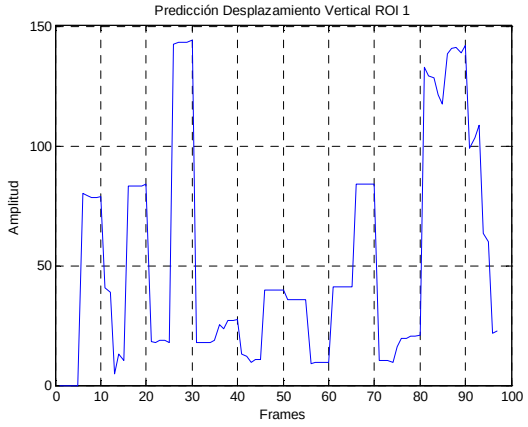
Las siguientes imágenes corresponden a los resultados obtenidos para el Video C6.avi. En la secuencia de video, se generan las medidas de los atributos seleccionados para cada *frame*. En este video se presenta un sujeto en la escena caminando normalmente hasta el *frame* 39 de la secuencia, a partir del *frame* siguiente hasta el 65 yace parado evaluando la escena, desde el *frame* 76 emprende una huida espontánea de la escena hasta finalizar la secuencia de video. El video cuenta con un total de 97 cuadros. En teoría podemos determinar qué cuadros se catalogan como anómalos y cuáles no. Desde el

frame 1 hasta el 75 se catalogan como *frames* sin anomalía dada la razón que el criterio de anormalidad no se ajusta a lo que realiza el sujeto en estos *frames*, sin embargo a partir del *frame* 76 incurre en anormalidad y por consiguiente los *frames* del 76 al 97 se dictaminan que poseen anomalía.

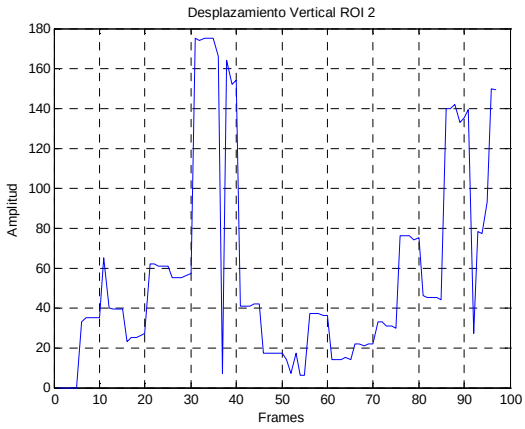
Para el atributo de desplazamiento vertical en la ROI 1 de la Figura 15 (a) (b), se observa que la amplitud en los *frames* alrededor del 28 y 85 es elevada comparativamente con la amplitud que tomaron los demás *frames*. Ese mismo comportamiento se registró en la predicción. El cálculo del error posibilita que se identifique con mayor exactitud la ubicación de las anomalías, ya que en los *frames* donde se registraron valores de amplitud elevado no exactamente significa que las posean. Como se puede ver en Figura 15 (d), el predictor no va a ser capaz de prever el valor de un dato que tomará un valor errático o brusco, pero trata de ajustarse al máximo a dicho valor, sin embargo nunca alcanzará a predecirlo exactamente, lo que permite discriminar de forma más selectiva y con mayor sustento teórico la presencia de anomalías. Esto lo podemos ver en los alrededores del *frame* 28, donde eventualmente se registraban como anomalías en Figura 15 a pesar que en el *ground truth* no lo eran, pero posterior al cálculo del error en Figura 16 con ayuda de la predicción fueron filtrados y posibilitó una detección de anomalía correcta.



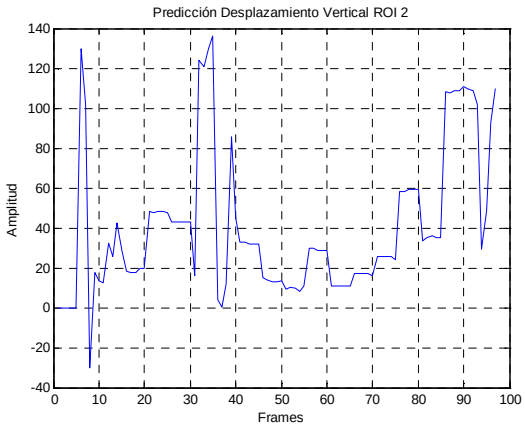
(a)



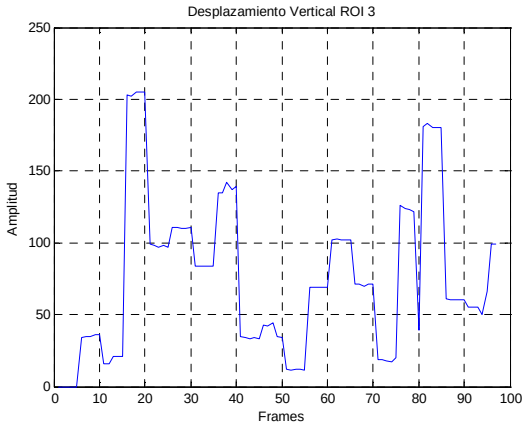
(b)



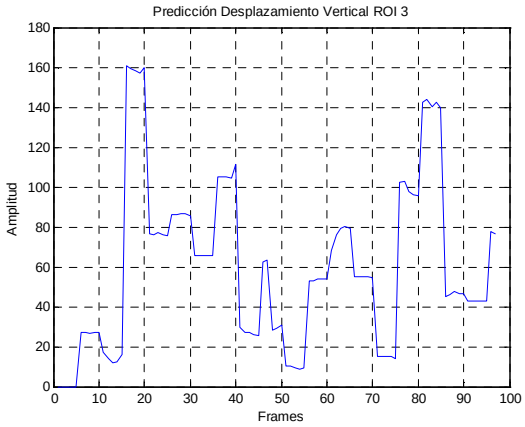
(c)



(d)

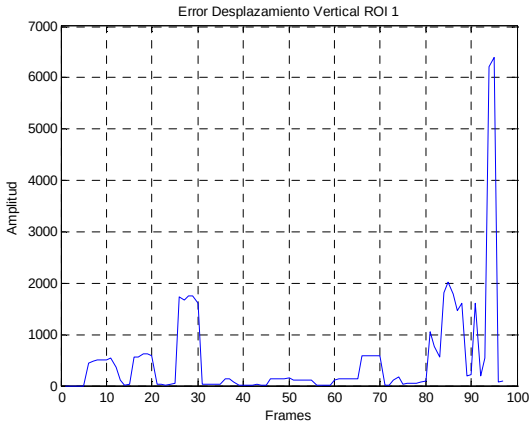


(e)

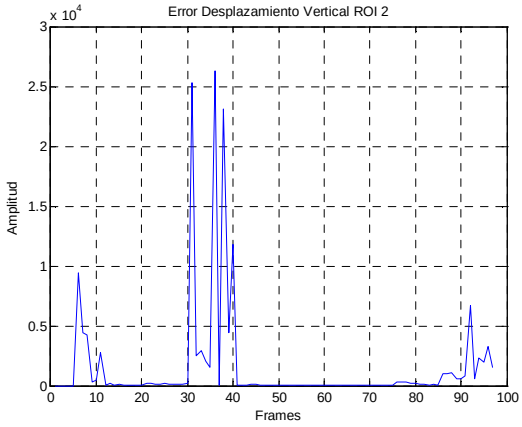


(f)

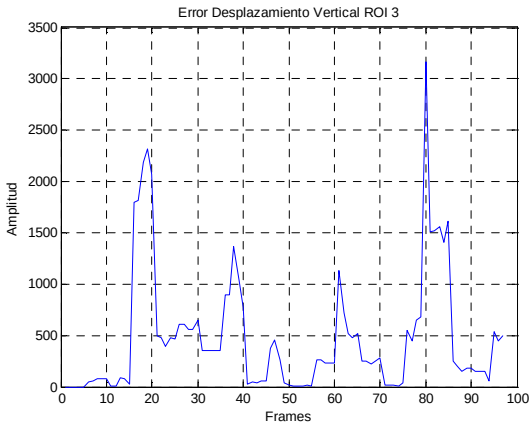
Fig. 15 Resultados correspondientes al atributo Desplazamiento Vertical para el video C6.avi: (a) Resultado obtenido del tracking para la ROI 1, (b) Resultado obtenido de la predicción para la ROI 1, (c) Resultado obtenido del tracking para la ROI 2, (d) Resultado obtenido de la predicción para la ROI 2, (e) Resultado obtenido del tracking para la ROI 3, (e) Resultado obtenido de la predicción para la ROI 3.



(a)



(b)



(c)

Fig. 16 Señales de error para el atributo Desplazamiento Vertical en las diferentes regiones de interés: (a) Señal de error de Desplazamiento Vertical para ROI 1, (b) Señal de error de Desplazamiento Vertical para ROI 2, (c) Señal de error de Desplazamiento Vertical para ROI 3.

Algunos atributos funcionan mejor en unos escenarios que en otros, y este es el caso para el desplazamiento horizontal. El desplazamiento horizontal para este tipo de videos ofrece información útil que describe perfectamente el recorrido del sujeto durante el video. El sujeto se desplaza en la escena de izquierda a derecha caminando normalmente hasta el *frame* 39, esto se evidencia en la Figura 17 (a), donde el desplazamiento horizontal para la ROI 1 toma valores relativamente altos y constantes, esto también se observa para la ROI 2 y para la ROI 3 en la Figura 17 (c) y (e) respectivamente aunque para la ROI 3 no es tan evidente, esto debido a la elección en determinados instantes de la ROI y a la detección de la misma ya que no fue la mejor, porque como se observa en la Figura 8 (f) a (j), el área de las regiones de interés del video C6.avi fueron pequeñas y especialmente la

región marcada en color verde. La disminución en la amplitud de los valores desde el *frame* 40 hasta el 65 se debe a que la actividad del sujeto es casi nula en estos *frames*, ya que yace parado evaluando la escena y las regiones de interés nos experimentan cambios posicionales.

En contraste, en los *frames* 76 a 97 se registraron las mayores amplitudes, coherente con lo sucedido en la secuencia del video, donde el sujeto en este intervalo de *frames* bruscamente pasa de un estado de reposo en el que evaluaba la escena a correr repentinamente hasta finalizar la misma por lo que el desplazamiento horizontal tomó valores grandes.

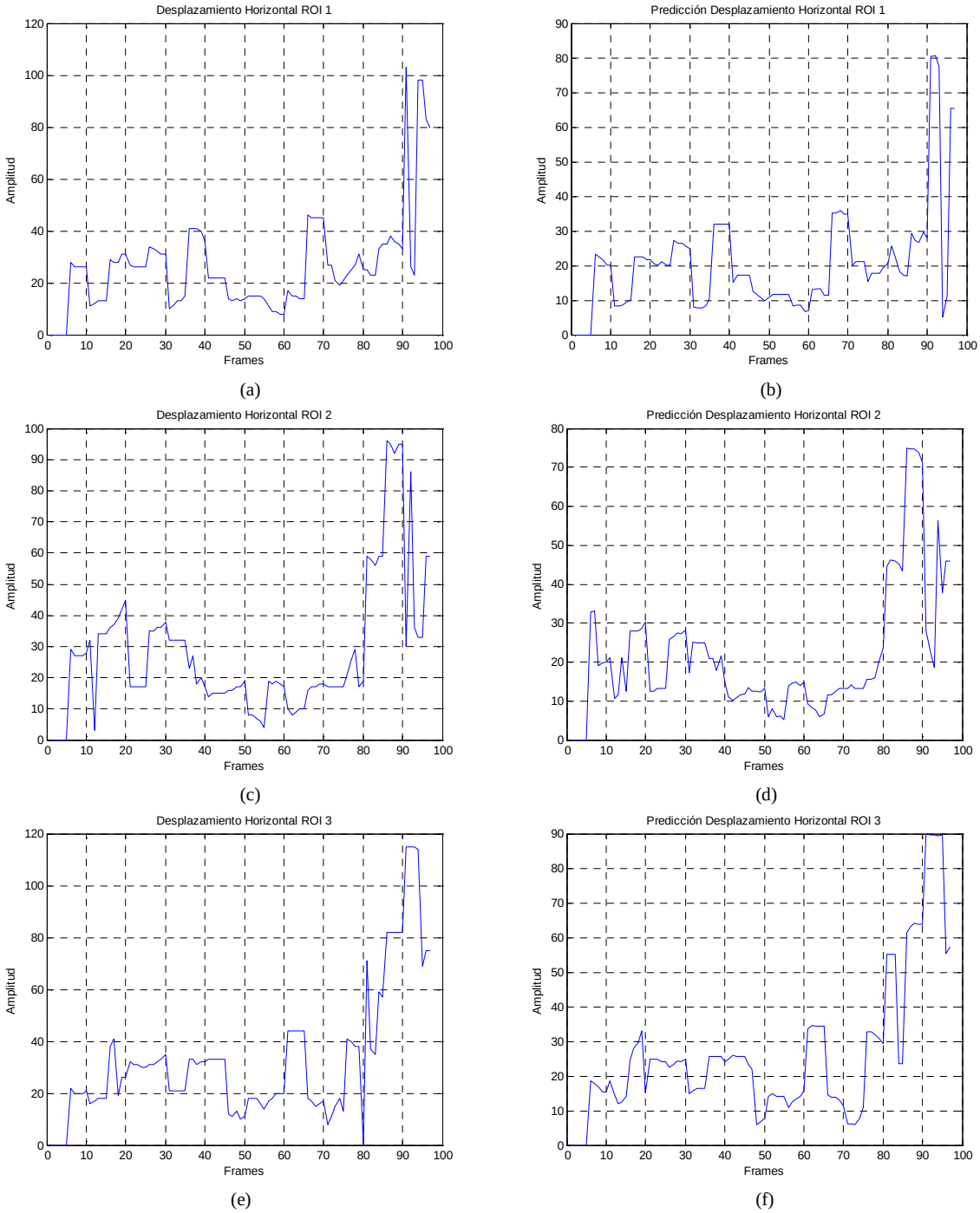


Fig. 17 Resultados correspondientes al atributo Desplazamiento Horizontal para el video C6.avi: (a) Resultado obtenido del tracking para la ROI 1, (b) Resultado obtenido de la predicción para la ROI 1, (c) Resultado obtenido del tracking para la ROI 2, (d) Resultado obtenido de la predicción para la ROI 2, (e) Resultado obtenido del tracking para la ROI 3, (e) Resultado obtenido de la predicción para la ROI 3.

El cálculo de los errores para cada región de interés para el atributo de desplazamiento horizontal fue acorde a lo sucedido en la secuencia de video y a lo que se describe en el *ground truth*. En cada una de las figuras mostradas en Figura 18 se registraron valores considerables de amplitud, lo que describe la

repentina huida del sujeto en la escena, discriminando acertadamente los *frames* donde se presentan eventuales anomalías.

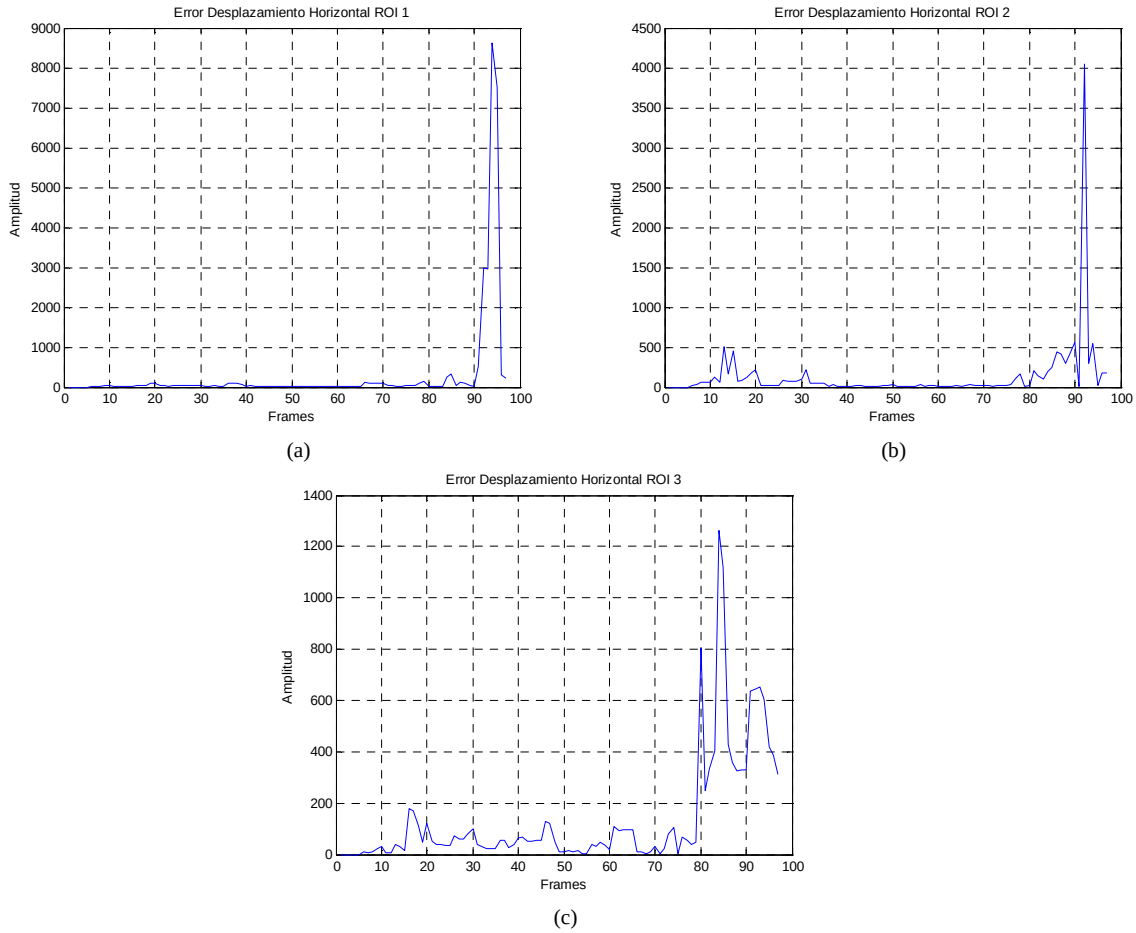


Fig. 18 Señales de error para el atributo Desplazamiento Horizontal en las diferentes regiones de interés: (a) Señal de error de Desplazamiento Horizontal para ROI 1, (b) Señal de error de Desplazamiento Horizontal para ROI 2, (c) Señal de error de Desplazamiento Horizontal para ROI 3.

Como es bien sabido, la varianza es uno de los indicadores de dispersión más importantes porque proporciona una medida acerca de la variabilidad de un conjunto de datos. En la Figura 19 (a), (c) y (e) se muestra el resultado del cálculo de la varianza para todas las regiones de interés del video C6.avi. El cálculo es realizado posicionando el valor de cada uno de los pixeles en tono de gris dentro de la región de interés en un vector, y es a este al que se le calcula la varianza.

La varianza en este caso no brinda información determinante exceptuando el resultado obtenido para la ROI 2 de la Figura 19 (c), donde se obtienen picos en los valores de amplitud para los *frames* donde se estipuló que existía anomalía, esto es, en los *frames* 76 a 97.

Al observar las regiones de interés marcadas por el algoritmo, se observa que en algunos intervalos de la secuencia de video éstas poseen áreas reducidas, lo que representa un problema al momento de detectar en el siguiente *frame* la misma región ya que la probabilidad de encontrar una región con similares características (igual tono de los pixeles) es elevada, por lo que podría incurrir en error. Un ejemplo de esto se observa en la Figura 21 (a) a (e), donde se muestra que las regiones generadas en el video C6.avi del *frame* 41 a 45 fueron pequeñas e incluso la región etiquetada en color rojo lo fue en extremo, igualmente, en la Figura 21 (f) a (j), la región etiquetada en color verde de *frame* 6 al 10 mostró la misma deficiencia, lo que conllevaría a que se registren resultados inconsistentes respecto a la descripción del video hecha en el *ground truth*.

Sin embargo, y como se especificó anteriormente, el cálculo del error posibilita que errores surgidos durante el *tracking* y la generación de la regiones de interés se vean minimizados ante la imposibilidad del predictor de augurar futuros valores que cambian bruscamente en el tiempo. Esto se ve perfectamente en los resultados obtenidos de los errores obtenidos para el atributo de varianza en la

Figura 20 (a), (b) y (c), donde el algoritmo registró valores elevados en los *frames* marcados como anómalos.

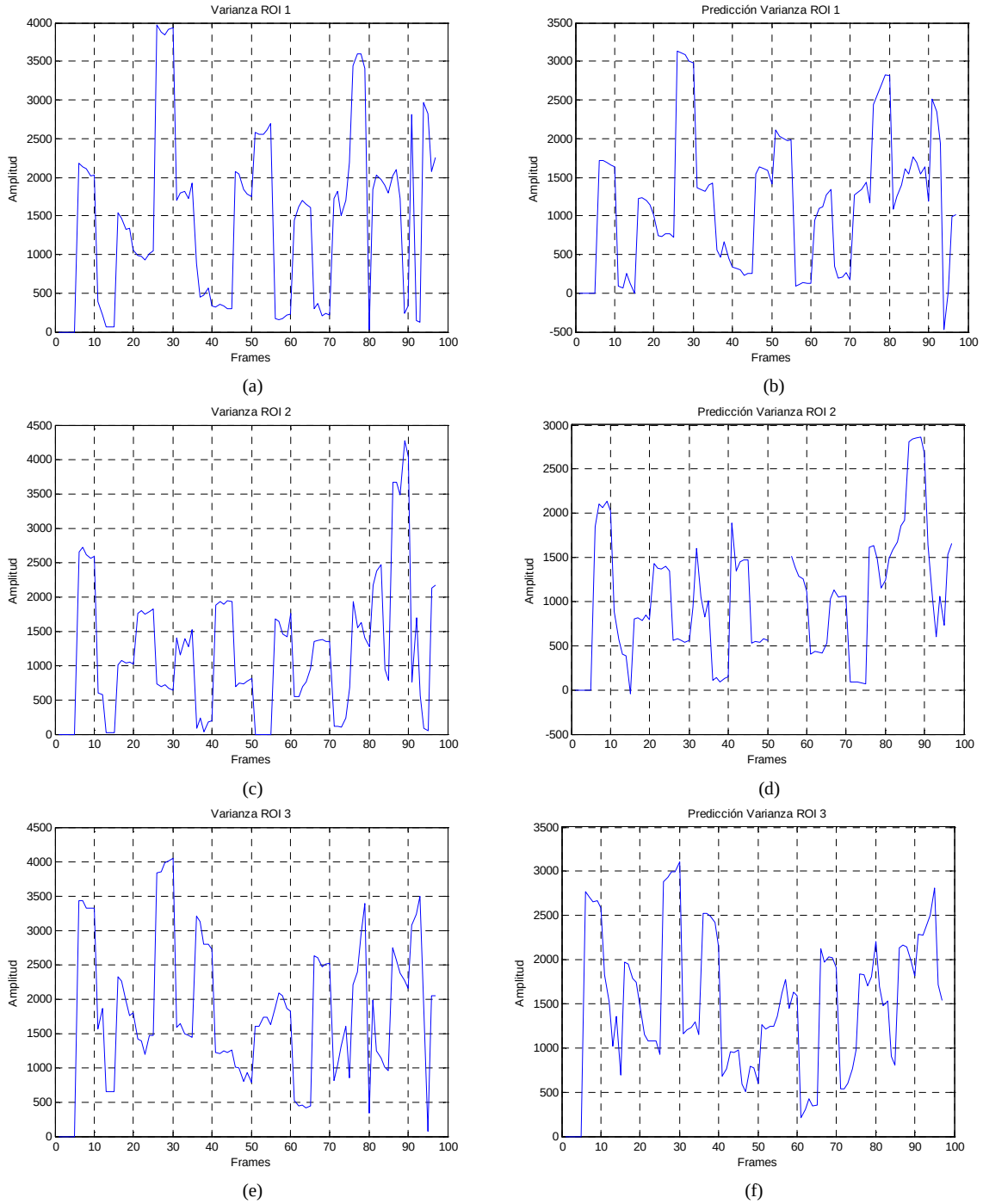


Fig. 19. Resultados correspondientes al atributo Varianza para el video C6.avi: (a) Resultado obtenido del tracking para la ROI 1, (b) Resultado obtenido de la predicción para la ROI 1, (c) Resultado obtenido del tracking para la ROI 2, (d) Resultado obtenido de la predicción para la ROI 2, (e) Resultado obtenido del tracking para la ROI 3, (e) Resultado obtenido de la predicción para la ROI 3.

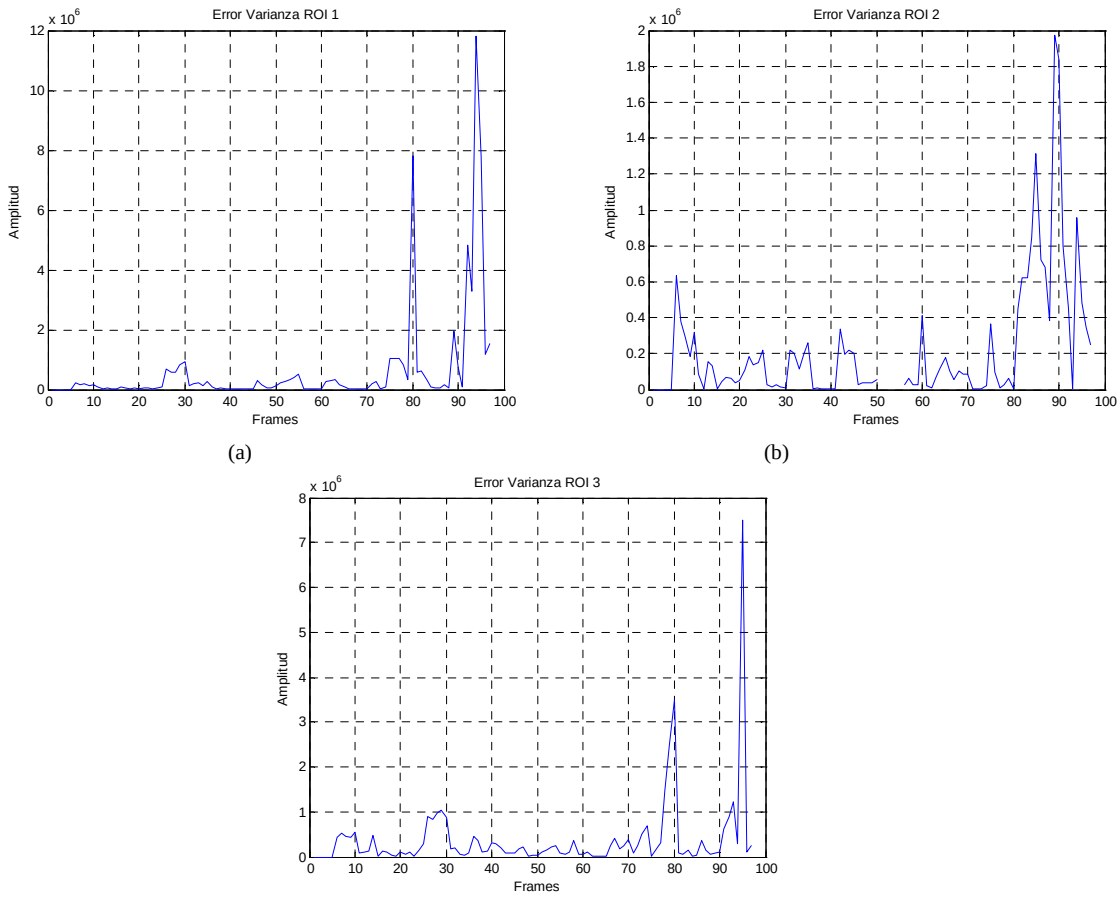
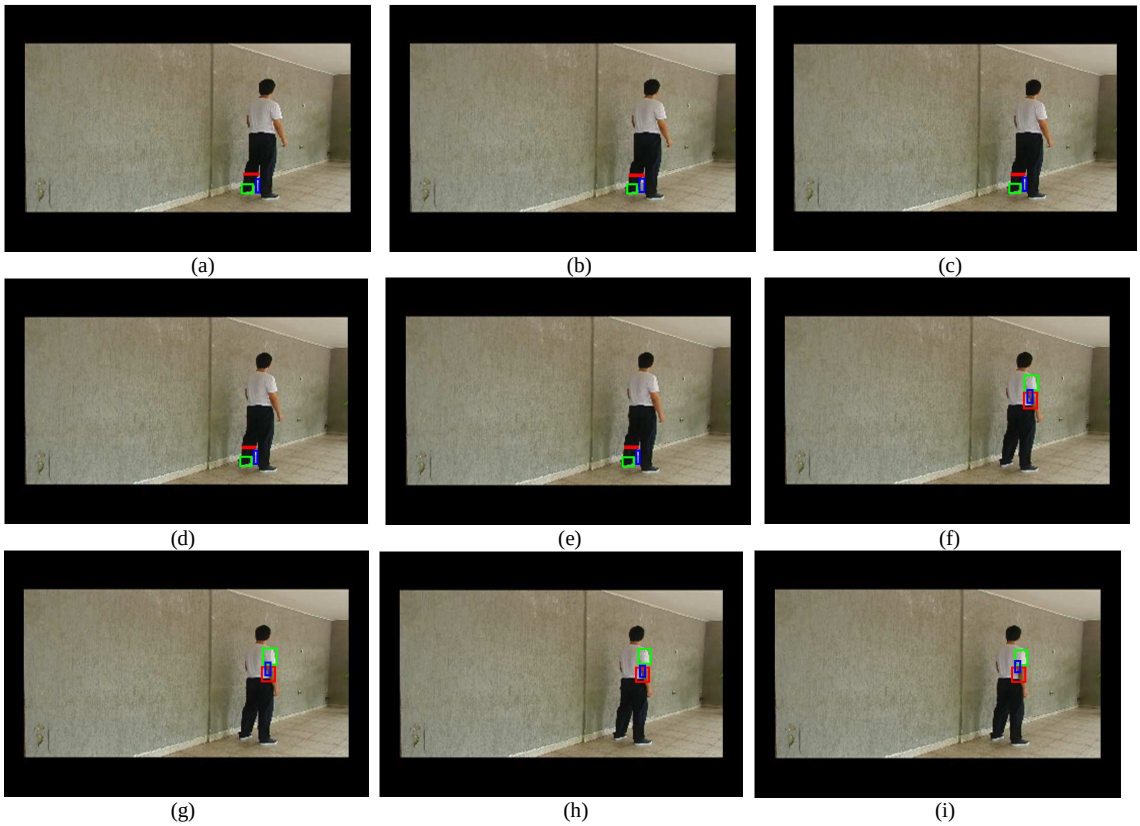
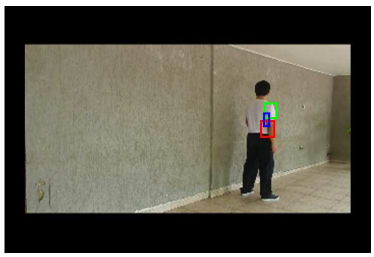


Fig. 20 Señales de error para el atributo Varianza en las diferentes regiones de interés: (a) Señal de error de Varianza para ROI 1, (b) Señal de error de Varianza para ROI 2, (c) Señal de error de Varianza para ROI 3.





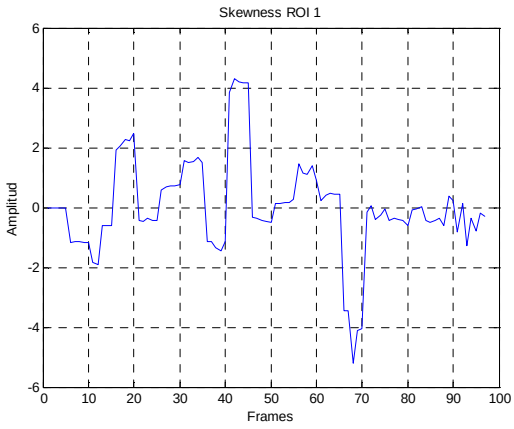
(j)

Fig. 21 *Tracking* realizado a las regiones de interés en una ventana de 10 *frames* para el video C6.avi: (a) Regiones de interés *frame* 41, (b) Regiones de interés *frame* 42, (c) Regiones de interés *frame* 43, (d) Regiones de interés *frame* 44, (e) Regiones de interés *frame* 45, (f) Regiones de interés *frame* 46, (g) Regiones de interés *frame* 47, (h) Regiones de interés *frame* 48, (i) Regiones de interés *frame* 49, (j) Regiones de interés *frame* 50.

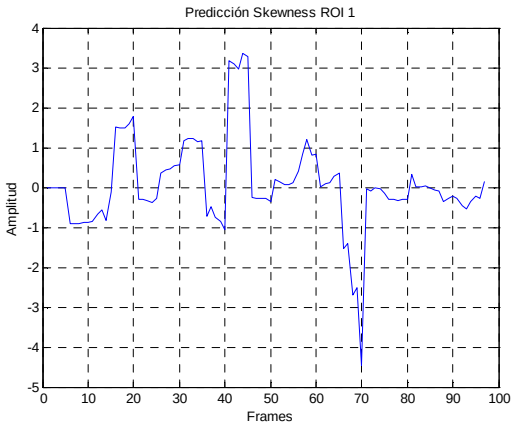
La asimetría o *skewness* permite identificar si los datos se distribuyen de forma uniforme alrededor del punto central (media aritmética). La asimetría presenta tres estados diferentes, cada uno de los cuales define de forma cómo están distribuidos los datos respecto al eje de asimetría, es por esta razón que puede tomar valores tanto positivos como negativos. Al igual que con la varianza, cada uno de los valores del tono de los pixeles en tono de gris se disponen en un vector que finalmente es al que se le calcula la asimetría.

Por cada región de interés se calcula el valor del tono promedio y éste valor será el punto de comparación o eje de asimetría con el que se identificará el tipo de distribución del grupo de datos presente en cada ROI.

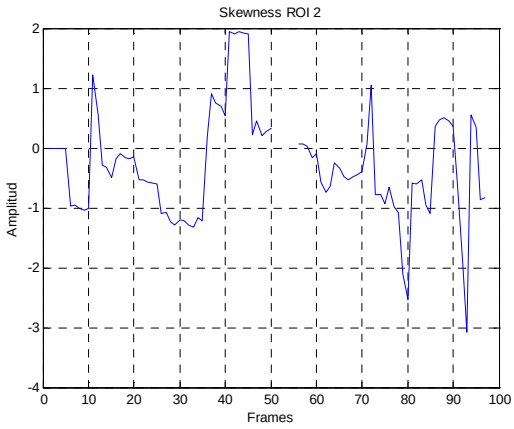
Particularmente se presentaron valores cambiantes, esto se observa en la Figura 22 (a), (c) y (e). No se logra determinar con claridad en este instante en que *frames* pueden existir anomalías. Al igual con lo que hemos detallado anteriormente, la reducida área de las regiones de interés e incluso el ambiente en que se desarrolla el video influye altamente en este resultado.



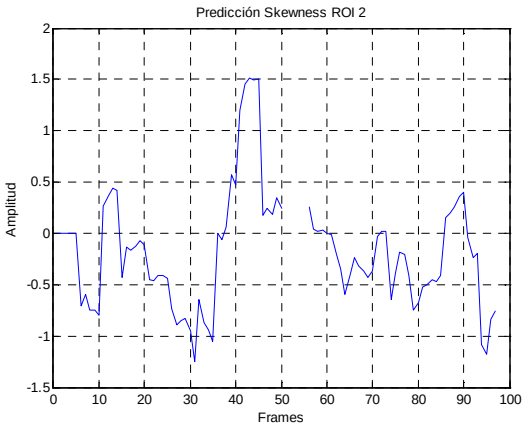
(a)



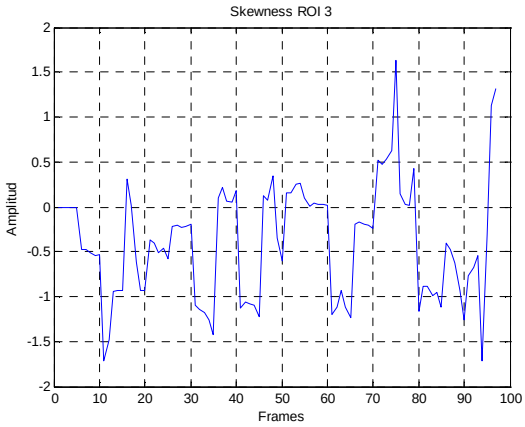
(b)



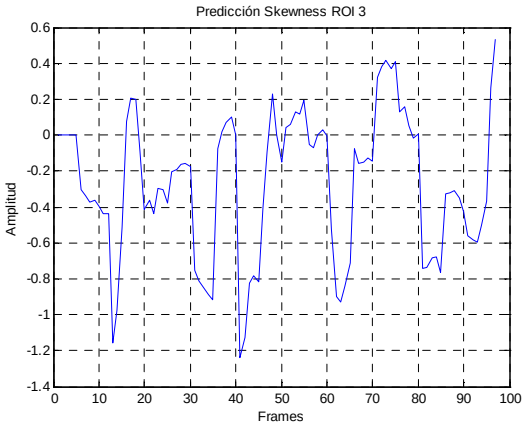
(c)



(d)



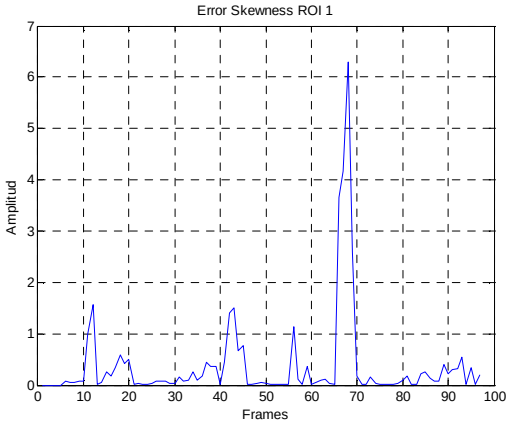
(e)



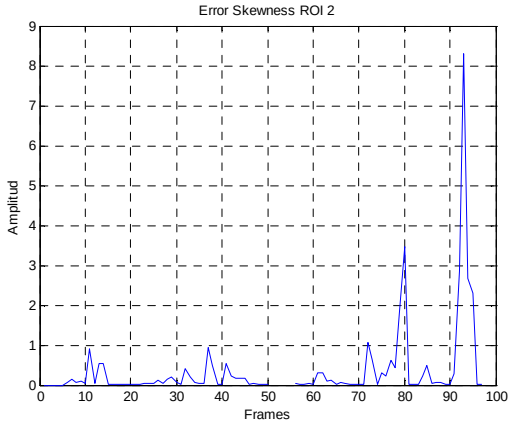
(f)

Fig. 22 Resultados correspondientes al atributo Asimetría o Skewness para el video C6.avi: (a) Resultado obtenido del tracking para la ROI 1, (b) Resultado obtenido de la predicción para la ROI 1, (c) Resultado obtenido del tracking para la ROI 2, (d) Resultado obtenido de la predicción para la ROI 2, (e) Resultado obtenido del tracking para la ROI 3, (e) Resultado obtenido de la predicción para la ROI 3.

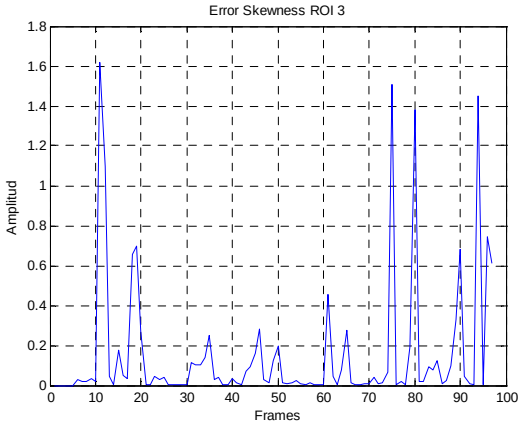
El cálculo del error para el atributo de asimetría, por el contrario, muestra unos resultados mucho mejores. Se consiguió obtener unos gráficos que compensaban las fluctuaciones mostradas en primera instancia en la Figura 22 a pesar que para la ROI 3 Figura 23 (c) no sólo detectó anomalías en los *frames* donde se especificó que existían, sino también en *frames* los cuáles habían sido catalogados como normales en el *ground truth*.



(a)



(b)



(c)

Fig. 23 Señales de error para el atributo Asimetría o Skewness en las diferentes regiones de interés: (a) Señal de error de Asimetría para ROI 1, (b) Señal de error de Asimetría para ROI 2, (c) Señal de error de Asimetría para ROI 3.

Uno de los atributos que mayor información aportó en el análisis de anomalías fue la medida de *kurtosis*. La *kurtosis* determina el grado de concentración que presentan los valores en la región central de la distribución. Por medio del Coeficiente de *Kurtosis*, podemos identificar si existe una concentración de valores elevada, normal o baja.

Para calcular el Coeficiente de *Kurtosis* en cada uno de los *frames* de la secuencia de video, se identificaron las regiones de interés en cada *frame*, dichas regiones, fueron consignadas en una matriz, que posteriormente fue dispuesta en un vector, que contenía todos y cada uno de los valores de los pixeles en tono de gris. A este vector que contenía toda la información de la región de interés, fue al que se le determinó el Coeficiente de *Kurtosis* a través de una función ya existente en MATLAB que lo calculaba. Con este tono promedio se calcula el nivel de concentración de tonos similares al de esta media. Es por esta razón que en la Figura 24 (a) entre los *frames* 40 y 45 se presentan valores elevados, es decir, el valor de los pixeles dentro de la región de interés tomaron un valor similar al promedio en esa ROI en ese instante de tiempo.

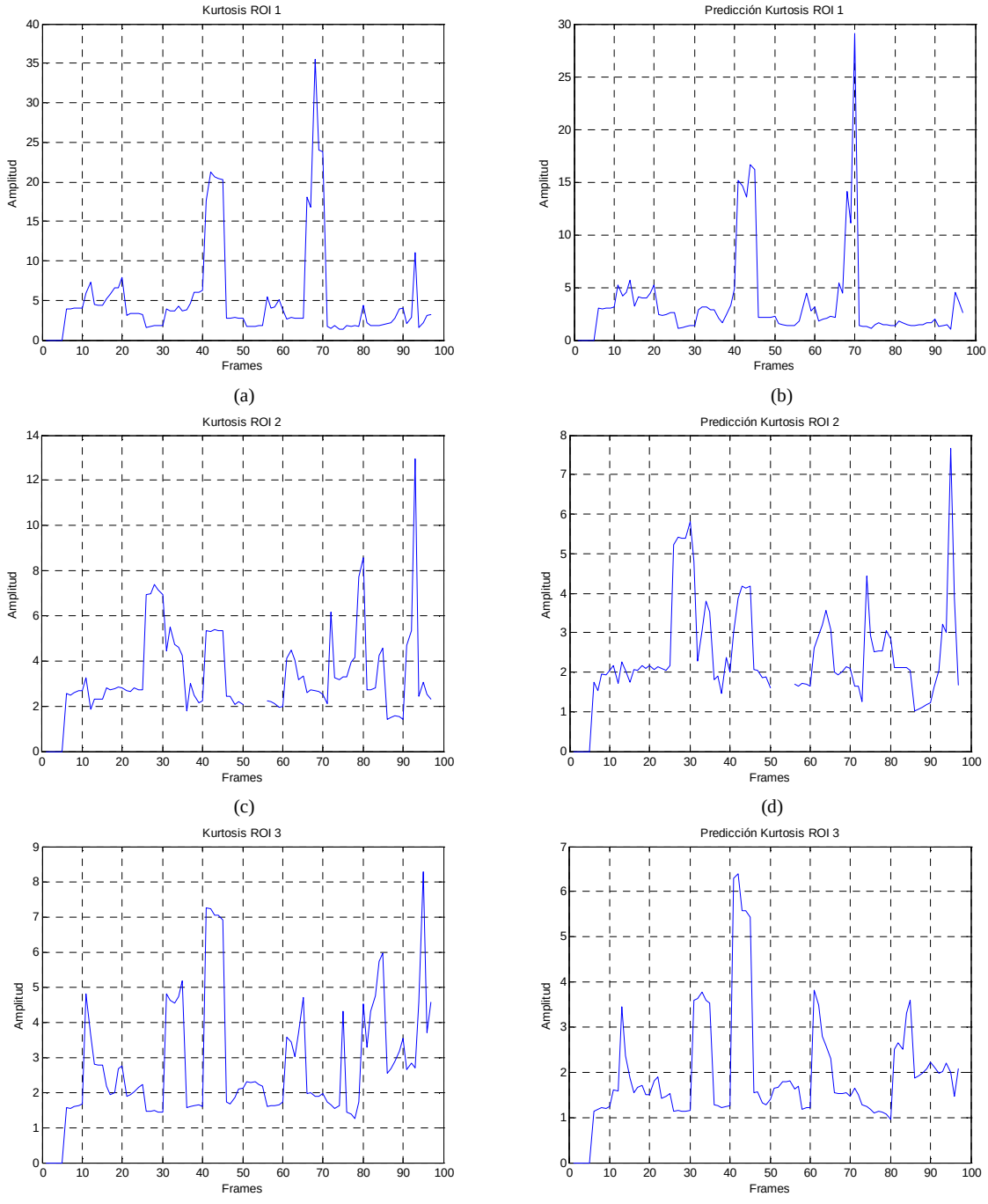


Fig. 24 Resultados correspondientes al atributo *Kurtosis* para el video C6.avi: (a) Resultado obtenido del tracking para la ROI 1, (b) Resultado obtenido de la predicción para la ROI 1, (c) Resultado obtenido del tracking para la ROI 2, (d) Resultado obtenido de la predicción para la ROI 2, (e) Resultado obtenido del tracking para la ROI 3, (e) Resultado obtenido de la predicción para la ROI 3.

Observamos nuevamente que el cálculo del error para el atributo de *kurtosis* Figura 25 (a), (b) y (c) resulta beneficioso en pro de la detección de anomalías. Esto puede verse debido a que los múltiples picos que tomaban los gráficos en la Figura 24 (a), (c) y (e) se vieron suavizados, y las porciones de *frames* donde se supone que existe la posibilidad de que hayan anomalías fueron las que tomaron valores elevados.

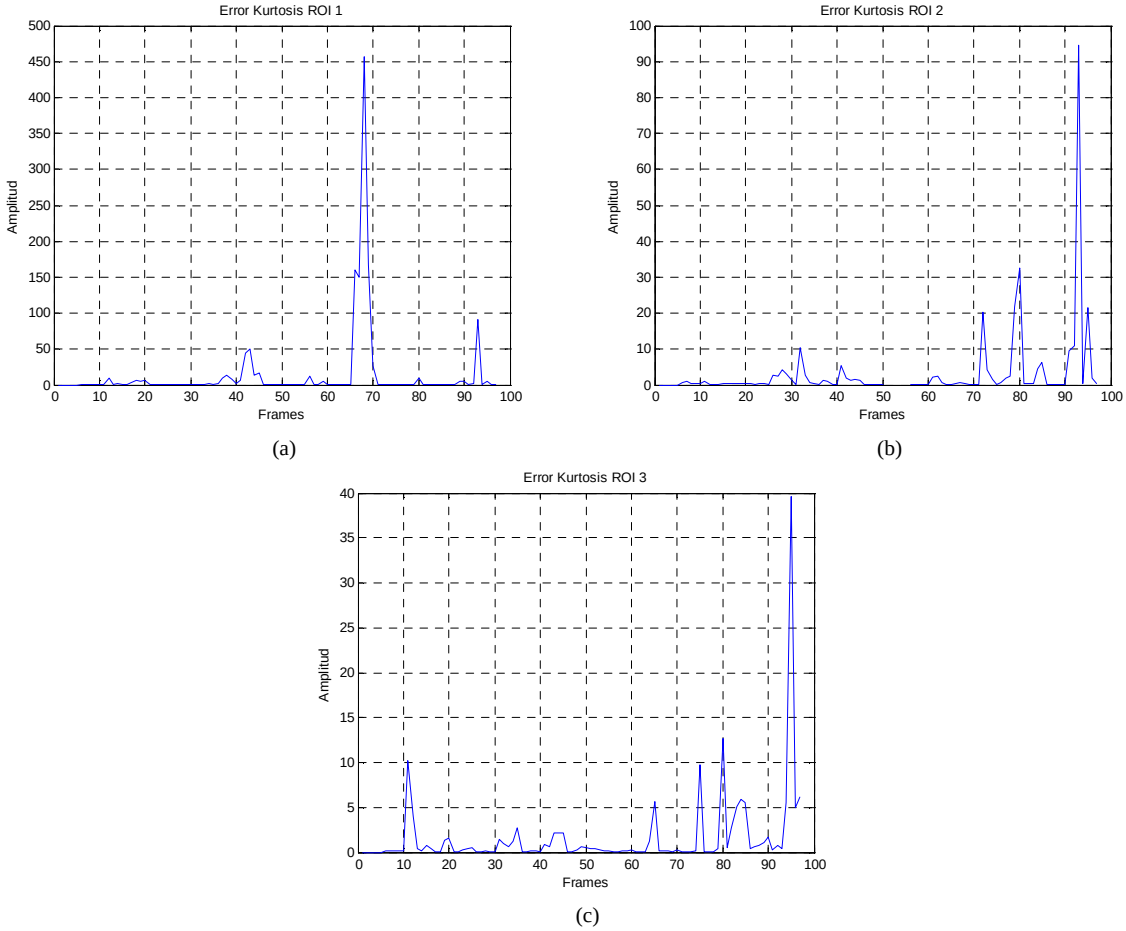


Fig. 25 Señales de error para el atributo *Kurtosis* en las diferentes regiones de interés: (a) Señal de error de *Kurtosis* para ROI 1, (b) Señal de error de *Kurtosis* para ROI 2, (c) Señal de error de *Kurtosis* para ROI 3.

Ahora bien, el procedimiento del cálculo del error ofrece la posibilidad de una detección más selectiva de anomalías, sin embargo, el cálculo de la suma de los resultados de los errores de cada atributo será el paso que finalmente conformará un panorama propicio en el que se pueda discriminar tipos de movimientos anómalos y normales.

El proceso de decisión se lleva a cabo luego de la generación de los resultados de las sumas de cada uno de los atributos teniendo en cuenta los resultados de error, es justo en este punto donde se discrimina entre qué comportamientos son anómalos y cuáles son normales. El proceso de decisión involucra los resultados de las sumas registradas para cada uno de los atributos: desplazamiento horizontal, desplazamiento vertical, Varianza, Asimetría y *Kurtosis*

Como se muestra en la Figura 14, el circuito recibe como entradas las señales de suma de cada uno de los atributos Figura 26 y arroja como única salida la decisión de si el *frame* analizado presenta o no anomalía.

En la Figura 26 (a), (b), (c), (d) y (e) se muestras los gráficos de las sumas de los errores de los atributos. La información contenida dentro de los gráficos mencionados se irá generando *online* conforme el video es analizado por el algoritmo, por lo que la decisión se tomará en tiempo real. La decisión esta ligada a la implementación de un umbral adaptativo, cambiante en el tiempo, y un factor de escala. El uso del umbral adaptativo ofrece robustez ante parámetros como escenarios en los que se desarrolla la secuencia de video, iluminación, sombras y movimientos de los sujetos. Para cada

atributo de cada *frame* en análisis se revisa si la información resultante supera o no el umbral, esto es, si el dato supera el umbral significa que presenta anomalía y por lo tanto, el atributo en dicho *frame*. El factor de escala se utiliza para ajustar el umbral, su valor por defecto es la unidad, y al aumentarlo el algoritmo se vuelve más selectivo.

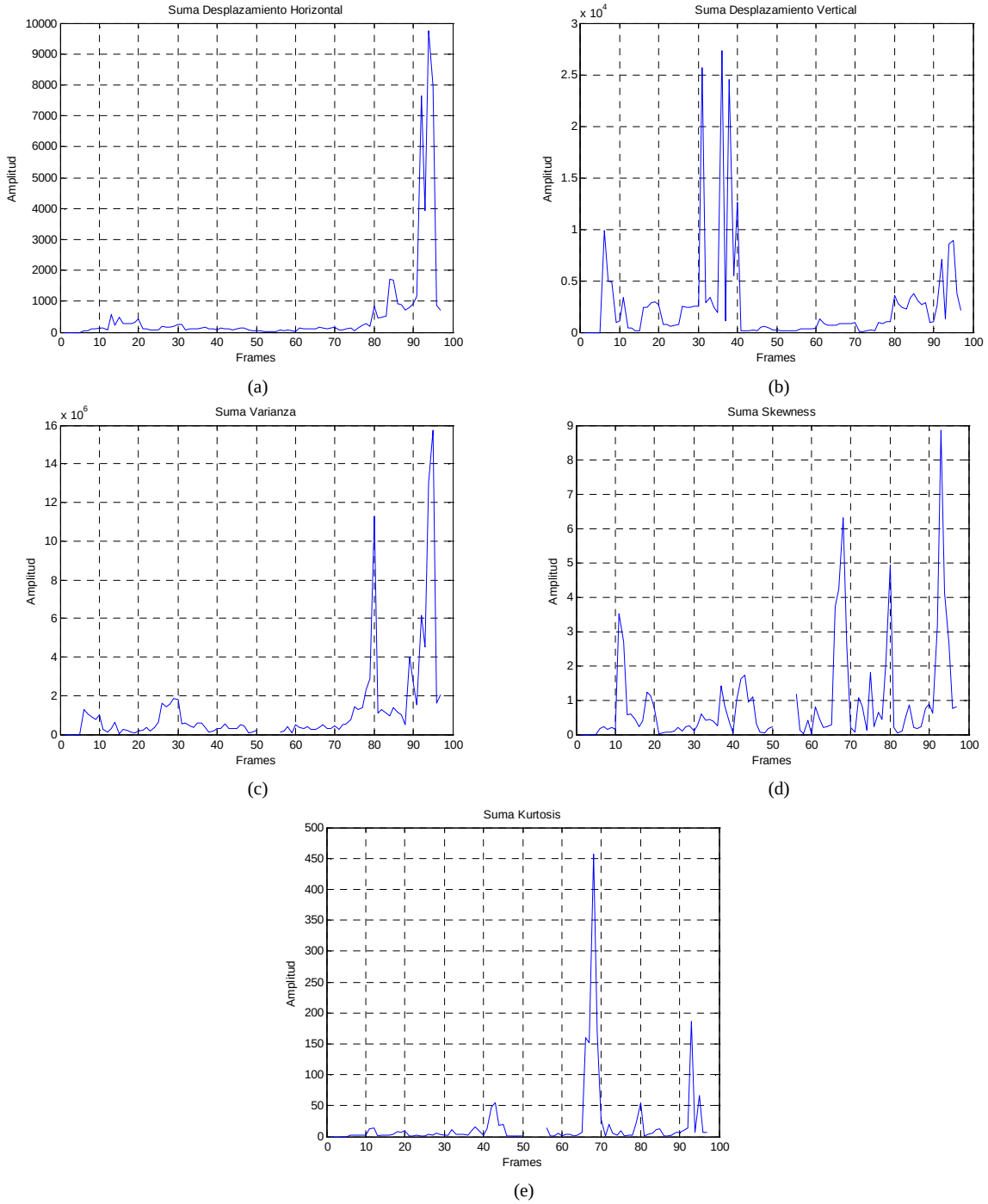


Fig. 26 Señales de suma de error para cada uno de los atributos: (a) Señal de suma de Desplazamiento horizontal, (b) Señal de suma de Desplazamiento vertical, (c) Señal de suma de Varianza, (d) Señal de suma de Asimetría, (e) Señal de suma de *Kurtosis*.

El algoritmo fue puesto a prueba en 33 videos con diferentes escenarios y ambientes todo esto para determinar el desempeño general del mismo. Los resultados obtenidos para el video C6.avi fueron calculados del mismo modo para todos los videos.

En el ANEXO D fueron consignados los resultados encontrados luego de poner a prueba el algoritmo de tal forma que se pudiese determinar cuales eran los videos que tenían mejor desempeño. Se dispuso una tabla que identificaba en una de sus columnas todos los *frames* para los videos, y en las columnas

subsiguientes se consignaron los resultados de la determinación de anomalía para cada video en cada *frame*.

Con el fin de generar un resultado numérico que fuera medible se llevo a cabo un análisis de sensibilidad y especificidad del algoritmo. Debido a la susceptibilidad de incurrir en una errada determinación de anomalías y por ser un clasificador de tipo binario, se analizó cuáles *frames* habían sido catalogados por el programa como anómalos y no lo eran (Falsos Positivos - FP) y cuáles *frames* habían sido catalogados como anómalos y efectivamente poseían anomalía (Verdaderos Positivos - VP). Lo anterior condujo a representar gráficamente la sensibilidad del algoritmo frente a 1 – especificidad, que en otras palabras, es la probabilidad de aciertos -detectar anomalía cuando efectivamente la hay-, contra la de desaciertos -detectar anomalía cuando no la hay-, en la determinación de las anomalías.

En cada *frame* para cada video se definió la existencia de Falso Positivo (FP) o Verdadero Positivo (VP) y fueron totalizados. La probabilidad de aciertos (Razón de Verdaderos Positivos VPR) se calculó según la razón del total de VP y el total de *frames* catalogados con anomalías, conforme el *ground truth*. Así mismo, la probabilidad de desaciertos (Razón de Falsos Positivos FPR) se calculó teniendo en cuenta la razón del total de FP y el total de *frames* etiquetados como normales.

Los resultados de la probabilidad de aciertos y desaciertos para cada video se dispusieron en la Figura 27, cada punto corresponde a un video en particular, los videos que tuvieron mejor desempeño son aquellos que tienen valores de sensibilidad elevados y valores de 1 – especificidad bajos, en el caso ideal todos los puntos deberían estar concentrados en la coordenada (0,1), lo que implica una probabilidad de aciertos del 100% sin cometer desacierto alguno. El video que tuvo mejor desempeño fue el video C6.avi con 34.3% de aciertos contra 8.3% de desaciertos, seguido por el video C19.avi con 29.5% de aciertos y 5.7% de desaciertos. El resultado de todos los videos puede ser consultado en el ANEXO D.

Aparentemente la cantidad de aciertos del algoritmo es bajo, pero teniendo en cuenta que el análisis se lleva a cabo *frame* a *frame* los resultados obtenidos cumplen las especificaciones planteadas.

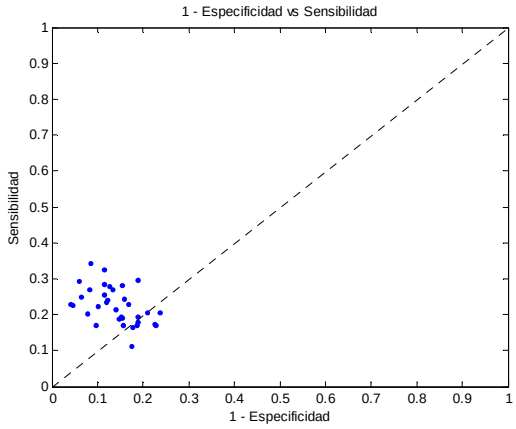


Fig. 27 Curva ROC Nombre video: C6.avi: Tasa de *frames*: 30 *frames*/s, Longitud de ventana: 10 *frames*, número de ROIs: 3, orden del predictor: 2, factor de umbral: 1,0.

5.2 Desempeño de Atributos

Con el fin de establecer cuál atributo fue el que tuvo mejor desempeño en la determinación de anomalías, se tomarán 4 videos en los que se observaron los mejores resultados y cuantitativamente se analizará el resultado en cada *frame* para cada atributo verificando la concordancia con la descripción de los sucesos en cada uno de ellos en la secuencia del video con ayuda del *ground truth*.

En las tablas que se muestran en el ANEXO B (Tabla 1, Tabla 2, Tabla 3, Tabla 4) se observan los resultados para los videos C6.avi, C19.avi, D3.avi. y B8.avi. En la columna *Frame* se confinan los cuadros del video donde el algoritmo detectó la presencia de anomalía. Las columnas Dv, Dh, V, K y

S hacen referencia a los atributos de desplazamiento vertical, desplazamiento horizontal, varianza, *kurtosis* y asimetría respectivamente. Como se mencionó con anterioridad, como mínimo deben existir 3 atributos que cataloguen a cada *frame* como anómalo para que el algoritmo marque en el *frame* correspondiente una anomalía. La marca (**O**) hace referencia a que el atributo marcó anomalía en un *frame* donde, sustentado en el *ground truth*, no existe anomalía (Falso Positivo). Por el contrario, la marca (**X**) significa que el atributo señaló anomalía en un *frame* donde sí existía (Verdadero Positivo).

Para cuantificar los resultados, se totalizaron los resultados de falsos positivos y verdaderos positivos para cada atributo, se calculó el total de *frames* anómalos detectados y se comprobó cuál era el atributo que tuvo mayor incidencia en la decisión de anomalía.

Como se ve en el ANEXO B, la Tabla 1, para el video C6.avi el atributo que tuvo mejor desempeño fue el de desplazamiento horizontal, resultado lógico en vista a los sucesos que acontecen en la secuencia de video, ya que el movimiento errático del sujeto facilita que éste atributo identifique con un mayor grado de acierto las anomalías.

En el video C19.avi, ANEXO B Tabla 2, destacaron fundamentalmente los atributos de desplazamiento horizontal, desplazamiento horizontal, *kurtosis* y asimetría. La secuencia de video contemplaba dos sujetos incidiendo en la escena y en determinado instante se ensañaban en empujones y puñetazos erráticos, estos movimientos erráticos permitieron que el algoritmo detectara en la mayoría de sus atributos anomalías, debido a que se presentaban movimientos en todos los sentidos posibilitando variaciones de los tonos de los pixeles bruscas que permitieron que casi todos los atributos contribuyeran en buena medida al resultado.

En el video D3.avi, ANEXO B Tabla 3, se destacaron 2 atributos, el de desplazamiento horizontal y el de varianza. Esto es porque en la secuencia de video inciden dos sujetos que al llegar al encuentro se ensañan en golpe y empujones. La manera en que se empujaron marcó valores altos para el desplazamiento horizontal, ya que de estar en una posición casi fija los sujetos pasaban a ubicarse en una posición totalmente diferente en un lapso de tiempo corto. Estos movimientos brindaron un buen escenario para se presentaran dispersiones de datos elevadas, implicando que los valores de los pixeles tuvieron una gran variabilidad y, por consiguiente, que el atributo de varianza fuera determinante en la detección de anomalías.

Definitivamente para el video B8.avi, ANEXO B Tabla 4, los atributos de desplazamiento vertical y asimetría fueron los que tuvieron mayor incidencia en la determinación de anomalías. El desplazamiento vertical fue trascendental en la decisión principalmente porque en el desarrollo del video se presenta un sujeto desplazándose en la escena que ejecuta un salto; este salto permite que el atributo de desplazamiento vertical tome valores elevados cuando el sujeto realiza el movimiento. La asimetría es el otro atributo que incidió en gran medida en las decisiones de anomalías en lo *frames*, se debe a que por ser un video que se desarrolla en exteriores o *outdoors* el valor de la asimetría tuvo variaciones grandes por razones de iluminación y escenario donde se despliegan los sucesos.

En conclusión, el desempeño de los atributos depende del contexto en el que se desarrolle el video, es decir, para los videos en que se presentaban saltos, cuclillas, desmayos, y en general, movimientos que implicaran un desplazamiento vertical, el atributo denominado de la misma forma influía en mayor grado que los demás, caso contrario a cuando en el video se presentaban acciones que involucraban desplazamientos horizontales como huidas o apariciones repentinas de sujetos, en estos casos el atributo de desplazamiento horizontal destacaba. Adicionalmente, los momentos estadísticos mostraron un buen desempeño en general, promediando valores elevados, lo que implicó que fueran atributos decisivos al momento de la determinación de las anomalías en los comportamientos.

5.3 Análisis por Grupos de *frames* y Desempeño del Algoritmo en Interiores y Exteriores

Con el fin de poder obtener mejores resultados fue pensado el análisis por grupos de *frames*, este procedimiento implica marcar *frames* como anómalos a aquellos que el algoritmo no los identificó con anomalía. Las agrupaciones se realizaron en grupos de 3, 5 y 7 *frames*. El procedimiento de marcación de *frames* se hace como sigue:

Para realizar el análisis en grupos de 3 *frames* se tuvo en cuenta aquellos *frames* los cuáles el algoritmo identificó con anomalía. Al momento de encontrar *frames* con anomalía se marcaban con anomalía el *frame* anterior y el *frame* siguiente independientemente de la existencia de anomalía en estos *frames*. Para el análisis en grupos de 5 y 7 *frames* se marcaron los dos *frames* anteriores y los dos *frames* siguientes y, los 3 *frames* anteriores y 3 *frames* siguientes respectivamente. Para visualizar lo anteriormente dicho se supondrá el resultado para un video que cuenta con 20 *frames* luego de analizarlo con el algoritmo propuesto Figura 28. Este análisis puede ser consultado en el ANEXO D

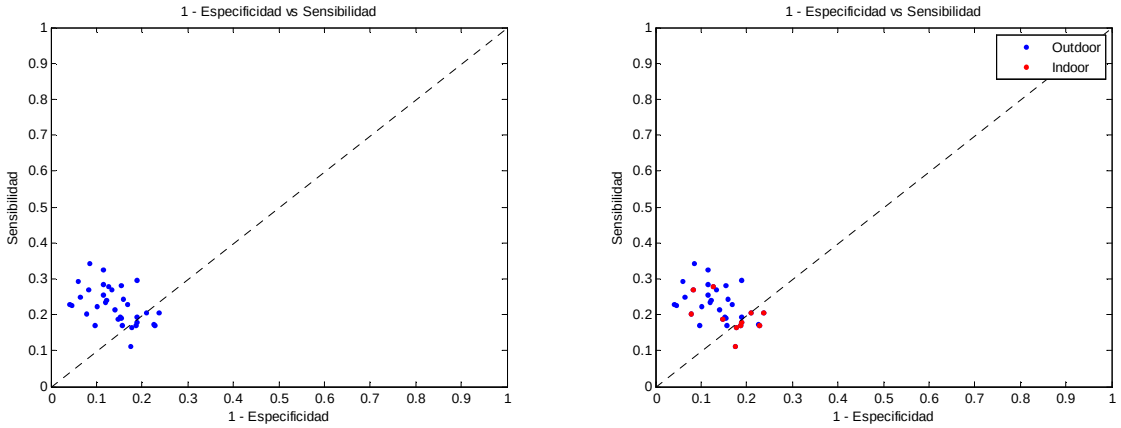
Frames	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Resultado Original	N	N	O	O	N	N	N	N	N	N	N	N	N	N	N	O	N	N	N	N
Análisis en Grupos de 3	N	A	O	O	A	N	N	N	N	N	N	N	N	N	A	O	A	N	N	N
Análisis en Grupos de 5	A	A	O	O	A	A	N	N	N	N	N	N	N	A	A	O	A	A	N	N
Análisis en Grupos de 7	A	A	O	O	A	A	A	N	N	N	N	N	A	A	A	O	A	A	A	N

Fig. 28 Ejemplo de Análisis en grupos de 3, 5 y 7 *frames*: O: *frame* originalmente identificado con anomalía, N: *frame* originalmente identificado como normal y A: *frame* catalogado con anomalía luego de realizar el análisis por grupos.

El análisis por grupos de *frames* fue pensado en base a 2 hipótesis fundamentales: 1) El algoritmo incurre en error marcando anomalías en *frames* consecutivos que no la poseían debido a errores mencionados con anterioridad, lo que el aumento de 1-especificidad sería reducido comparado al de la sensibilidad. 2) Se presenta un número elevado de *frames* marcados con anomalía correctamente (verdadero positivo) que están distribuidos a lo largo de la secuencia de video, lo que presuntamente no ocurriría con los *frames* falsos positivos, por lo que el análisis de grupos permitiría aumentar la sensibilidad del algoritmo.

En base a la evidencia teórica planteada, se muestran los resultados luego de realizar el análisis de grupos. Adicional a esto, se varió el factor de escala para ajustar el valor del umbral con el fin de confirmar la selectividad del algoritmo y verificar si se presenta un mejor desempeño del mismo, asimismo, se etiquetaron los puntos de los videos cuyos escenarios fueron exteriores (*outdoors*) con color azul e interiores (*indoors*) con color rojo.

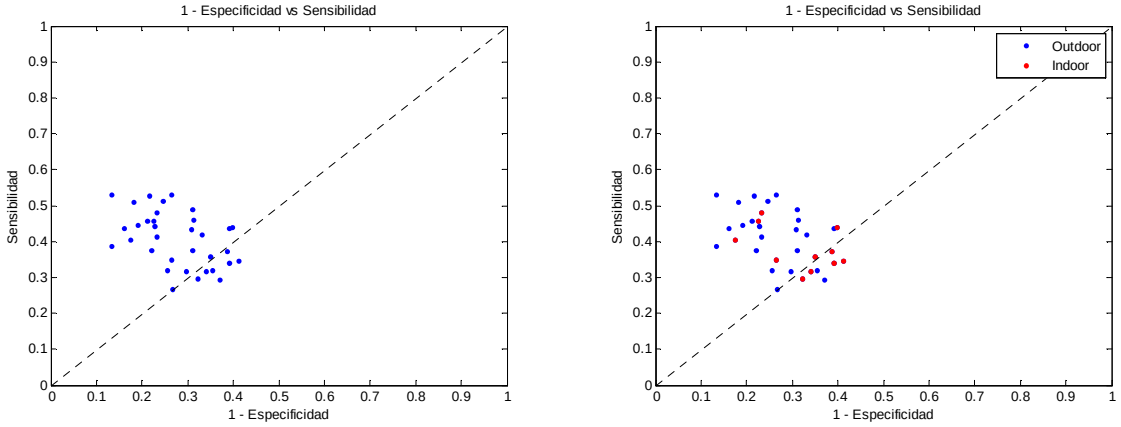
Podemos ver en la Figura 29 (a), que el resultado entregado por el programa es muy bueno, se consiguen valores de sensibilidad elevados comparados con su contraparte. Al momento de hacer la distinción entre los videos llevados a cabo en exteriores e interiores Figura 29 (b), se pudo constatar que el programa tuvo mejor desempeño con los videos hechos en exteriores (*outdoors*). De los 11 videos hechos en interiores, sólo 3 destacaron con un buen resultado, caso contrario se observó con los videos hechos en exteriores, donde la gran mayoría estuvo por encima de la diagonal mostrada, lo que implicaba que el programa estaba haciendo una detección correcta de anomalías. Este resultado se puede deber a diversos factores, la iluminación en exteriores es mayor, esto implica la inexistencia de sombras que perturben la buena segmentación de fondo; se pueden conseguir planos más amplios, captando todos los sucesos con mayor exactitud a pesar de que el área efectiva se reduzca; en interiores generalmente el fondo es complejo, sillas, mesas, espejos, televisores, objetos que generan reflejos que estimulan al programa a cometer error por inducirlo a generar falsas regiones de interés.



(a) (b)

Fig. 29 (a) Resultado entregado por el programa sin hacer análisis por grupo de *frames* con factor de escala del umbral de 1. (b) Distinción entre los videos realizados en exteriores e interiores.

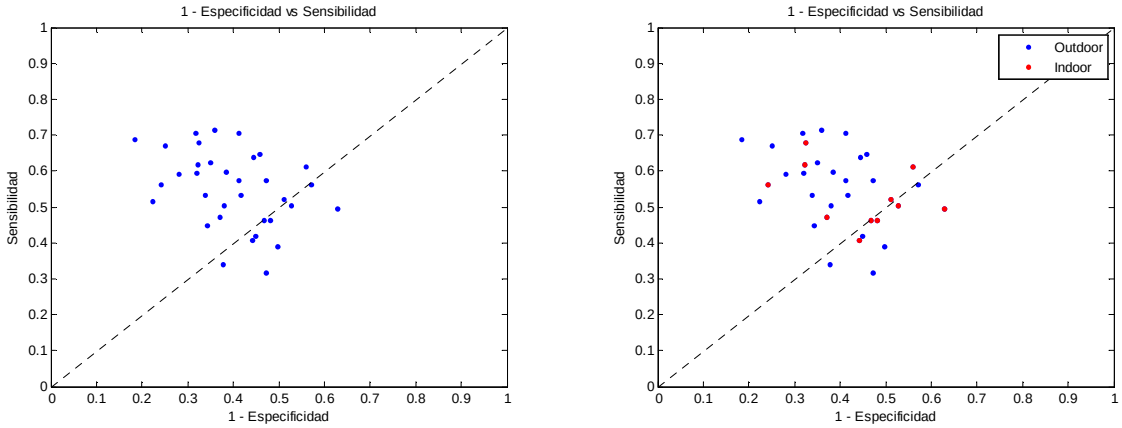
Al realizar el análisis en grupos de 3 *frames* se observa, como es lógico, que la sensibilidad y 1-especificidad se eleven. Al etiquetar *frames* que antes no poseían anomalías provocó que el cúmulo de datos se viera forzado a ubicarse en inmediaciones del gráfico. Al comprobar si la razón entre la detección correcta o no de anomalías mejoraba, se logró concluir que cambió poco o nada el resultado, incluso, se degradó, ya que al no realizar el análisis de grupos se tenía una razón promedio de 1,639, en cambio al llevar el proceso de análisis en grupos de 3 *frames*, la razón disminuyó a 1,475, lo que para este caso no fue conveniente realizar el análisis. Se observa en la Figura 30 (b), que los videos en exteriores siguen teniendo un mejor desempeño, el análisis en grupos no cambió en nada los resultados vistos anteriormente.



(a) (b)

Fig. 30 (a) Resultado entregado por el programa haciendo análisis por grupo de 5 *frames* con factor de escala del umbral de 1. (b) Distinción entre los videos realizados en exteriores e interiores.

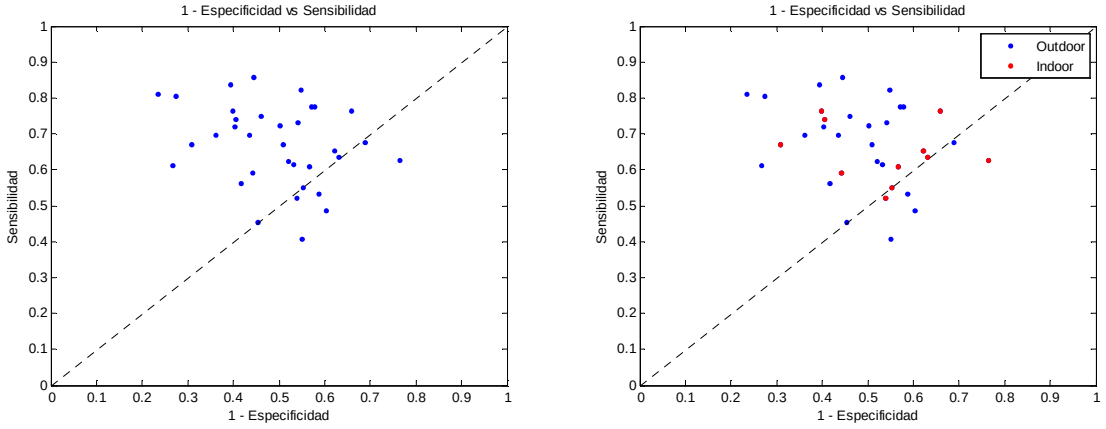
Como sucede en el caso de agrupación de 3 *frames*, al realizar grupos de 5 *frames*, Figura 31 (a) la razón promedio entre la sensibilidad y 1-especificidad se vio disminuida a 1,378. La dispersión de los resultados no se afectó, además de esto, la constate de que lo videos de exteriores presentan mejor desempeño persiste, Figura 31 (b).



(a) (b)

Fig. 31 (a) Resultado entregado por el programa haciendo análisis por grupo de 5 *frames* con factor de escala del umbral de 1. (b) Distinción entre los videos realizados en exteriores e interiores en el videoC6.avi.

La Figura 32 (a), es resultado de llevar a cabo el análisis en grupos de 7 *frames*. La dispersión de los datos es muy alta puesto que la mayoría de ellos se catalogaron con anomalía, adicionalmente, muchos *frames* fueron etiquetados con anomalía incorrectamente, es por eso que los valores de 1-especificidad se ubican acentuadamente a la derecha del gráfico, por tal razón el resultado no es bueno. La razón promedio entre la sensibilidad y 1-especificidad se reduce aún más luego del análisis en grupos de 7 *frames* a 1,363.



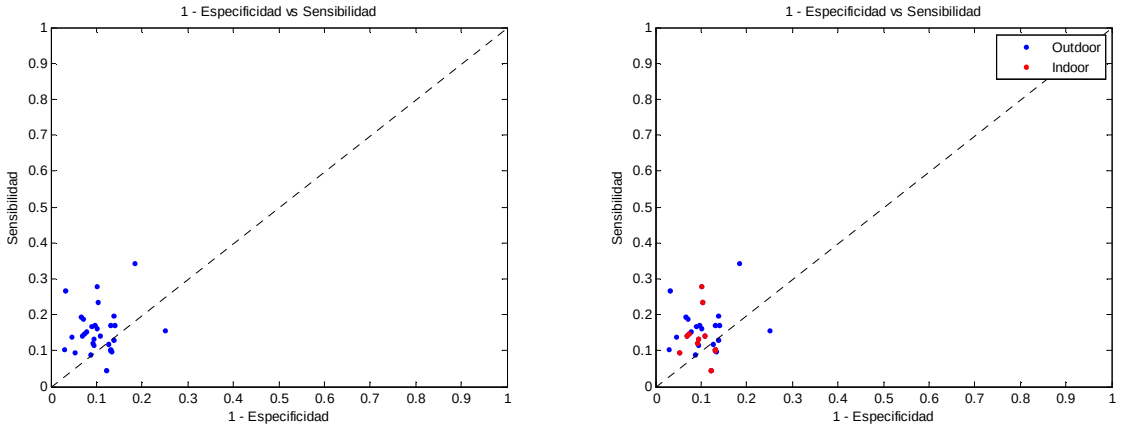
(a) (b)

Fig. 32 (a) Resultado entregado por el programa haciendo análisis por grupo de 7 *frames* con factor de escala del umbral de 1. (b) Distinción entre los videos realizados en exteriores e interiores.

Para tener en cuenta todos los factores que pudiesen influir en los resultados que mostrara el algoritmo, se concibió modificar el factor de escala del umbral para que éste fuera más selectivo y discriminara de mejor manera las anomalías. Con esto se busca disminuir la detección incorrecta de anomalías, sustentado en el argumento que la detección errada se debe a la participación de atributos donde el valor del umbral no fue suficiente para descartarlos.

Se espera reducir el número de detecciones incorrectas en mayor medida a costa de perder información pertinente. Se empezó elevando el factor de escala del umbral a 1.2, el resultado de cambiar el umbral con este factor se observa en la Figura 33 (a).

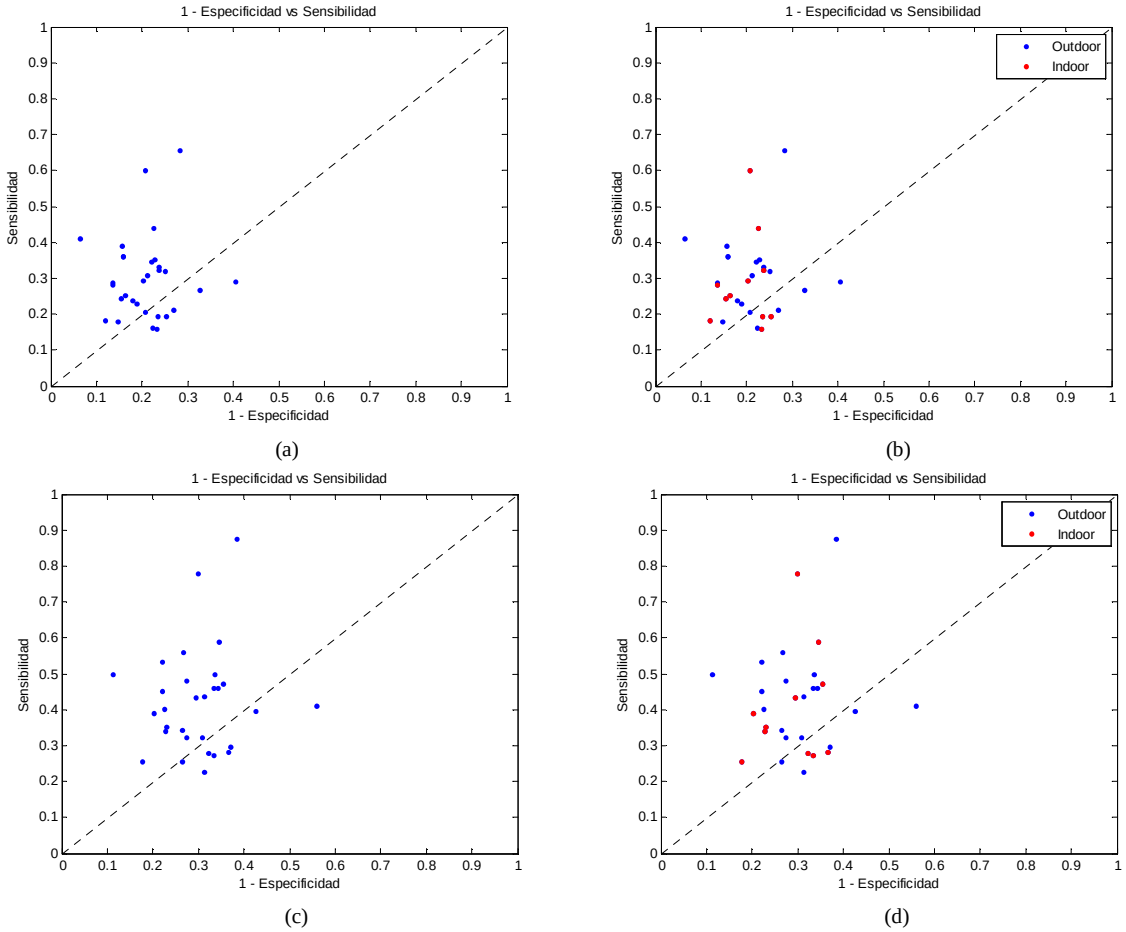
Al aplicar un umbral más elevado, se logra independizar el algoritmo de los escenarios en los que se desarrollen las secuencias de video. Es como en la Figura 33 (b) se puede observar que los videos en interiores, que antes no sobrepasaban la diagonal o estaban sobre la misma, implicando que el algoritmo casi aleatoriamente determinara anomalías, ahora con el nuevo valor de umbral se desempeñaron favorablemente.



(a) (b)

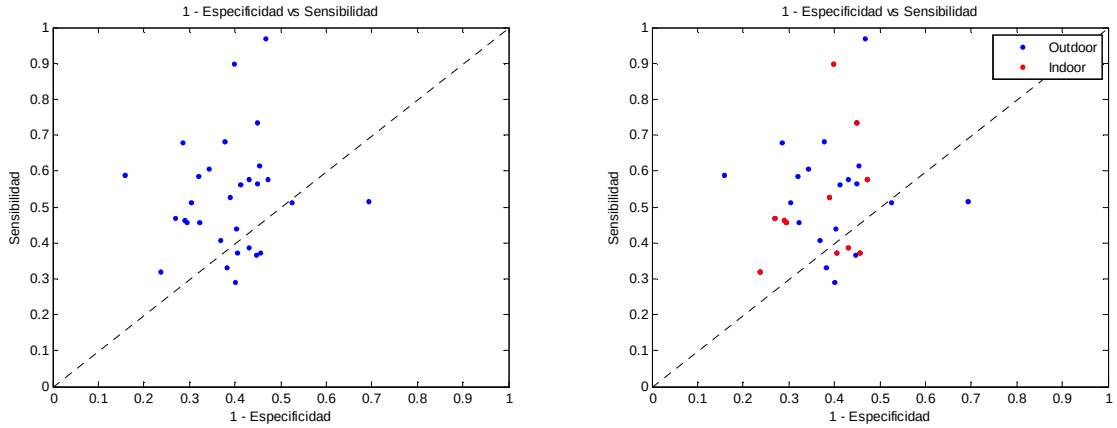
Fig. 33 (a) Resultado original entregado por el programa con un factor de escala de umbral de 1.2. (b) Distinción entre los videos realizados en exteriores e interiores.

Se realizó igualmente el análisis por grupos como se describió anteriormente para el factor de escala de umbral de 1 pero ahora con 1.2. Los resultados obtenidos luego de este procedimiento son los mostrados en las Figura 34.



(a) (b)

(c) (d)



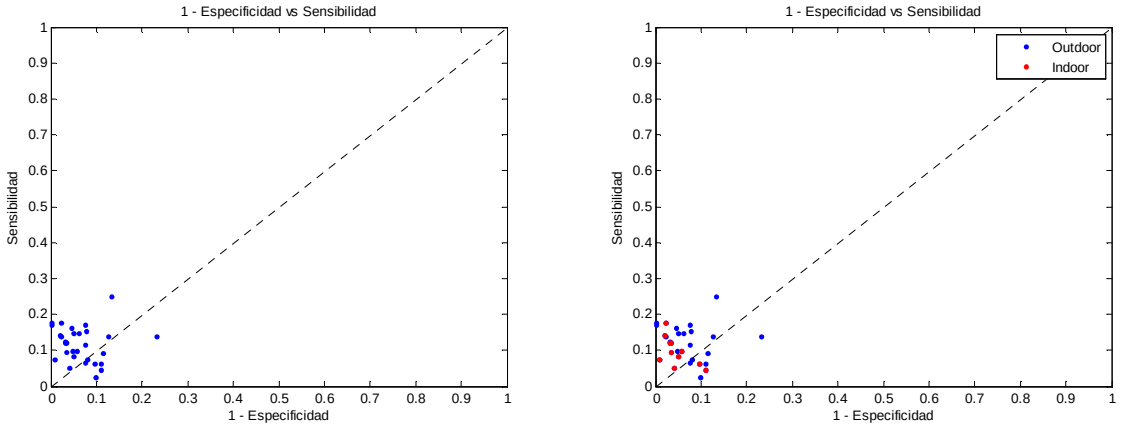
(f) (g)

Fig. 34 (a) Resultado entregado por el programa haciendo análisis por grupos de 3 *frames* con un factor de escala de umbral de 1.2. (b) Distinción entre los videos realizados en exteriores e interiores para el análisis de grupos de 3 *frames* y factor de escala de 1.2. (c) Resultado entregado por el programa haciendo análisis por grupos de 5 *frames* con un factor de escala de umbral de 1.2. (d) Distinción entre los videos realizados en exteriores e interiores para el análisis de grupos de 5 *frames* y factor de escala de 1.2. (f) Resultado entregado por el programa haciendo análisis por grupos de 7 *frames* con un factor de escala de umbral de 1.2. (g) Distinción entre los videos realizados en exteriores e interiores de 7 *frames* y factor de escala de 1.2.

Se determinó que al ir aumentando la agrupación de *frames*, la dispersión de puntos era proporcional lo que de ninguna forma fue beneficioso ya que idealmente se requieren de datos agrupados para garantizar que el algoritmo actúe como un clasificador. El programa se desempeño independientemente del tipo de video, ya sea *outdoor* o *indoor*.

La razón promedio entre la sensibilidad y 1-especificidad se reduce a medida que se aumenta la cantidad de *frames* agrupados, esto fue, 1,516 para el resultado original sin aplicar el análisis de grupos, 1,466484932 al realizar el análisis en grupos de 3 *frames*, y 1,419 y 1,366 al concebir el análisis en grupos de 5 y 7 *frames*.

Nuevamente se incrementó el factor de escala del umbral a 1.4. En este caso se observó un resultado destacable respecto a los resultados obtenidos con otros umbrales. La razón promedio obtenida sin realizar el análisis en grupos de *frames* fue la más alta registrada Figura 35 (a) con un 1,806. Esto indica que la cantidad de aciertos comparada con la de desaciertos fue la más alta, por lo que se determina que este factor de escala del umbral es el óptimo debido a que es capaz de eliminar *frames* falsos positivos y hacer prevalecer los verdaderos positivos. Al aumentar el factor de escala del umbral es evidente que los parámetros de sensibilidad y 1- especificidad se reduzcan viéndose esta última con un mayor impacto. Se logró acentuar de forma más clara el clasificador que se pretendía construir. Es destacable que se consigue por primera vez obtener una discriminación sin falsos positivos, es decir, el algoritmo no incurrió en error al detectar anomalías que no existían, además de esto la naturaleza del video (*indoor*, *outdoor*) no fue determinante al momento de arrojar el resultado del algoritmo Figura 35 (b).



(a) (b)

Fig. 35 (a) Resultado original entregado por el programa con un factor de escala de umbral de 1.4. (b) Distinción entre los videos realizados en exteriores e interiores.

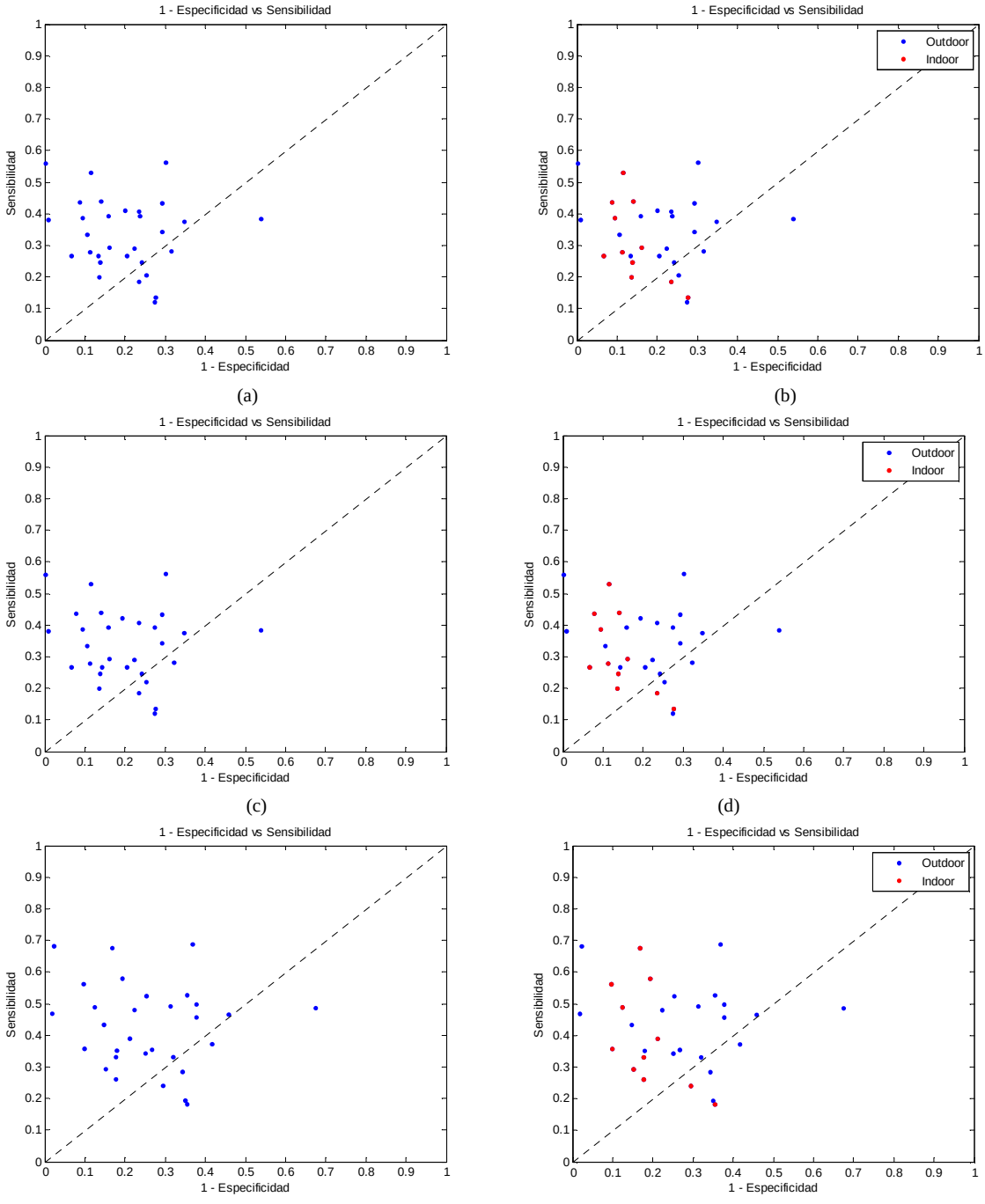
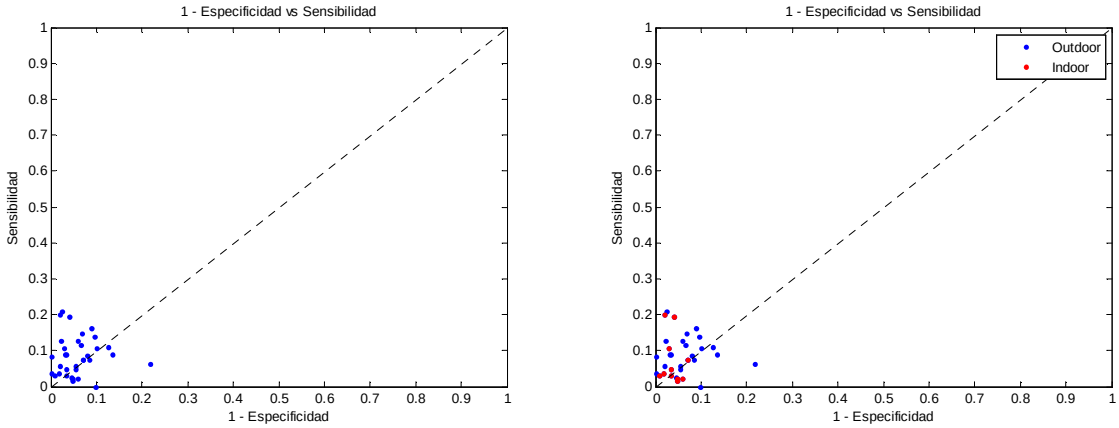


Fig. 36 (a) Resultado original entregado por el programa haciendo análisis por grupos de 3 *frames* con un factor de escala de umbral de 1.4. (b) Distinción entre los videos realizados en exteriores e interiores para el análisis de grupos de 3 *frames* y factor de escala de 1.4. (c) Resultado original entregado por el programa haciendo análisis por grupos de 5 *frames* con un factor de escala de umbral de 1.4. (d) Distinción entre los videos realizados en exteriores e interiores para el análisis de grupos de 5 *frames* y factor de escala de 1.4. (e) Resultado entregado por el programa haciendo análisis por grupos de 7 *frames* con un factor de escala de umbral de 1.4. (f) Distinción entre los videos realizados en exteriores e interiores para el análisis de grupos de 7 *frames* y factor de escala de 1.4.

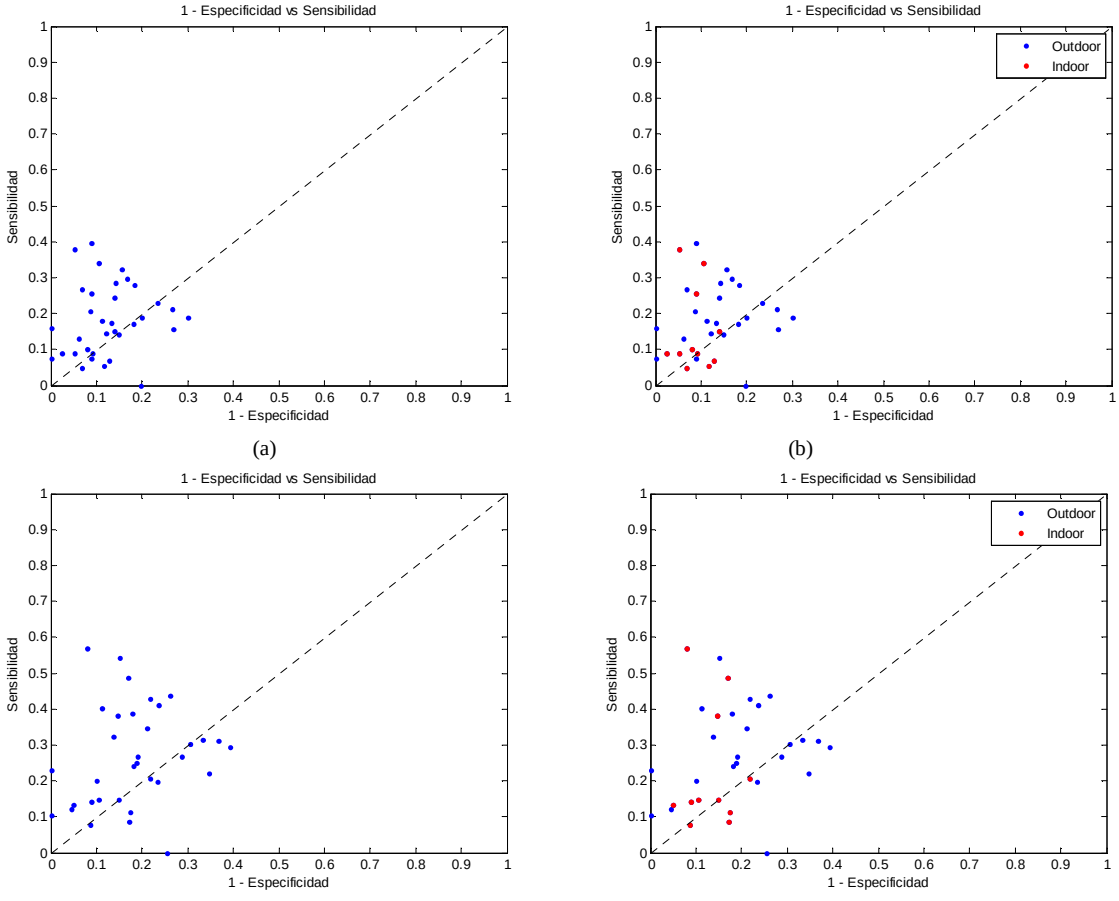
Por último se incrementó el factor de escala de umbral a un valor de 1.6. Se observa en la Figura 37 que el grupo de puntos se aglomeran hacia la parte inferior izquierda del gráfico, lo que implica que el nuevo valor de umbral descartó un número significativo tanto de aciertos como de fallos posicionando muchos de los puntos sobre la diagonal (aleatoriedad) y perdiendo la independencia acerca del tipo de video analizado (*outdoor* o *indoor*) ya que el algoritmo para este caso fue mas eficiente con los videos realizados en exteriores.



(a) (b)

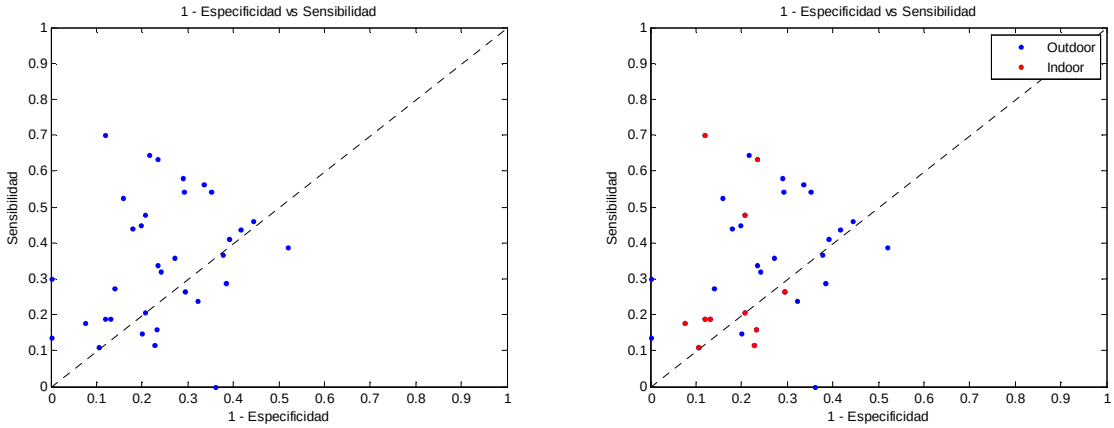
Fig. 37 (a) Resultado original entregado por el programa con un factor de escala de umbral de 1.6. (b) Distinción entre los videos realizados en exteriores e interiores.

En la Figura 38**,** luego de realizar el análisis de grupos, se verificó una dispersión elevada de los datos lo que es negativo al momento de tomar decisión ya que el algoritmo en este caso no se comporta como clasificador. La dependencia acerca del tipo de video persiste, desempeñándose de mejor manera para los tipos de videos realizados en exteriores.



(a) (b)

(c) (d)



(e) (f)

Fig. 38 (a) Resultado original entregado por el programa haciendo análisis por grupos de 3 *frames* con un factor de escala de umbral de 1.6. (b) Distinción entre los videos realizados en exteriores e interiores para el análisis de grupos de 3 *frames* y factor de escala de 1.6. (c) Resultado original entregado por el programa haciendo análisis por grupos de 5 *frames* con un factor de escala de umbral de 1.6. (d) Distinción entre los videos realizados en exteriores e interiores para el análisis de grupos de 5 *frames* y factor de escala de 1.6. (e) Resultado entregado por el programa haciendo análisis por grupos de 7 *frames* con un factor de escala de umbral de 1.6. (f) Distinción entre los videos realizados en exteriores e interiores para el análisis de grupos de 7 *frames* y factor de escala de 1.6.

6. CONCLUSIONES

Se cumplieron a cabalidad los objetivos trazados desde el planteamiento del desarrollo de este trabajo de grado que eran el construir perfiles de comportamiento de regiones de interés en una secuencia de video con el propósito de detectar anomalías. A nivel de detalle, los objetivos de realizar estimación de fondo correspondiente a la secuencia de video, identificar puntos de interés en objetos equiparados como no pertenecientes al fondo, agrupar puntos de interés en regiones o ventanas de interés, seguir regiones o ventanas de interés en una línea de tiempo para realizar un modelamiento de sus atributos característicos y clasificar los atributos entre comportamientos normales y anómalos fueron alcanzados satisfactoriamente.

Luego de realizar las pruebas se evaluó la experiencia como muy positiva y se constató el buen funcionamiento de la rutina desarrollada. El algoritmo fue capaz de detectar anomalías con un buen grado de acierto. El procedimiento que se llevó a cabo brindó un panorama expedito en pos de la detección de anomalías.

El algoritmo tiene la capacidad de determinar individualmente cuáles *frames* presentan o no anomalía durante el video mediante la utilización de un umbral adaptativo, lo que supone para trabajos posteriores, se pueda proponer una nueva forma de análisis en base a éstos resultados para así mejorar el desempeño de la detección de las anomalías, ya que la existencia de anomalía en un *frame*, puede suponer la existencia de anomalías en grupos de *frames*.

De los resultados se infirió que seleccionando un factor de escala de umbral adecuado se disminuyó ampliamente la dependencia del algoritmo por escenarios apropiados donde éste se desempeñara bien, ya que con un factor de escala propicio, factores como sombras, oclusión parcial, tipo de entorno (interiores y/o exteriores), resolución efectiva (distancia a la cámara), iluminación, número de sujetos que intervienen en la escena y ángulo y posición de la cámara fueron irrelevantes en mayor grado.

Lo más destacable del trabajo que se desarrolló son las propiedades del algoritmo que se obtuvo. Su capacidad de adaptabilidad, es decir, su aptitud de actualizar periódicamente las regiones de interés permitieron obtener información acorde a lo que estaba ocurriendo en cada instante de la escena analizada, por lo que brindó la posibilidad de tratar información en tiempo real (*online*) y así generar una detección temprana de anomalías, ésta es una de las virtudes del algoritmo.

Otra gran virtud que posee el programa desarrollado es que éste no requiere de un entrenamiento previo que proporcione los medios adecuados para un buen funcionamiento, por la razón que limitaría al algoritmo a desempeñarse de buena forma solamente en determinados tipos de secuencias y

situaciones que cumplan características específicas limitándolo altamente, por el contrario, no es necesario un entrenamiento o preparación del algoritmo, sino que todas las herramientas necesarias para una buena detección las obtiene de información histórica y reciente a la escena analizada.

El proceso seguido para identificar las anomalías fue acertado, ya que con solo los resultados de los atributos el algoritmo hubiese incurrido en errores. Hacer el cálculo del error de los atributos para las diferentes regiones de interés permitió descartar resultados erróneos y resaltar los válidos, adicionalmente, la suma de los errores para cada atributo ofreció un panorama con condiciones propicias para la toma de decisiones.

Se pudo constatar que los atributos tenían mejor desempeño dependiendo del contexto en el que se desarrollaran los hechos en la secuencia de video, es decir, existen atributos que son capaces de describir fielmente los acontecimientos ocurridos para actividades específicas dentro de la escena. Es por esto que el atributo de desplazamiento horizontal para videos en los que el sujeto de forma inesperada corría luego de haber estado caminando por algunos *frames* con normalidad mostró resultados acertados. Al estar el sujeto en determinado punto en un *frame* y en *frames* posteriores encontrarse en una posición completamente diferente significó pues, para este atributo, que el sujeto se estaba desplazando a gran velocidad comparado a la manera en que se desplazó normalmente, por lo que fue capaz de reproducir este comportamiento en valores elevados al momento de calcular el grado de desplazamiento horizontal del objeto de interés.

Así como ocurrió para el atributo de desplazamiento horizontal, ocurrió para el desplazamiento vertical, obviamente en otros contextos. Éste atributo describió con mejores resultados movimientos en las secuencias de video como caídas, saltos, desmayos, cuclillas, etc. y, en general, movimientos que implicaran que las regiones de interés en determinado instante de tiempo estuviesen posicionadas a una altura diferente de la que estuvieron posicionadas en *frames* cercanos.

Cabe destacar que los atributos restantes, *kurtosis*, asimetría y varianza fueron de gran ayuda para describir no sólo los movimientos en los que los atributos de desplazamiento tuvieron gran aporte sino también en los movimientos donde éstos mismos no contaban con un buen desempeño, como en peleas de sujetos y movimientos erráticos en general.

El factor más desafiante para hacer una buena detección de anomalías residía al momento de hacer sustracción de fondo, debido a que una mala segmentación del objeto de interés implicaba que los procesos posteriores a este se vieran afectados generando resultados poco confiables e inesperados. Videos donde el fondo fuera complejo y la iluminación no estuviese controlada generaba que la creación de las regiones de interés no fuera óptima, al observar regiones de interés con poca área, y como consecuencia de esto un *tracking* inadecuado debido a que podía identificar dicha región incorrectamente en el siguiente *frame*.

El análisis en grupos de *frames* fue pensado en primera instancia para mejorar los resultados obtenidos del algoritmo en detectar efectivamente anomalías y evitar en lo posible falsos positivos, sin embargo, al observar los resultados que se registraron, el análisis de grupos ayudó en nada respecto a este objetivo. Se verificó que conforme se aumentaba el número de *frames* en las agrupaciones, el resultado de la razón promedio entre la sensibilidad y 1-especificidad se degradaba, por lo que se infiere, que debe replantearse la forma en que se lleva a cabo este análisis para que los resultados que arroja el algoritmo se vean afectados positivamente.

Debido a que todos los procesos que se llevan a cabo durante la generación de los resultados y al hacer el perfilamiento *frame* a *frame*, la ejecución del algoritmo toma un tiempo considerable. Existen alternativas viables que contribuyen a mejorar el rendimiento del programa y su desempeño, se concluye pues, que replanteando el análisis de grupos se puede aumentar la cantidad de aciertos, esto es, no sólo considerando los *frames* anómalos en si mismos para realizar la agrupación, sino también, tener en cuenta cada atributo que determinó que dicho *frame* fuera anómalo, es decir, se realizarán grupos de *frames* únicamente a los *frames* que fueron marcados por el algoritmo como anómalos cuando sus cinco atributos así lo determinaron y no cuando cuatro o menos atributos destacaron anomalía en este punto. Esta forma de análisis pondera *frames* donde todos los atributos coincidieron

manifestando anomalía, siendo un buen camino para realzar la sensibilidad del algoritmo y consigo, la cantidad de aciertos.

Desde el punto de vista de rendimiento, se puede llevar un proceso donde se eliminen filas y columnas de pixeles de la secuencia de video sin perder información importante, lo que implica que el algoritmo tenga menos información que procesar y arroje resultados en menor tiempo, además, se puede llevar acabo una redefinición del algoritmo en un lenguaje de programación que brinde mejores bondades en cuanto a rendimiento como C, ya que se demostró que la detección de anomalías fue una realidad.

7. BIBLIOGRAFÍA

[1] Centro de Investigación del Consumidor, "Octavo Censo Nacional de Mermas,". *FENALCO, 2010.*

[2] Xiao-Ning Zhang, Jue Jiang, Zhi-Hu Liang, Chun-Liang Liu, "Skin Color Enhancement Based on Favorite Skin Color in HSV Color Space". *IEEE, 2010, pp. 1789-1793*

[3] Xiao C.H., Yung, N.H.C.: Corner detector based on global and local curvature properties. Optical Engineering, 47(5), p. 057008(2008).

[4] Cluster Analysis: Basic Concepts and Algorithms Kumar Cluster Analysis: Basic Concepts and Algorithms, Prepublication chapter from a book, PANG- NING TAN, MICHAEL STEINBACH, VIPIN KUMAR

[5] F. Essannouni, R. Oulad Haj Thami, D. Aboutajdine, A. Salam, "Simple noncircular correlation method for exhaustive sum square difference matching," *Optical Engineering 46 (10), 107004 ,October 2007.*

[6] Tao Xiang, Shaogang Gong , "Video behavior profiling for anomaly detection,". *IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 30, no. 5, pp. 893-908, May 2008.*

[7] Erhan Baki Ermis, Venkatesh Saligrama, Pierre-Marc Jodoin, Janusz Konrad, "Abnormal behavior detection and behavior matching for networked cameras,". *IEEE, 2008.*

[8] Tian Xuemin, Deng Xiaogang, **"**A Fault Detection Method Using Multi-Scale Kernel Principal Component Analysis," *Proceedings of the 27th Chinese Control Conference, pp. 25-29, July 16-18, 2008, Kunming,Yunnan, China.*

[9] Saligrama, V. Konrad, J. Jodoin, P., "Video Anomaly Identification," *Signal Processing Magazine, IEEE, pp. 18-33, september 2010*

[10] Yang Yang, Tian Guohui, Yun-Xia Liu, "A Fast Background Estimation Method For Vehicle Surveillance". *2010 IEEE.*

[11] Yanjiang Wang; Baozong Yuan, "Segmentation method for face detection in complex background," *ELECTRONICS LETTERS, pp. 213-214, 2000.*

[12] Zang, Qi and Klette, Renhard, "Parameter Analysis for Mixture of Gaussians Model," *Department of Computer Science, Tamaki Campus, The University of Auckland, New Zealand.*

[13] JunJie Liu, Anthony Jackas, Ala Al-Obaidi, Yonghuai Liu, "A comparative study of different corner detection methods". *Department of Computer Science, Aberystwyth University, UK.*

[14] Zhiyong Ye, Yijian Pei, Jihong Shi, "An Improved Algorithm for Harris Corner Detection". *IEEE, School of Information, Yunnan University, Kunming, China.*

[15] Yang Xingfang, Huang Yumei, Li Yan, "An improved SUSAN corner detection algorithm based on adaptive threshold". *2nd International Conference on Signal Processing Systems (ICSPS), pp. V2-613-V2-616. 2010 IEEE.*

[16] Prof. Mohammad Reza Alsharif , Md. Foisal Hossain and Miyahira Yoshihiko, "dction based on curvature scale space and adaptive thresholding,". *Department of Information Engineering, Ryukyu University. Japan.*

[17] Data Clustering: A Review A.K. JAIN, M.N. MURTY AND P.J. FLYNN ACM Computing Surveys, Vol. 31, No. 3, September 1999

[18] k-means Requires Exponentially Many Iterations Even in the Plane Andrea Vattani, University of California, San Diego 2009

[19] A systematic evaluation of different methods for initializing the K-means clustering algorithm, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, Ranjan Maitra, Anna D. Peterson and Arka P. Ghosh 2010

[20] S. Yu and J. Shi, "Multiclass Spectral Clustering," *Proc. Ninth IEEE Int'l Conf. Computer Vision, pp. 313-319, 2003.*
[21] Chris Stauffer W. Eric L. Grimson, "Learning patterns of activity using real-time tracking", Artifcial Intelligence Laboratory Massachusetts Institute of Technology Cambridge.
[22] S. Haykin, "Adaptative Filter Theory", 3rd Edition, Prentice Hall Information and System Sciences Series.
[23] Noble, J.A.: Finding corners. Image and Vision Computing, 6(2), 1988.
[24] Tom Fawcett, "ROC Graphs: Notes and Practical Considerations for Researchers," HP Laboratories, March 16, 2004
[25] M.J. Burgueñoa, J.L. García-Bastosb y J.M. González-Buitrago "Las curvas ROC en la evaluación de las pruebas diagnósticas", Medicina Clínica Vol. 104 Núm. 17. 1.995
[26] Image Processing Toolbox 7, User's Guide
[27] Compossed by Seth Benton, August, 2008
[28] Compossed by He Xiaochen, HKU EEE Dept. ITSR, April. 2005
[29] Compossed by Yue Wu, ECE Dept. TUFTS University, October. 2011

8. ANEXOS

ANEXO A: ALGORITMOS

1. Detección de Puntos de Interés

1.1 Lista inicial de esquinas candidatas

El listado inicial de las candidatas a esquinas se lleva a cabo por intermedio de la definición del j-ésimo contorno extraído, así,

$$A^j = \{P_1^j, P_2^j, \dots, P_N^j\},$$

donde $P_i^j = (x_i^j, y_i^j)$ son los pixeles en el contorno. N se refiere al número de pixeles en el contorno, y x_i^j, y_i^j son las coordenadas del i-ésimo pixel en el j-ésimo contorno.

Un contorno se define como cerrado si la distancia entre sus puntos de fin es considerablemente pequeña, de otra forma es abierto:

$$A^j = \begin{cases} cerrado\ si\ \left|\overline{P_1^J P_N^J}\right| < T, \\ abierto\ si\ \left|\overline{P_1^J P_N^J}\right| > T, \end{cases}$$

donde T se usa para determinar si dos puntos de fin están considerablemente cerca.

Para un contorno cerrado, se aplica convolución circular directamente para suavizar el contorno, sin embargo, para un contorno abierto un cierto número de puntos debe ser compensado en los dos extremos del contorno cuando ha sido suavizado.

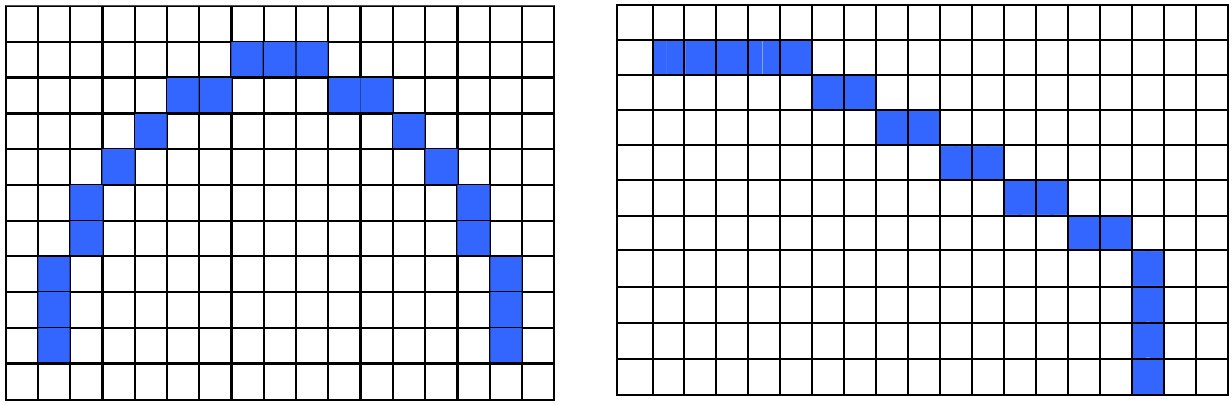
$A_{suavizado}^i = A^j \otimes g$; convolución del contorno con suavizador Gaussiano, g : función digital Gaussiana de con ancho controlado por σ.

Luego de esto, se calcula el valor de la curvatura de cada pixel del contorno,

$$K_j^j = \frac{\Delta x_i^j \Delta^2 y_i^j - \Delta^2 x_i^j \Delta y_i^j}{\left[\left(\Delta x_i^j\right)^2 + \left(\Delta y_i^j\right)^2\right]^{1.5}} \text{ para } i = 1, 2, \ldots, N, \qquad (1)$$

Donde $\Delta x_i^j = \left(x_{i+1}^j - x_{i-1}^j\right)/2$, $\Delta y_i^j = \left(y_{i+1}^j - y_{i-1}^j\right)/2$, $\Delta^2 x_i^j = \left(\Delta x_{i+1}^j - \Delta x_{i-1}^j\right)/2$, $\Delta^2 y_i^j = \left(\Delta y_{i+1}^j - \Delta y_{i-1}^j\right)/2$. De la ecuación (1), todos los máximos locales de la función curvatura son incluidos en la lista inicial de esquinas candidatas.

En la Figura 1. se muestran ejemplos tipos de esquinas circulares y obtusas.



a) b)

Figura 1. Ejemplos de: a) esquina circular y b) esquina obtusa.

1.2 Evaluación de esquinas

Se define el término de región de soporte (ROS) con el fin de discriminar entre esquinas circulares y obtusas.

La región de soporte de una esquina se define como el segmento del contorno delimitado por las dos esquinas más cercanas con curvatura mínima. El ROS de cada esquina se usa para calcular un umbral local adaptativo, donde u es la posición de la esquina candidata en el contorno, L_1+L_2 es el tamaño de la ROS centrada en u, y R es un coeficiente:

$$T(u) = R \times \overline{K} = R \times \frac{1}{L_1+L_2+1} \sum_{i=u-L_2}^{u+L_1} |K(i)|$$

$\overline{K}$: es la curvatura media de la ROS. Si la curvatura de una esquina candidata es mayor que $T(u)$, se declara dicha esquina como verdadera, de otra manera es eliminada de la lista.

La razón por la que se pueden eliminar las esquinas circulares es porque las esquinas obtusas caen con mayor rapidez en L_1+L_2 que una esquina circular en la misma ROS.
La curvatura media de una esquina es menor que la curvatura de una esquinas circular. Las esquinas circulares tienden a tener una curvatura menor que $T(u)$, en caso contrario las esquinas obtusas, aunque depende del valor seleccionado para R.
En teoría, con un valor de R seleccionado correctamente se puede lograr diferenciar entre estos dos tipos de esquinas sin mayor inconveniente.

1.3 Eliminación de esquinas falsas

La clave del éxito reside en la correcta definición del ángulo de una esquina, en particular, el ángulo de una esquina puede ser una característica ambigua que varía en función de su definición.

El hecho de desconocer a-priori la característica de curvatura global, motivó a proponer el método con el cual se logre determinar el rango apropiado para evaluar esquinas potencialmente candidatas vasado en lo definido como ROS.

En la Figura 2. se muestran 5 puntos etiquetados en una curva. Cada uno de ellos representa máximos locales de curvatura y pueden ser catalogados como esquinas.

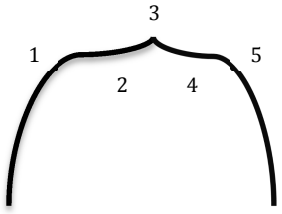


Figura 2. Caso de esquinas ambiguas.

Para el punto 3 se generará una nueva ROS que abarcará desde los puntos 2 a 4 y será considerada como esquinas por tener un ángulo agudo.

En otras palabras, los puntos 2 y 4 pueden ser potencialmente removidos después del proceso de eliminación de esquinas circulares, por tal motivo, el nuevo ROS para el punto 3 serán la región que abarca el 1 a 5.Luego de determinar las ROS de las esquinas candidatas el ángulo de cada esquina candidata se puede definir como aquel que está entre las líneas que unen el punto de esquinas en cuestión y los dos centros de masa de ambos lados de la ROS, donde los centros de masa se definen como la posición media de todos los pixeles de un brazo de la ROS.

Sin embargo, el proceso anterior falla cuando se trata de variaciones locales a lo largo de un arco.

Para considerar determinado ángulo $\angle$ como candidato a eliminar, debe ser suficientemente obtuso, para el caso de un arco, no es de gran ayuda que el arco se extienda demasiado (ROS grande).

Con el fin de evitar el problema anterior, se redefine el ángulo de una esquina usando tangente. Con éste método, tanto arcos como líneas pueden analizarse desde el mismo enfoque.

Descripción:

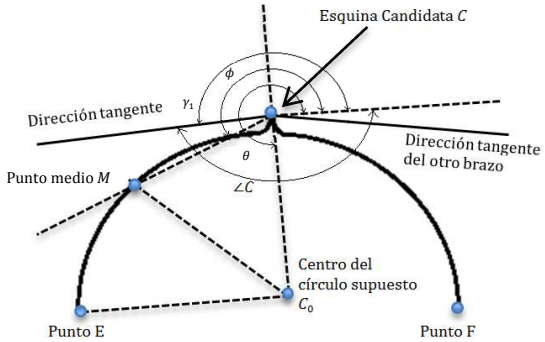
Primero, como se muestra en la Figura 3. a), y ubicado en un brazo de la ROS (de C a E), se ubican 3 puntos: C, un punto medio M y E. Si esos 3 puntos son colineales, la dirección de la tangente del brazo de aquella ROS se define de C a E, sino, se supone la existencia de un círculo de centro C_0 que tiene la misma distancia (radio de la curvatura de esa ROS) a los tres puntos.

$$C = (x_1, y_1), M = (x_2, y_2), E = (x_3, y_3)\ y\ C_0(x_0, y_0)$$

$$x_0 = \frac{({x_1}^2 + {y_1}^2)(y_2 - y_3) + ({x_2}^2 + {y_2}^2)(y_3 - y_1) + ({x_3}^2 + {y_3}^2)(y_1 - y_2)}{2[x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)]}$$

$$y_0 = \frac{({x_1}^2 + {y_1}^2)(y_2 - y_3) + ({x_2}^2 + {y_2}^2)(y_3 - y_1) + ({x_3}^2 + {y_3}^2)(y_1 - y_2)}{2[x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)]}$$

a) Definición de centro de masa.



b) Definición de tangente.

Figura 3. Definición de ángulos de una esquina. a) Definición de centro de masas, b) definición de tangente.

Segundo, se dibuja una línea de C a C_0, y se usa θ para representar la dirección de C a C_0, De la misma forma se usa Φ para denotar la dirección de C a M. Luego, se puede calcular la tangente de C para este brazo de la ROS, así:

$$\gamma_1 = \theta + \text{sign}\big(\sin(\Phi - \theta)\big) \cdot \frac{\pi}{2}$$

Como tercer paso, la tangente de la ROS de C a F se determina de la misma manera como se hizo de C a E. Las dos líneas tangentes producto del análisis anterior son las que formarán el ángulo de la esquina C.

$$\angle C = \begin{cases} |\gamma_1 - \gamma_2| & \text{si} \quad |\gamma_1 - \gamma_2| < \pi \\ 2\pi - |\gamma_1 - \gamma_2| & \text{otro caso} \end{cases}$$

Por último, el criterio de decisión,
C_i es una esquina verdadera si $\angle C_i \leq \theta_{obt}$
C_i es una esquina falsa si $\angle C_i > \theta_{obt}$

Donde θ_{obt} se refiere al máximo valor de ángulo obtuso que una esquina puede tener para que pueda ser considerada como esquina.

2. Seguimiento

El *frame* actual se divide en bloques cuadrados de $B \times B$ y se asume que cada uno de los bloques presenta movimiento de traslación por lo que es posible calcular un vector de movimiento, que describa el desplazamiento en las dos coordenadas. Para un bloque *g* que pertenece al *frame* actual, como se muestra en la Figura 4., se realiza una búsqueda en el *frame* de referencia sobre un área búsqueda f y con una ventana de búsqueda de tamaño fijo. El objetivo de la búsqueda es encontrar el bloque me mejor se ajuste y cuyo error sea mínimo, para este algoritmo esto se consigue minimizando la suma de diferencia de cuadrados (SSD).

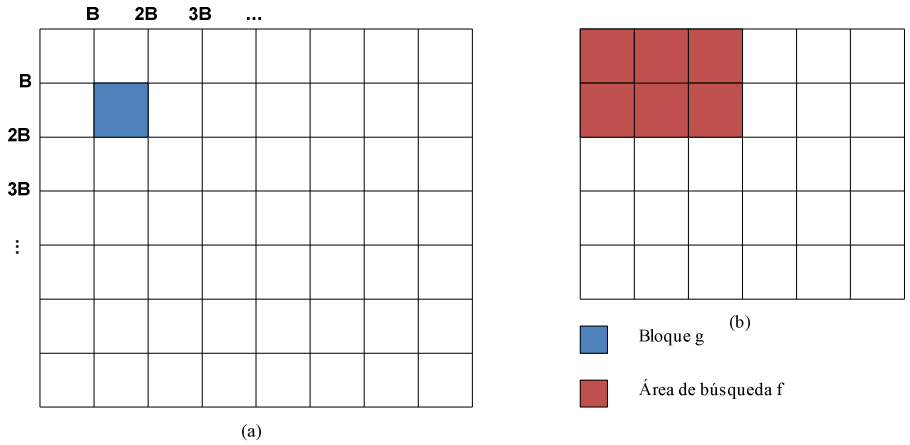


Figura 4. *Frames* a) *Frame* actual b) *Frame* de referencias

La expresión para la SSD es

$$SSD(x,y) = \sum_{k=0}^{B-1}\sum_{l=0}^{B-1}[f(x+k,y+l) - g(k,l)]^2$$

Se recorre todo el bloque g y el área de búsqueda, por otro lado x y y son las coordenadas del vector que se postula como vector de movimiento óptimo. Las coordenadas pertenecen al rango $[0, 2w]$, valores entre 0 y w indican un desplazamiento negativo, mientras que si se encuentran entre w y $2w$ indica un desplazamiento positivo.

Como los píxeles de la imagen solo asumen valores reales se puede realizar operaciones con números complejos para luego obtener la parte real; entonces

$$[f(x+k,y+l) - g(k,l)]^2 =$$
$$\Re\{[f(x+k,y+l) - g(k,l)]^2 + j\,[2g(k,l)f^2(x+k,y+l)]\; + jf(x+k,y+l)\}$$

Remplazando el resultado obtenido y rescribiendo la expresión de la SSD, tenemos

$$SSD(x,y) = \Re\{SSD(x,y) + j\sum_{k=0}^{B-1}\sum_{l=0}^{B-1}[2g(k,l)f^2(x+k,y+l)]\; + f(x+k,y+l)$$

El objetivo de este desarrollo matemático es rescribir la expresión de la suma de diferencias de cuadrados, para esto se plantean dos nuevas variables f_c y g_c

$$f_c(k,l) = f^2(k,l) + jf(k,l)$$

$$g_c(k,l) = 2jg(k,l) - 1$$

Con estas nuevas variables se puede rescribir la expresión para SSD,

$$f_c(x+k,y+l)g_c^*(k,l) =$$
$$-[f^2(x+k,y+l) - 2g(k,l)f(x+k,y+l)] - j[2g(k,l)f^2(x+k,y+l) + f(x+k,y+l)]$$

$$SSD(x,y) = -\Re[\sum_{k=0}^{B-1}\sum_{l=0}^{B-1} f_c(x+k,y+l)g_c^*(k,l)] + \sum_{k=0}^{B-1}\sum_{l=0}^{B-1} g_c^2(k,l)$$

Ésta última expresión es exactamente igual a la primera que se mencionó para la SSD, pero tiene dos cosas particulares, la primera es un término constante que corresponde al bloque g que se está analizando, con este término resulta inútil trabajar debido a que es constante y no depende de las

coordenadas del vector que se postula como vector óptimo; el primer término es el que resulta atractivo, ya que está en función del vector y es posible minimizarlo para minimizar la SSD. Éste término se puede ver como la correlación cruzada, ya que $f_c(x+k, y+l)g_c^*(k,l)$ hace referencia a la convolución entre f_c y g_c, en este caso g_c se deja quieta y f_c se desplaza. }

Utilizando la transformada rápida de Fourier (FFT) y sus propiedades se tiene el siguiente resultado para la convolución entre f_c y g_c

$$f_c * g_c = F_c(m,n)G_c^*(m,n)$$

Donde F_c y G_c^* corresponde a la transformadas de Fourier de f_c y g_c respectivamente.

El resultado final para la SSD es

$$SSD(x,y) = -\Re[IFFT(F_c(m,n)G_c^*(m,n))] + \sum_{k=0}^{B-1}\sum_{l=0}^{B-1} g_c^2(k,l)$$

ANEXO B: TABLAS DE RESULTADOS

Frame	Dv	Dh	V	K	S
26	O	O	O	O	O
27	O	O	O	O	O
28	O	O	O	O	O
29	O	O	O	O	O
30	O	O	O		
31	O	O			O
32	O			O	O
66		X		X	X
67		X	X	X	X
69		X		X	X
70		X	X	X	
78	X	X	X		
79	X	X	X	X	X
80	X	X	X	X	X
92	X	X	X	X	X
93		X	X	X	X
94	X	X	X		X
95	X	X	X	X	X
TOTAL VP(X)	6	11	9	9	9
TOTAL FP(O)	7	6	5	5	6
Anomalias = 11	0,54545455	1	0,81818182	0,81818182	0,81818182

Tabla 1. Desempeño de los atributos para el video C6.avi

Frame	Dv	Dh	V	K	S
22	O	O	O	O	O
23	X	X	X	X	X
24	X	X	X	X	
25	X	X	X		X
47		X		X	X
50		X		X	X
56	X			X	X
57	X			X	X
58	X			X	X
59	X			X	X
60	X			X	X
61	X	X	X		
62	X	X	X		
63	X	X	X		
64	X	X	X		
71	X	X		X	
81		X	X	X	X
82	X	X	X	X	X
83	X	X	X	X	
105	O	O		O	O
117			X	X	X
119			X	X	X
134	X			X	X
135	X			X	X
136		X	X	X	X
137	X		X	X	X
138	X		X	X	X
139	X	X		X	
140	X	X		X	
150	X	X	X		
151		X		X	X
152		X		X	X
153		X		X	X
154		X		X	X
155		X		X	X
171		X		X	X
173	X	X		X	X
174	X	X		X	X
175		X		X	X
183	X	X		X	X
184	X	X		X	X
185	X	X		X	X
199	X	X	X	X	X
200	X	X	X	X	X
210		X	X	X	X
221	X	X		X	
222	X	X		X	
223	X	X		X	X
224	X			X	X
225	X			X	X
231	X	X	X		
232	X	X	X		
233	X	X	X		
234	X	X	X		
235	X	X	X		
236	X	X		X	X
237	X	X		X	X
238	X			X	X
240	X	X			X
251	O			O	O
255	O			O	O
258		X	X	X	X
259		X	X	X	X
270	X	X	X		X
276	X	X	X	X	X
277	X	X	X	X	X
278	X	X	X	X	X
279		X		X	X
280		X		X	X
287	X		X		X
288	X			X	X
308		X	X	X	
309	X	X	X	X	X
310	X	X	X	X	X
TOTAL VP(X)	51	54	34	56	52
TOTAL FP(O)	3	1	0	3	3
Anomalias = 70	0,72857143	0,77142857	0,48571429	0,8	0,74285714

Tabla 2. Desempeño de los atributos para el video C19.avi

Frame	Dv	Dh	V	K	S
26	O		O		O
28	O		O	O	
30	O		O	O	
55	X	X	X	X	X
59	X	X	X	X	X
60	X	X		X	X
66	X		X	X	X
67	X	X	X		
68	X	X	X		X
69	X	X	X	X	X
70	X	X	X	X	X
75	X		X		X
81		X	X		X
83			X	X	X
84			X	X	X
91	X	X		X	
92	X	X	X	X	X
93	X	X	X	X	X
94	X	X	X	X	X
95	X	X	X	X	X
104		X	X	X	X
105	X	X	X	X	X
118		X	X	X	X
119		X	X	X	X
127	X	X	X		
128	X	X	X		
130		X		X	X
146	X			X	X
147	X		X		X
148	X			X	X
149	X	X		X	X
150	X		X	X	X
156	X	X	X		
157	X	X	X		
158	X	X	X		
159	X	X	X		
160	X	X	X		
164	X	X		X	X
165	X	X		X	X
174		X	X	X	X
175		X		X	X
176	X	X	X		
177	X	X	X		
178	X	X	X		
179	X	X	X		
180	X	X	X		
196	O	O	O		
197	O	O	O		
198	O	O	O		
199	O	O	O		
205		O	O	O	
206		O	O		O
TOTAL VP(X)	7	35	34	26	29
TOTAL FP(O)	7	6	9	3	2
Anomalias =43	0,1627907	0,81395349	0,79069767	0,60465116	0,6744186

Tabla 3. Desempeño de los atributos para el video D3.avi

Frame	Dv	Dh	V	K	S
25	O	O			O
37	O	O		O	O
39	O	O			O
40	O	O			O
55		O	O		O
59	O			O	O
74			O	O	O
75			O	O	O
81	O	O	O		
103		O	O		O
111	X			X	X
112	X			X	X
113	X			X	X
114	X			X	X
122		X	X		X
124	X	X	X		X
125	X	X	X		X
133	X		X	X	X
134	X		X	X	X
155	O		O		O
167	O	O		O	
169	O	O		O	
170	O	O		O	
174	O		O	O	
175	O		O		O
188	O	O		O	
199	O			O	O
202	O	O	O	O	
205	O	O		O	
TOTAL VP(X)	8	3	5	6	9
TOTAL FP(O)	16	13	9	12	12
Anomalias = 9	0,88888889	0,33333333	0,55555556	0,66666667	1

Tabla 4. Desempeño de los atributos para el video B8.avi

ANEXO C: CÓDIGO

1. Función mixog (Sustracción de Fondo)

```
function [cout,auxiliar,G,COR]=mixog(source,l,h,G)
% source=mmreader('input.avi');
fr = read(source,l); % read in 1st frame as background frame
% fr=imresize(fr,[120 160]);
fr_bw = rgb2gray(fr); % convert background to greyscale
fr_size = size(fr);
width = fr_size(2);
height = fr_size(1);
fg = zeros(height, width);
bg_bw = zeros(height, width);

% ---------------------- mog variables ---------------------------------

C = 3; % number of gaussian components (typically 3-5)
M = 3; % number of background components
D = 2.5; % positive deviation threshold
alpha = 0.01; % learning rate (between 0 and 1) (from paper 0.01)
thresh = 0.25; % foreground threshold (0.25 or 0.75 in paper)
sd_init = 6; % initial standard deviation (for new components) var = 36 in
paper
w = zeros(height,width,C); % initialize weights array
mean = zeros(height,width,C); % pixel means
```

```
sd = zeros(height,width,C); % pixel standard deviations
u_diff = zeros(height,width,C); % difference of each pixel from mean
p = alpha/(1/C); % initial p variable (used to update mean and sd)
rank = zeros(1,C); % rank of components (w/sd)


% --------------------- initialize component means and weights -----------

pixel_depth = 8; % 8-bit resolution
pixel_range = 2^pixel_depth -1; % pixel range (# of possible values)

for i=1:height
for j=1:width
for k=1:C

mean(i,j,k) = rand*pixel_range; % means random (0-255)
w(i,j,k) = 1/C; % weights uniformly dist
sd(i,j,k) = sd_init; % initialize to sd_init

end
end
end
frames=source.NumberOfFrames;
%--------------------- process frames ---------------------------------

for n = l:h

fr = read(source,n); % read in frame
% fr=imresize(fr,[120 160]);

fr_bw = rgb2gray(fr); % convert frame to grayscale

% calculate difference of pixel values from mean
for m=1:C
u_diff(:,:,m) = abs(double(fr_bw) - double(mean(:,:,m)));
end

% update gaussian components for each pixel
for i=1:height
for j=1:width

match = 0;
for k=1:C
if (abs(u_diff(i,j,k)) <= D*sd(i,j,k)) % pixel matches component

match = 1; % variable to signal component match

% update weights, mean, sd, p
w(i,j,k) = (1-alpha)*w(i,j,k) + alpha;
p = alpha/w(i,j,k);
mean(i,j,k) = (1-p)*mean(i,j,k) + p*double(fr_bw(i,j));
sd(i,j,k) = sqrt((1-p)*(sd(i,j,k)^2) + p*((double(fr_bw(i,j)) -
mean(i,j,k)))^2);
else % pixel doesn't match component
w(i,j,k) = (1-alpha)*w(i,j,k); % weight slighly decreases

end
end

w(i,j,:) = w(i,j,:)./sum(w(i,j,:));
```

```
bg_bw(i,j)=0;
for k=1:C
bg_bw(i,j) = bg_bw(i,j)+ mean(i,j,k)*w(i,j,k);
end

% if no components match, create new component
if (match == 0)
[min_w, min_w_index] = min(w(i,j,:));
mean(i,j,min_w_index) = double(fr_bw(i,j));
sd(i,j,min_w_index) = sd_init;
end

rank = w(i,j,:)./sd(i,j,:); % calculate component rank
rank_ind = [1:1:C];

% sort rank values
for k=2:C
for m=1:(k-1)

if (rank(:,:,k) > rank(:,:,m))
% swap max values
rank_temp = rank(:,:,m);
rank(:,:,m) = rank(:,:,k);
rank(:,:,k) = rank_temp;

% swap max index values
rank_ind_temp = rank_ind(m);
rank_ind(m) = rank_ind(k);
rank_ind(k) = rank_ind_temp;

end
end
end

% calculate foreground
match = 0;
k=1;

fg(i,j) = 0;
while ((match == 0)&&(k<=M))

if (w(i,j,rank_ind(k)) >= thresh)
if (abs(u_diff(i,j,rank_ind(k))) <= D*sd(i,j,rank_ind(k)))
fg(i,j) = 0;
match = 1;
else
fg(i,j) = fr_bw(i,j);
end
end
k = k+1;
end
end
end

% figure(1),subplot(3,1,1),imshow(fr)
% subplot(3,1,2),imshow(uint8(bg_bw))
% subplot(3,1,3),
% b = bwmorph(fg,'close');
% fg= bwmorph(b,'open');
% figure;
% imshow((fg))
```

```
auxiliar=zeros(height, width,3);
auxiliar(:,:,1)=im2bw(fg);
auxiliar(:,:,2)=im2bw(fg);
auxiliar(:,:,3)=im2bw(fg);

G(n,1)=im2frame(auxiliar);

end

 I=bwmorph(fg,'close');
 [L,n] = bwlabel(I);
 S = regionprops(L);
 X=[S.Area];
%  BB=[S.BoundingBox];
 [areamax, pos]=max(X);
 Seg=ismember(L,pos);

% %  Seg=imfill(Seg);
%  figure;
%  imshow(Seg)
BB=regionprops(Seg);
COR=BB.BoundingBox;
COR=floor(COR);
frame=read(source,h);
I=(frame(COR(2):COR(2)+COR(4),COR(1):COR(1)+COR(3),:));
% figure
% imshow(I)
[cout,marked] = corner(I,[],[],[],0.2);
cout(:,1)=cout(:,1)+COR(2);
cout(:,2)=cout(:,2)+COR(1);
cout=round(cout);
auxiliar=zeros(height, width,3);
auxiliar(COR(2):COR(2)+COR(4),COR(1):COR(1)+COR(3),:)=I;
% auxiliar(COR(2):COR(2)+COR(4),COR(1):COR(1)+COR(3),1)=marked;
% auxiliar(COR(2):COR(2)+COR(4),COR(1):COR(1)+COR(3),2)=marked;
% auxiliar(COR(2):COR(2)+COR(4),COR(1):COR(1)+COR(3),3)=marked;
auxiliar=uint8(auxiliar);
```

2. Función corner (Detección de Puntos de Interés)

```
function [cout,marked_img]=corner(varargin)

%   CORNER Find corners in intensity image.
%
%       CORNER works by the following step:
%       1.  Apply the Canny edge detector to the gray level image and
obtain a
%       binary edge-map.
%       2.  Extract the edge contours from the edge-map, fill the gaps in
the
%       contours.
%       3.  Compute curvature at a low scale for each contour to retain all
%       true corners.
%       4.  All of the curvature local maxima are considered as corner
%       candidates, then rounded corners and false corners due to boundary
%       noise and details were eliminated.
%       5.  End points of line mode curve were added as corner, if they are
not
%       close to the above detected corners.
%
%       Syntax :
```

```
%       [cout,marked_img]=corner(I,C,T_angle,sig,H,L,Endpiont,Gap_size)
%
%       Input :
%       I -  the input image, it could be gray, color or binary image. If I
is
%           empty([]), input image can be get from a open file dialog box.
%       C -  denotes the minimum ratio of major axis to minor axis of an
ellipse,
%           whose vertex could be detected as a corner by proposed
detector.
%           The default value is 1.5.
%       T_angle -  denotes the maximum obtuse angle that a corner can have
when
%           it is detected as a true corner, default value is 162.
%       Sig -  denotes the standard deviation of the Gaussian filter when
%           computeing curvature. The default sig is 3.
%       H,L -  high and low threshold of Canny edge detector. The default
value
%           is 0.35 and 0.
%       Endpoint -  a flag to control whether add the end points of a curve
%           as corner, 1 means Yes and 0 means No. The default value is 1.
%       Gap_size -  a paremeter use to fill the gaps in the contours, the
gap
%           not more than gap_size were filled in this stage. The default
%           Gap_size is 1 pixels.
%
%       Output :
%       cout -  a position pair list of detected corners in the input
image.
%       marked_image -  image with detected corner marked.
%
%       Examples
%       -------
%       I = imread('alumgrns.tif');
%       cout = corner(I,[],[],[],0.2);
%
%       [cout, marked_image] = corner;
%
%       cout = corner([],1.6,155);
%
%
%   Composed by He Xiaochen
%   HKU EEE Dept. ITSR, Apr. 2005
%
%   Algorithm is derived from :
%       X.C. He and N.H.C. Yung, Curvature Scale Space Corner Detector with
%       Adaptive Threshold and Dynamic Region of Support, Proceedings of
the
%       17th International Conference on Pattern Recognition, 2:791-794,
August 2004.
%   Improved algorithm is included in :
%       X.C. He and N.H.C. Yung, “Corner detector based on global and local
curvature properties”,
%       Optical Engineering, 47(5), pp: 057008, 2008.
%

[I,C,T_angle,sig,H,L,Endpoint,Gap_size] = parse_inputs(varargin{:});

if size(I,3)==3
    I=rgb2gray(I); % Transform RGB image to a Gray one.
end

tic
```

```
BW=EDGE(I,'canny',[L,H]);  % Detect edges
time_for_detecting_edge=toc

tic
[curve,curve_start,curve_end,curve_mode,curve_num]=extract_curve(BW,Gap_siz
e);  % Extract curves
time_for_extracting_curve=toc

tic
cout=get_corner(curve,curve_start,curve_end,curve_mode,curve_num,BW,sig,End
point,C,T_angle); % Detect corners
time_for_detecting_corner=toc

img=I;
for i=1:size(cout,1)
    img=mark(img,cout(i,1),cout(i,2),5);
end
marked_img=img;
% figure(2)
% imshow(marked_img);
% title('Detected corners')
% imwrite(marked_img,'corner.jpg');

function
[curve,curve_start,curve_end,curve_mode,cur_num]=extract_curve(BW,Gap_size)

%   Function to extract curves from binary edge map, if the endpoint of a
%   contour is nearly connected to another endpoint, fill the gap and
continue
%   the extraction. The default gap size is 1 pixles.

[L,W]=size(BW);
BW1=zeros(L+2*Gap_size,W+2*Gap_size);
BW_edge=zeros(L,W);
BW1(Gap_size+1:Gap_size+L,Gap_size+1:Gap_size+W)=BW;
[r,c]=find(BW1==1);
cur_num=0;

while size(r,1)>0
    point=[r(1),c(1)];
    cur=point;
    BW1(point(1),point(2))=0;
    [I,J]=find(BW1(point(1)-Gap_size:point(1)+Gap_size,point(2)-
Gap_size:point(2)+Gap_size)==1);
    while size(I,1)>0
        dist=(I-Gap_size-1).^2+(J-Gap_size-1).^2;
        [min_dist,index]=min(dist);
        point=point+[I(index),J(index)]-Gap_size-1;
        cur=[cur;point];
        BW1(point(1),point(2))=0;
        [I,J]=find(BW1(point(1)-Gap_size:point(1)+Gap_size,point(2)-
Gap_size:point(2)+Gap_size)==1);
    end

    % Extract edge towards another direction
    point=[r(1),c(1)];
    BW1(point(1),point(2))=0;
    [I,J]=find(BW1(point(1)-Gap_size:point(1)+Gap_size,point(2)-
Gap_size:point(2)+Gap_size)==1);
    while size(I,1)>0
        dist=(I-Gap_size-1).^2+(J-Gap_size-1).^2;
        [min_dist,index]=min(dist);
```

```
        point=point+[I(index),J(index)]-Gap_size-1;
        cur=[point;cur];
        BW1(point(1),point(2))=0;
        [I,J]=find(BW1(point(1)-Gap_size:point(1)+Gap_size,point(2)-
Gap_size:point(2)+Gap_size)==1);
    end

    if size(cur,1)>(size(BW,1)+size(BW,2))/25
        cur_num=cur_num+1;
        curve{cur_num}=cur-Gap_size;
    end
    [r,c]=find(BW1==1);

end

for i=1:cur_num
    curve_start(i,:)=curve{i}(1,:);
    curve_end(i,:)=curve{i}(size(curve{i},1),:);
    if (curve_start(i,1)-curve_end(i,1))^2+...
        (curve_start(i,2)-curve_end(i,2))^2<=32
        curve_mode(i,:)='loop';
    else
        curve_mode(i,:)='line';
    end

    BW_edge(curve{i}(:,1)+(curve{i}(:,2)-1)*L)=1;
end
% figure(1)
% imshow(~BW_edge)
% title('Edge map')
% imwrite(~BW_edge,'edge.jpg');


function
cout=get_corner(curve,curve_start,curve_end,curve_mode,curve_num,BW,sig,End
point,C,T_angle)

corner_num=0;
cout=[];

GaussianDieOff = .0001;
pw = 1:30;
ssq = sig*sig;
width = max(find(exp(-(pw.*pw)/(2*ssq))>GaussianDieOff));
if isempty(width)
    width = 1;
end
t = (-width:width);
gau = exp(-(t.*t)/(2*ssq))/(2*pi*ssq);
gau=gau/sum(gau);

for i=1:curve_num;
    x=curve{i}(:,1);
    y=curve{i}(:,2);
    W=width;
    L=size(x,1);
    if L>W

        % Calculate curvature
        if curve_mode(i,:)=='loop'
            x1=[x(L-W+1:L);x;x(1:W)];
            y1=[y(L-W+1:L);y;y(1:W)];
```

```
        else
            x1=[ones(W,1)*2*x(1)-x(W+1:-1:2);x;ones(W,1)*2*x(L)-x(L-1:-1:L-
W)];
            y1=[ones(W,1)*2*y(1)-y(W+1:-1:2);y;ones(W,1)*2*y(L)-y(L-1:-1:L-
W)];
        end

        xx=conv(x1,gau);
        xx=xx(W+1:L+3*W);
        yy=conv(y1,gau);
        yy=yy(W+1:L+3*W);
        Xu=[xx(2)-xx(1) ; (xx(3:L+2*W)-xx(1:L+2*W-2))/2 ; xx(L+2*W)-
xx(L+2*W-1)];
        Yu=[yy(2)-yy(1) ; (yy(3:L+2*W)-yy(1:L+2*W-2))/2 ; yy(L+2*W)-
yy(L+2*W-1)];
        Xuu=[Xu(2)-Xu(1) ; (Xu(3:L+2*W)-Xu(1:L+2*W-2))/2 ; Xu(L+2*W)-
Xu(L+2*W-1)];
        Yuu=[Yu(2)-Yu(1) ; (Yu(3:L+2*W)-Yu(1:L+2*W-2))/2 ; Yu(L+2*W)-
Yu(L+2*W-1)];
        K=abs((Xu.*Yuu-Xuu.*Yu)./((Xu.*Xu+Yu.*Yu).^1.5));
        K=ceil(K*100)/100;

        % Find curvature local maxima as corner candidates
        extremum=[];
        N=size(K,1);
        n=0;
        Search=1;

        for j=1:N-1
            if (K(j+1)-K(j))*Search>0
                n=n+1;
                extremum(n)=j;  % In extremum, odd points is minima and
even points is maxima
                Search=-Search;
            end
        end
        if mod(size(extremum,2),2)==0
            n=n+1;
            extremum(n)=N;
        end

        n=size(extremum,2);
        flag=ones(size(extremum));

        % Compare with adaptive local threshold to remove round corners
        for j=2:2:n
            %I=find(K(extremum(j-1):extremum(j+1))==max(K(extremum(j-
1):extremum(j+1))));
            %extremum(j)=extremum(j-1)+round(mean(I))-1; % Regard middle
point of plateaus as maxima

            [x,index1]=min(K(extremum(j):-1:extremum(j-1)));
            [x,index2]=min(K(extremum(j):extremum(j+1)));
            ROS=K(extremum(j)-index1+1:extremum(j)+index2-1);
            K_thre(j)=C*mean(ROS);
            if K(extremum(j))<K_thre(j)
                flag(j)=0;
            end
        end
        extremum=extremum(2:2:n);
        flag=flag(2:2:n);
        extremum=extremum(find(flag==1));
```

```
        % Check corner angle to remove false corners due to boundary noise
and trivial details
        flag=0;
        smoothed_curve=[xx,yy];
        while sum(flag==0)>0
            n=size(extremum,2);
            flag=ones(size(extremum));
            for j=1:n
                if j==1 & j==n

ang=curve_tangent(smoothed_curve(1:L+2*W,:),extremum(j));
                elseif j==1

ang=curve_tangent(smoothed_curve(1:extremum(j+1),:),extremum(j));
                elseif j==n
                    ang=curve_tangent(smoothed_curve(extremum(j-
1):L+2*W,:),extremum(j)-extremum(j-1)+1);
                else
                    ang=curve_tangent(smoothed_curve(extremum(j-
1):extremum(j+1),:),extremum(j)-extremum(j-1)+1);
                end
                if ang>T_angle & ang<(360-T_angle)
                    flag(j)=0;
                end
            end

            if size(extremum,2)==0
                extremum=[];
            else
                extremum=extremum(find(flag~=0));
            end
        end

        extremum=extremum-W;
        extremum=extremum(find(extremum>0 & extremum<=L));
        n=size(extremum,2);
        for j=1:n
            corner_num=corner_num+1;
            cout(corner_num,:)=curve{i}(extremum(j),:);
        end
    end
end


% Add Endpoints
if Endpoint
    for i=1:curve_num
        if size(curve{i},1)>0 & curve_mode(i,:)=='line'

            % Start point compare with detected corners
            compare_corner=cout-ones(size(cout,1),1)*curve_start(i,:);
            compare_corner=compare_corner.^2;
            compare_corner=compare_corner(:,1)+compare_corner(:,2);
            if min(compare_corner)>25       % Add end points far from
detected corners
                corner_num=corner_num+1;
                cout(corner_num,:)=curve_start(i,:);
            end

            % End point compare with detected corners
            compare_corner=cout-ones(size(cout,1),1)*curve_end(i,:);
```

```
            compare_corner=compare_corner.^2;
            compare_corner=compare_corner(:,1)+compare_corner(:,2);
            if min(compare_corner)>25
                corner_num=corner_num+1;
                cout(corner_num,:)=curve_end(i,:);
            end
        end
    end
end


function ang=curve_tangent(cur,center)

for i=1:2
    if i==1
        curve=cur(center:-1:1,:);
    else
        curve=cur(center:size(cur,1),:);
    end
    L=size(curve,1);

    if L>3
        if sum(curve(1,:)~=curve(L,:))~=0
            M=ceil(L/2);
            x1=curve(1,1);
            y1=curve(1,2);
            x2=curve(M,1);
            y2=curve(M,2);
            x3=curve(L,1);
            y3=curve(L,2);
        else
            M1=ceil(L/3);
            M2=ceil(2*L/3);
            x1=curve(1,1);
            y1=curve(1,2);
            x2=curve(M1,1);
            y2=curve(M1,2);
            x3=curve(M2,1);
            y3=curve(M2,2);
        end

        if abs((x1-x2)*(y1-y3)-(x1-x3)*(y1-y2))<1e-8  % straight line
            tangent_direction=angle(complex(curve(L,1)-
curve(1,1),curve(L,2)-curve(1,2)));
        else
            % Fit a circle
            x0 = 1/2*(-y1*x2^2+y3*x2^2-y3*y1^2-y3*x1^2-
y2*y3^2+x3^2*y1+y2*y1^2-y2*x3^2-y2^2*y1+y2*x1^2+y3^2*y1+y2^2*y3)/(-
y1*x2+y1*x3+y3*x2+x1*y2-x1*y3-x3*y2);
            y0 = -1/2*(x1^2*x2-x1^2*x3+y1^2*x2-y1^2*x3+x1*x3^2-x1*x2^2-
x3^2*x2-y3^2*x2+x3*y2^2+x1*y3^2-x1*y2^2+x3*x2^2)/(-y1*x2+y1*x3+y3*x2+x1*y2-
x1*y3-x3*y2);
            % R = (x0-x1)^2+(y0-y1)^2;

            radius_direction=angle(complex(x0-x1,y0-y1));
            adjacent_direction=angle(complex(x2-x1,y2-y1));
            tangent_direction=sign(sin(adjacent_direction-
radius_direction))*pi/2+radius_direction;
        end

    else % very short line
```

```
        tangent_direction=angle(complex(curve(L,1)-curve(1,1),curve(L,2)-
curve(1,2)));
    end
    direction(i)=tangent_direction*180/pi;
end
ang=abs(direction(1)-direction(2));



function img1=mark(img,x,y,w)

[M,N,C]=size(img);
img1=img;

if isa(img,'logical')
    img1(max(1,x-floor(w/2)):min(M,x+floor(w/2)),max(1,y-
floor(w/2)):min(N,y+floor(w/2)),:)=...
        (img1(max(1,x-floor(w/2)):min(M,x+floor(w/2)),max(1,y-
floor(w/2)):min(N,y+floor(w/2)),:)<1);
    img1(x-floor(w/2)+1:x+floor(w/2)-1,y-floor(w/2)+1:y+floor(w/2)-1,:)=...
        img(x-floor(w/2)+1:x+floor(w/2)-1,y-floor(w/2)+1:y+floor(w/2)-1,:);
else
    img1(max(1,x-floor(w/2)):min(M,x+floor(w/2)),max(1,y-
floor(w/2)):min(N,y+floor(w/2)),:)=...
        (img1(max(1,x-floor(w/2)):min(M,x+floor(w/2)),max(1,y-
floor(w/2)):min(N,y+floor(w/2)),:)<128)*255;
    img1(x-floor(w/2)+1:x+floor(w/2)-1,y-floor(w/2)+1:y+floor(w/2)-1,:)=...
        img(x-floor(w/2)+1:x+floor(w/2)-1,y-floor(w/2)+1:y+floor(w/2)-1,:);
end


function [I,C,T_angle,sig,H,L,Endpoint,Gap_size] = parse_inputs(varargin);

error(nargchk(0,8,nargin));

Para=[1.5,162,3,0.35,0,1,1]; %Default experience value;

if nargin>=2
    I=varargin{1};
    for i=2:nargin
        if size(varargin{i},1)>0
            Para(i-1)=varargin{i};
        end
    end
end

if nargin==1
    I=varargin{1};
end

if nargin==0 | size(I,1)==0
    [fname,dire]=uigetfile('*.bmp;*.jpg;*.gif','Open the image to be
detected');
    I=imread([dire,fname]);
end

C=Para(1);
T_angle=Para(2);
sig=Para(3);
H=Para(4);
L=Para(5);
```

```
Endpoint=Para(6);
Gap_size=Para(7);
```

3. Función SSDXCORR (Función de Tracking)

```
function [S,c,r,w,h] = SSDXCORR(f,t)

% function SSDXCORR: accelerated SSD using XCORR
%
% Detail method information, check the paper below
% ------------------------------------------------------------------------
% F. Essannouni, R. Oulad Haj Thami, D. Aboutajdine and A. Salam,
% "Simple noncircular correlation method for exhaustive sum square
difference matching",
% Opt. Eng. 46, 107004 (Oct 03, 2007); doi:10.1117/1.2786469
% ------------------------------------------------------------------------
%
% Inputs: f = frame image, 2 dimensional image
%         t = template image, 2 diminsional image
% Output: S = SSD score space
%         figure with the global maximum marked as the ROI
%
% ------------------------------------------------------------------------
% By Yue Wu
% ECE Dept
% Tufts University
% 04/10/2011
% ------------------------------------------------------------------------
%
% Demo:
% f = imread('cameraman.tif');figure,imshow(f),title('Frame Image')
% t = imcrop(f,[190,104,31,65]);figure,imshow(t),title('Template Image')
% S = SSDXCORR(f,t);
%
% ------------------------------------------------------------------------

% Initialization

t = double(t);
f = double(f);

% Complex template construction
tc = 2*t*1i-1;
fc = f.^2+f*1i;

% SSD using XCORR
tc = rot90(tc,2);
m = conv2(fc,conj(tc),'same');
S = real(m);

% Result display
% figure,imshow(uint8(f),[]),colormap(gray)
[v,ind] = max(S(:));
[c,r] = ind2sub([size(S,1),size(S,2)],ind);
[w,h] = size(t);
% rectangle('Position',[r-round(h/2), c-round(w/2), h,
w],'EdgeColor','g','LineWidth',2);
```

4. Interfaz Gráfica del algoritmo.

```
function varargout = GUI_Final(varargin)
% GUI_FINAL M-file for GUI_Final.fig
%      GUI_FINAL, by itself, creates a new GUI_FINAL or raises the existing
%      singleton*.
%
%      H = GUI_FINAL returns the handle to a new GUI_FINAL or the handle to
%      the existing singleton*.
%
%      GUI_FINAL('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI_FINAL.M with the given input
arguments.
%
%      GUI_FINAL('Property','Value',...) creates a new GUI_FINAL or raises
the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before GUI_Final_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to GUI_Final_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 GUI_Final

% Last Modified by GUIDE v2.5 12-Mar-2012 17:40:46

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUI_Final_OpeningFcn, ...
                   'gui_OutputFcn',  @GUI_Final_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 GUI_Final is made visible.
function GUI_Final_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 GUI_Final (see VARARGIN)

% Choose default command line output for GUI_Final
handles.output = hObject;
```

```
% Update handles structure
guidata(hObject, handles);

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


% --- Outputs from this function are returned to the command line.
function varargout = GUI_Final_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)
% FINAL_6

% Limpiar Comand Window y Workspace
clc
% clear all

% Lectura del video y almacenamiento en una variale, se obtiene información
% a partir del video como número de frames y duración.
Name=get(handles.edit1,'String');
Video=mmreader(Name);
Info=mmfileinfo(Name);
Frames=Video.NumberOfFrames; % Número de Frames del Video
Duracion=Info.Duration; % Duración del Video
Tasa=round(Frames/Duracion); % Frames/s

% Se define el tamaño de la ventana, el numero de ROI's, orden de predictor
% y todos los elementos de memoria que almacenarán los datos del video
% (Tracking, Predicción, Error)
M=1/2; % Factor de escala de la ventana
l=1; % Condiciones iniciales para usarse con la función mixog
h=(Tasa/3)*M;
minimo=1; % Condiciones que definene el tamaño de la ventana
maximo=2*h;
k=get(handles.edit2,'String');
k=str2num(k);
ROIS=zeros(k,4);

f=get(handles.edit4,'String');
f=str2num(f); % Factor de escala de umbral usado para determinar la
anomalia.

N=h-1;

Orden=2; % Orden para el predictor lineal

% Se crean vectores para almacenar los datos obtenidos de tracking y de
% predicción de cada uno de los atributos. En 1 se guarda tracking, en 2
% predicción, en 3 se guarda los datos usados para ser ingresados al
% predictor lineal
Desplazamientov=zeros(Frames,3,k);
```

```
Desplazamientoh=zeros(Frames,3,k);
Varianza=zeros(Frames,3,k);
Kurtosis=zeros(Frames,3,k);
Skewness=zeros(Frames,3,k);

% Vectores usados para almacenar la diferencia entre los datos obtenidos de
% tracking y los datos obtenidos mediante el predictor linel, se hace con
% cada atributo
edv=zeros(Frames,1,k);
edh=zeros(Frames,1,k);
ev=zeros(Frames,1,k);
ek=zeros(Frames,1,k);
es=zeros(Frames,1,k);



% Vectores usados para almacenar la suma punto a punto de las tres ROIS
% (diferencia) para cada uno de los atributos
sdv=zeros(Frames);
sdh=zeros(Frames);
sv=zeros(Frames);
sk=zeros(Frames);
ss=zeros(Frames);

css=1;% Contador
% contador=0;

while h<=Frames %Condición para recorrer todos los frames del Video
    if maximo>Frames
        maximo=Frames;
    end

    % Se realiza la separación de fondo y se obtienen los puntos de
    % interes (almacenados en cout), en la variable auxiliar se tiene la
    % región del frame (Despues de 5 frames) donde se hace puntos de
interes
    % para ser usada posteriormente, COR guarda las coordenadas del la
    % región despues de realizar MoG .
    [cout,auxiliar, G, COR]=mixog(Video,l,h);

    % Video de la separación de fondo usando Mezcla de Gaussianas (función
    % mixog)
    MOG(l:h)=G(l:h);


    % Se realizan los grupos o clusters de puntos de interes.
    [IDX,Cnt] = kmeans(cout,k);


    % En este ciclo se realizan las regiones de interes (ROI) y se marca en
    % la variable "auxiliar" con los puntos de interes para construir un
    % video de visualización
    for i=1:k
        x=cout(IDX==i,1);
        y=cout(IDX==i,2);
        ROIS(i,1)=min(cout(IDX==i,1));
        ROIS(i,2)=max(cout(IDX==i,1));
        ROIS(i,3)=min(cout(IDX==i,2));
        ROIS(i,4)=max(cout(IDX==i,2));

        for p=1:length(x)
```

```
            % Se pintan los puntos correspondientes a una ROI de un color
            % diferente, el numero de ROI va de 3 a 5
            if i==1
            auxiliar(x(p),y(p),1)=256;
            auxiliar(x(p),y(p),2)=0;
            auxiliar(x(p),y(p),3)=0;
            end
            if i==2
            auxiliar(x(p),y(p),1)=0;
            auxiliar(x(p),y(p),2)=256;
            auxiliar(x(p),y(p),3)=0;
            end
            if i==3
            auxiliar(x(p),y(p),1)=0;
            auxiliar(x(p),y(p),2)=0;
            auxiliar(x(p),y(p),3)=256;
            end
            if i==4
            auxiliar(x(p),y(p),1)=256;
            auxiliar(x(p),y(p),2)=256;
            auxiliar(x(p),y(p),3)=0;
            end
            if i==5
            auxiliar(x(p),y(p),1)=0;
            auxiliar(x(p),y(p),2)=256;
            auxiliar(x(p),y(p),3)=256;
            end
        end

    end

    % Se construye el video de los frames donde se realiza la detección y
    % marcación de puntos de interes.
    CSS(css)=im2frame(auxiliar);
    css=css+1;

    frame=read(Video,h);

    % En este ciclo se recorre cada uno de los frames de que componen la
    % ventana, se carga el frame donde se realizan puntos de interes.
    for j=minimo:maximo

            % Se carga cada frame de la ventana y se obtiene sus
            % dimensiones, en cada frame se realiza un seguimiento
            % (Tracking) de la región extraida despues de realizar MOG
            % usado la función SSDXCORR.
            frameе=read(Video,j);
            frame1=frameе;
            frame2=frame1;
            [fil,col]=size(frameе(:,:,1));

            % Función SSDXCORR realiza un Block Matching entre una región
            % de referencia y un frame selececcionado, arrojando como
            % resultado las coordenadas del centro de la región encontrada
            % y el ancho y alto de la región
            [S,c1,r1,w1,hg1] =
SSDXCORR(rgb2gray(frame1),rgb2gray(frame(COR(2):COR(2)+COR(4),COR(1):COR(1)
+COR(3),:)));


            c1=floor(c1-(w1/2));
            r1=floor(r1-(hg1/2));
```

```
            % Estas condiciones se realizan para garantizar que la región
            % este dentro del frame

            if c1<1
               c1=1;
            end
            if r1<1
               r1=1;
            end
            if c1>fil-w1
               c1=fil-w1;
            end
            if r1>col-hg1
               r1=col-hg1;
            end

            % En la región encontrada en el tracking anterior se realiza un
            % nuevo tracking de cada una de las regiones de interes
            % definidas anteriormente

            for m=1:k

            ROI=frame(ROIS(m,1):ROIS(m,2),ROIS(m,3):ROIS(m,4),:);

            [S,c,r,w,hg] = SSDXCORR(rgb2gray(frame1(c1:c1+w1-1,r1:r1+hg1-
1,:)),rgb2gray(ROI));

            C=c1+floor(c-(w/2));
            R=r1+floor(r-(hg/2));

            if C<1
               C=1;
            end
            if R<1
               R=1;
            end
            if C>fil-w
               C=fil-w;
            end
            if R>col-hg
               R=col-hg;
            end

            % Se recorta la región encontrada, y se procede a encontrar
            % caractersticas de desplazamiento (horizontal, vertical), tono
            % promedio, varianza, kurtosis y skewness y se almacenan estos
            % valores.

            Roi=frame1(C:C+w-1,R:R+hg-1,:);
            Roi=double(rgb2gray(Roi));
            VRoi=reshape(Roi,w*hg,1);
            varv=var(VRoi);
            kur=kurtosis(VRoi);
            ske=skewness(VRoi);

            Desplazamientov(j,3,m)=c; % Desplazamiento Vertical
            Desplazamientoh(j,3,m)=r; % Desplazamiento Horizontal
            Varianza(j,3,m)=varv; % Varianza
            Kurtosis(j,3,m)=kur; % Kurtosis
            Skewness(j,3,m)=ske; % Skewness
```

```
% Construcción de los videos para la evaluación del algoritmo
% de seguimiento (Tracking)
if j<=(10*M)

    set(handles.text4,'String',j);

    frame2(c1,r1:r1+hg1-1,1)=256;
    frame2(c1,r1:r1+hg1-1,2:3)=0;
    frame2(c1+w1-1,r1:r1+hg1-1,1)=256;
    frame2(c1+w1-1,r1:r1+hg1-1,2:3)=0;
    frame2(c1:c1+w1-1,r1,1)=256;
    frame2(c1:c1+w1-1,r1,2:3)=0;
    frame2(c1:c1+w1-1,r1+hg1-1,1)=256;
    frame2(c1:c1+w1-1,r1+hg1-1,2:3)=0;
    VT(j)=im2frame(frame2);

    if m==1
    framee(C,R:R+hg-1,1)=256;
    framee(C,R:R+hg-1,2:3)=0;
    framee(C+w-1,R:R+hg-1,1)=256;
    framee(C+w-1,R:R+hg-1,2:3)=0;
    framee(C:C+w-1,R,1)=256;
    framee(C:C+w-1,R,2:3)=0;
    framee(C:C+w-1,R+hg-1,1)=256;
    framee(C:C+w-1,R+hg-1,2:3)=0;
    end
    if m==2
    framee(C,R:R+hg-1,1)=0;
    framee(C,R:R+hg-1,2)=256;
    framee(C,R:R+hg-1,3)=0;

    framee(C+w-1,R:R+hg-1,1)=0;
    framee(C+w-1,R:R+hg-1,2)=256;
    framee(C+w-1,R:R+hg-1,3)=0;

    framee(C:C+w-1,R,1)=0;
    framee(C:C+w-1,R,2)=256;
    framee(C:C+w-1,R,3)=0;

    framee(C:C+w-1,R+hg-1,1)=0;
    framee(C:C+w-1,R+hg-1,2)=256;
    framee(C:C+w-1,R+hg-1,3)=0;
    end
    if m==3
    framee(C,R:R+hg-1,1)=0;
    framee(C,R:R+hg-1,2)=0;
    framee(C,R:R+hg-1,3)=256;

    framee(C+w-1,R:R+hg-1,1)=0;
    framee(C+w-1,R:R+hg-1,2)=0;
    framee(C+w-1,R:R+hg-1,3)=256;

    framee(C:C+w-1,R,1)=0;
    framee(C:C+w-1,R,2)=0;
    framee(C:C+w-1,R,3)=256;

    framee(C:C+w-1,R+hg-1,1)=0;
    framee(C:C+w-1,R+hg-1,2)=0;
    framee(C:C+w-1,R+hg-1,3)=256;
    end
```

```
    if m==4
    framee(C,R:R+hg-1,1)=256;
    framee(C,R:R+hg-1,2)=256;
    framee(C,R:R+hg-1,3)=0;

    framee(C+w-1,R:R+hg-1,1)=256;
    framee(C+w-1,R:R+hg-1,2)=256;
    framee(C+w-1,R:R+hg-1,3)=0;

    framee(C:C+w-1,R,1)=256;
    framee(C:C+w-1,R,2)=256;
    framee(C:C+w-1,R,3)=0;

    framee(C:C+w-1,R+hg-1,1)=256;
    framee(C:C+w-1,R+hg-1,2)=256;
    framee(C:C+w-1,R+hg-1,3)=0;
    end

    if m==5
    framee(C,R:R+hg-1,1)=0;
    framee(C,R:R+hg-1,2)=256;
    framee(C,R:R+hg-1,3)=256;

    framee(C+w-1,R:R+hg-1,1)=0;
    framee(C+w-1,R:R+hg-1,2)=256;
    framee(C+w-1,R:R+hg-1,3)=256;

    framee(C:C+w-1,R,1)=0;
    framee(C:C+w-1,R,2)=256;
    framee(C:C+w-1,R,3)=256;

    framee(C:C+w-1,R+hg-1,1)=0;
    framee(C:C+w-1,R+hg-1,2)=256;
    framee(C:C+w-1,R+hg-1,3)=256;
    end

    if m==k
        V(j)=im2frame(framee);
        axes(handles.axes1)
        imshow(frame1)
    end

end

if j>minimo+N

    set(handles.text4,'String',j);

    frame2(c1,r1:r1+hg1-1,1)=256;
    frame2(c1,r1:r1+hg1-1,2:3)=0;
    frame2(c1+w1-1,r1:r1+hg1-1,1)=256;
    frame2(c1+w1-1,r1:r1+hg1-1,2:3)=0;
    frame2(c1:c1+w1-1,r1,1)=256;
    frame2(c1:c1+w1-1,r1,2:3)=0;
    frame2(c1:c1+w1-1,r1+hg1-1,1)=256;
    frame2(c1:c1+w1-1,r1+hg1-1,2:3)=0;
    VT(j)=im2frame(frame2);

    Desplazamientov(j,1,m)=c;
    Desplazamientoh(j,1,m)=r;
    Varianza(j,1,m)=varv;
```

```
Kurtosis(j,1,m)=kur;
Skewness(j,1,m)=ske;

if m==1
framee(C,R:R+hg-1,1)=256;
framee(C,R:R+hg-1,2:3)=0;
framee(C+w-1,R:R+hg-1,1)=256;
framee(C+w-1,R:R+hg-1,2:3)=0;
framee(C:C+w-1,R,1)=256;
framee(C:C+w-1,R,2:3)=0;
framee(C:C+w-1,R+hg-1,1)=256;
framee(C:C+w-1,R+hg-1,2:3)=0;
end
if m==2
framee(C,R:R+hg-1,1)=0;
framee(C,R:R+hg-1,2)=256;
framee(C,R:R+hg-1,3)=0;

framee(C+w-1,R:R+hg-1,1)=0;
framee(C+w-1,R:R+hg-1,2)=256;
framee(C+w-1,R:R+hg-1,3)=0;

framee(C:C+w-1,R,1)=0;
framee(C:C+w-1,R,2)=256;
framee(C:C+w-1,R,3)=0;

framee(C:C+w-1,R+hg-1,1)=0;
framee(C:C+w-1,R+hg-1,2)=256;
framee(C:C+w-1,R+hg-1,3)=0;
end
if m==3
framee(C,R:R+hg-1,1)=0;
framee(C,R:R+hg-1,2)=0;
framee(C,R:R+hg-1,3)=256;

framee(C+w-1,R:R+hg-1,1)=0;
framee(C+w-1,R:R+hg-1,2)=0;
framee(C+w-1,R:R+hg-1,3)=256;

framee(C:C+w-1,R,1)=0;
framee(C:C+w-1,R,2)=0;
framee(C:C+w-1,R,3)=256;

framee(C:C+w-1,R+hg-1,1)=0;
framee(C:C+w-1,R+hg-1,2)=0;
framee(C:C+w-1,R+hg-1,3)=256;
end

if m==4
framee(C,R:R+hg-1,1)=256;
framee(C,R:R+hg-1,2)=256;
framee(C,R:R+hg-1,3)=0;

framee(C+w-1,R:R+hg-1,1)=256;
framee(C+w-1,R:R+hg-1,2)=256;
framee(C+w-1,R:R+hg-1,3)=0;

framee(C:C+w-1,R,1)=256;
framee(C:C+w-1,R,2)=256;
framee(C:C+w-1,R,3)=0;

framee(C:C+w-1,R+hg-1,1)=256;
```

```
                framee(C:C+w-1,R+hg-1,2)=256;
                framee(C:C+w-1,R+hg-1,3)=0;
                end

                if m==5
                framee(C,R:R+hg-1,1)=0;
                framee(C,R:R+hg-1,2)=256;
                framee(C,R:R+hg-1,3)=256;

                framee(C+w-1,R:R+hg-1,1)=0;
                framee(C+w-1,R:R+hg-1,2)=256;
                framee(C+w-1,R:R+hg-1,3)=256;

                framee(C:C+w-1,R,1)=0;
                framee(C:C+w-1,R,2)=256;
                framee(C:C+w-1,R,3)=256;

                framee(C:C+w-1,R+hg-1,1)=0;
                framee(C:C+w-1,R+hg-1,2)=256;
                framee(C:C+w-1,R+hg-1,3)=256;
                end

                if m==k
                    V(j)=im2frame(framee);
                    axes(handles.axes1)
                    imshow(frame1)
                end

            end

            % En estas lineas de código se realiza el error cuadratico
            % medio punto a punto entre la señal que guarda los datos de
            % seguimiento y la señal obtenida de la predicción realizada

            edv(j,1,m)=(Desplazamientov(j,1,m)-Desplazamientov(j,2,m)).^2;
            edh(j,1,m)=(Desplazamientoh(j,1,m)-Desplazamientoh(j,2,m)).^2;
            ev(j,1,m)=(Varianza(j,1,m)-Varianza(j,2,m)).^2;
            ek(j,1,m)=(Kurtosis(j,1,m)-Kurtosis(j,2,m)).^2;
            es(j,1,m)=(Skewness(j,1,m)-Skewness(j,2,m)).^2;

            % Se realiza la predicción de un punto, con base en los N datos
            % anteriores almacenados, se usa un predictor lineal de orden 2
            if j>=minimo+N && j <maximo

                coef_d1=lpc(Desplazamientov(j-N:j,3,m),Orden);
                coef_d2=lpc(Desplazamientoh(j-N:j,3,m),Orden);
                coef_v=lpc(Varianza(j-N:j,3,m),Orden);
                coef_s=lpc(Skewness(j-N:j,3,m),Orden);
                coef_k=lpc(Kurtosis(j-N:j,3,m),Orden);

                est_d1= filter([0 -coef_d1(2:end)],1,Desplazamientov(j-
N:j,3,m));
                est_d2= filter([0 -coef_d2(2:end)],1,Desplazamientoh(j-
N:j,3,m));
                est_v= filter([0 -coef_v(2:end)],1,Varianza(j-N:j,3,m));
                est_s= filter([0 -coef_s(2:end)],1,Skewness(j-N:j,3,m));
                est_k= filter([0 -coef_k(2:end)],1,Kurtosis(j-N:j,3,m));

                Desplazamientov(j+1,2,m)=est_d1(end);
                Desplazamientoh(j+1,2,m)=est_d2(end);
                Varianza(j+1,2,m)=est_v(end);
```

```
            Skewness(j+1,2,m)=est_s(end);
            Kurtosis(j+1,2,m)=est_k(end);

        end
        end

        % Con los datos de las ROI se realiza la suma de estos para
        % cada atributo y se almacena en vectores suma.

        sdv(j)=sum(edv(j,1,:));
        axes(handles.axes2)
        hold on
        plot(j,sdv(j),'.');

        sdh(j)=sum(edh(j,1,:));
        axes(handles.axes3)
        hold on
        plot(j,sdh(j),'.');

        sv(j)=sum(ev(j,1,:));
        axes(handles.axes4)
        hold on
        plot(j,sv(j),'.');

        sk(j)=sum(ek(j,1,:));
        axes(handles.axes5)
        hold on
        plot(j,sk(j),'.');

        ss(j)=sum(es(j,1,:));
        axes(handles.axes6)
        hold on
        plot(j,ss(j),'.');


        % Se realiza la detección de anomalias con los vectores suma y
        % un maximo que se actualiza cada N frames, si el dato j supera
        % el maximo se considera anomalia, esto se realiza para cada
        % atributo.
        if j>minimo+N && j>(20*M)
            p=num2str(j);

            contador=0;
%             j
%             display('Hola')
%
            if sdv(j)>f*pdv
%                 display(horzcat('Anomalía Frame ',p,' Dv'))
                contador=contador+1;
            end
            if sdh(j)>f*pdh
%                 display(horzcat('Anomalía Frame ',p,' Dh'))
                contador=contador+1;
            end
            if sv(j)>f*pv
%                 display(horzcat('Anomalía Frame ',p,' V'))
                contador=contador+1;
            end
            if sk(j)>f*pk
%                 display(horzcat('Anomalía Frame ',p,' K'))
                contador=contador+1;
            end
```

```
                if ss(j)>f*ps
%                     display(horzcat('Anomalía Frame ',p,' S'))
                    contador=contador+1;
                end
%                 contador
                if contador >= 3
                    set(handles.text18,'String','Anomalía');
                else
                    set(handles.text18,'String','Normal');
                end



            end

            % Calculo del maximo, este se realiza para cada atributo y se
            % hace cuando se está en el último frame de la ventana, se
            % realiza con el valor máximo de los útimos N frames

            if j==maximo
                pdv=max(sdv(j-N:j));
                pdh=max(sdh(j-N:j));
                pv=max(sv(j-N:j));
                pk=max(sk(j-N:j));
                ps=max(ss(j-N:j));
            end

    end

    % Se actualiza la ventana y los valores para realizar MOG
    l=l+Tasa/3*M;
    h=h+Tasa/3*M;
    minimo=minimo+Tasa/3*M;
    maximo=maximo+Tasa/3*M;
end




function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit 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
```

```
function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit 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



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a
double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit 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



function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
```

```
%        str2double(get(hObject,'String')) returns contents of edit4 as a
double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit 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
```

9. LISTA DE FIGURAS

Fig. 1. Diagrama de bloques del algoritmo desarrollado.

Fig. 2. Descripción de proceso de construcción de perfiles de comportamiento. t_1: Estimación de fondo. t_2: Detección de objeto de interés. t_3: Eliminación de fondo. t_4: Detección de puntos de interés. t_5: Agrupamiento de puntos de interés en ROI's para posterior seguimiento.

Fig. 3. Sustracción de fondo para el video C6.avi, se indican los *frames* objeto de este proceso. En puntos blancos se muestra lo que no pertenece al fondo: (a) Sustracción de fondo para el *frame* 1, (b) Sustracción de fondo para el *frame* 2, (c) Sustracción de fondo para el *frame* 3, (d) Sustracción de fondo para el *frame* 4, (e) Sustracción de fondo para el *frame* 5.

Fig. 4. Sustracción de fondo para el video C19.avi, se indican los *frames* objeto de este proceso. En puntos blancos se muestra lo que no pertenece al fondo: (a) Sustracción de fondo para el *frame* 56, (b) Sustracción de fondo para el *frame* 57, (c) Sustracción de fondo para el *frame* 58, (d) Sustracción de fondo para el *frame* 59, (e) Sustracción de fondo para el *frame* 60.

Fig. 5. *Clusters* o agrupación de puntos de interés para el proceso de sustracción de fondo del video C6.avi realizado en el *frame* 5. Cada agrupación corresponde una región de interés.

Fig. 6. *Clusters* o agrupación de puntos de interés para el proceso de sustracción de fondo del video C19.avi realizado en el *frame* 60. Cada agrupación corresponde una región de interés.

Fig. 7. *Pre-tracking* realizado en una ventana de 10 *frames* para el video C6.avi: (a) *Pre-tracking frame* 1, (b) *Pre-tracking frame* 2, (c) *Pre-tracking frame* 3, (d) *Pre-tracking frame* 4, (e) *Pre-tracking frame* 5, (f) *Pre-tracking frame 6*, (g) *Pre-tracking frame* 7, (h) *Pre-tracking frame* 8, (i) *Pre-tracking frame* 9, (j) *Pre-tracking frame* 10.

Fig. 8. *Tracking* realizado a las regiones de interés en una ventana de 10 *frames* para el video C6.avi: (a) Regiones de interés *frame* 1, (b) Regiones de interés *frame* 2, (c) Regiones de interés *frame* 3, (d) Regiones de interés *frame* 4, (e) Regiones de interés *frame* 5, (f) Regiones de interés *frame* 6, (g) Regiones de interés *frame* 7, (h) Regiones de interés *frame* 8, (i) Regiones de interés *frame* 9, (j) Regiones de interés *frame* 10.

Fig. 9. *Pre-tracking* realizado en una ventana de 10 *frames* para el video C19.avi: (a) *Pre-tracking frame* 56, (b) *Pre-tracking frame* 57, (c) *Pre-tracking frame* 58, (d) *Pre-tracking frame* 59, (e) *Pre-tracking frame* 60, (f) *Pre-tracking frame* 61, (g) *Pre-tracking frame* 62, (h) *Pre-tracking frame* 63, (i) *Pre-tracking frame* 64, (j) *Pre-tracking frame* 65.

Fig. 10. *Tracking* realizado a las regiones de interés en una ventana de 10 *frames* para el video C19.avi: (a) Regiones de interés *frame* 56, (b) Regiones de interés *frame* 57, (c) Regiones de interés *frame* 58, (d) Regiones de interés *frame* 59, (e) Regiones de interés *frame* 60, (f) Regiones de interés *frame* 61, (g) Regiones de interés *frame* 62, (h) Regiones de interés *frame* 63, (i) Regiones de interés *frame* 64, (j) Regiones de interés *frame* 65.

Fig. 11. Filtro usado para implementar un predictor lineal hacia delante utilizando la segunda forma traspuesta.

Fig. 12. Datos de Desplazamiento Horizontal ROI 1: (a) Datos obtenidos de seguimiento, (b) Datos obtenidos de predicción, (c) Señal de error.

Fig. 13. Datos de Desplazamiento Horizontal: (a) Señal de error ROI 1, (b) Señal de error ROI 2, (c) Señal de error ROI 3, (d) Suma de atributos.

Fig. 14. Circuito de decisión para detección de anomalías, recibe como entradas la suma de todos los atributos y arroja si el *frame* posee o no anomalía.

Fig. 15 Resultados correspondientes al atributo Desplazamiento Vertical para el video C6.avi: (a) Resultado obtenido del tracking para la ROI 1, (b) Resultado obtenido de la predicción para la ROI 1, (c) Resultado obtenido del tracking para la ROI 2, (d) Resultado obtenido de la predicción para la ROI 2, (e) Resultado obtenido del tracking para la ROI 3, (e) Resultado obtenido de la predicción para la ROI 3.

Fig. 16 Señales de error para el atributo Desplazamiento Vertical en las diferentes regiones de interés: (a) Señal de error de Desplazamiento Vertical para ROI 1, (b) Señal de error de Desplazamiento Vertical para ROI 2, (c) Señal de error de Desplazamiento Vertical para ROI 3.

Fig. 17 Resultados correspondientes al atributo Desplazamiento Horizontal para el video C6.avi: (a) Resultado obtenido del tracking para la ROI 1, (b) Resultado obtenido de la predicción para la ROI 1, (c) Resultado obtenido del tracking para la ROI 2, (d) Resultado obtenido de la predicción para la ROI 2, (e) Resultado obtenido del tracking para la ROI 3, (e) Resultado obtenido de la predicción para la ROI 3.

Fig. 18 Señales de error para el atributo Desplazamiento Horizontal en las diferentes regiones de interés: (a) Señal de error de Desplazamiento Horizontal para ROI 1, (b) Señal de error de Desplazamiento Horizontal para ROI 2, (c) Señal de error de Desplazamiento Horizontal para ROI 3.

Fig. 19 Resultados correspondientes al atributo Varianza para el video C6.avi: (a) Resultado obtenido del tracking para la ROI 1, (b) Resultado obtenido de la predicción para la ROI 1, (c) Resultado obtenido del tracking para la ROI 2, (d) Resultado obtenido de la predicción para la ROI 2, (e) Resultado obtenido del tracking para la ROI 3, (e) Resultado obtenido de la predicción para la ROI 3.

Fig. 20 Señales de error para el atributo Varianza en las diferentes regiones de interés: (a) Señal de error de Varianza para ROI 1, (b) Señal de error de Varianza para ROI 2, (c) Señal de error de Varianza para ROI 3.

Fig. 21 *Tracking* realizado a las regiones de interés en una ventana de 10 *frames* para el video C6.avi: (a) Regiones de interés *frame* 41, (b) Regiones de interés *frame* 42, (c) Regiones de interés *frame* 43, (d) Regiones de interés *frame* 44, (e) Regiones de interés *frame* 45, (f) Regiones de interés *frame* 46, (g) Regiones de interés *frame* 47, (h) Regiones de interés *frame* 48, (i) Regiones de interés *frame* 49, (j) Regiones de interés *frame* 50.

Fig. 22 Resultados correspondientes al atributo Asimetría o Skewness para el video C6.avi: (a) Resultado obtenido del tracking para la ROI 1, (b) Resultado obtenido de la predicción para la ROI 1, (c) Resultado obtenido del tracking para la ROI 2, (d) Resultado obtenido de la predicción para la ROI

2, (e) Resultado obtenido del tracking para la ROI 3, (e) Resultado obtenido de la predicción para la ROI 3.

Fig. 23 Señales de error para el atributo Asimetría o Skewness en las diferentes regiones de interés: (a) Señal de error de Asimetría para ROI 1, (b) Señal de error de Asimetría para ROI 2, (c) Señal de error de Asimetría para ROI 3.

Fig. 24 Resultados correspondientes al atributo *Kurtosis* para el video C6.avi: (a) Resultado obtenido del tracking para la ROI 1, (b) Resultado obtenido de la predicción para la ROI 1, (c) Resultado obtenido del tracking para la ROI 2, (d) Resultado obtenido de la predicción para la ROI 2, (e) Resultado obtenido del tracking para la ROI 3, (e) Resultado obtenido de la predicción para la ROI 3.

Fig. 25 Señales de error para el atributo *Kurtosis* en las diferentes regiones de interés: (a) Señal de error de *Kurtosis* para ROI 1, (b) Señal de error de *Kurtosis* para ROI 2, (c) Señal de error de *Kurtosis* para ROI 3.

Fig. 26 Señales de suma de error para cada uno de los atributos: (a) Señal de suma de Desplazamiento horizontal, (b) Señal de suma de Desplazamiento vertical, (c) Señal de suma de Varianza, (d) Señal de suma de Asimetría, (e) Señal de suma de *Kurtosis*.

Fig. 27 Curva ROC Nombre video: C6.avi: Tasa de *frames*: 30 *frames*/s, Longitud de ventana: 10 *frames*, número de ROIs: 3, orden del predictor: 2, factor de umbral: 1,0.

Fig. 28 Ejemplo de Análisis en grupos de 3, 5 y 7 *frames*: O: *frame* originalmente identificado con anomalía, N: *frame* originalmente identificado como normal y A: *frame* catalogado con anomalía luego de realizar el análisis por grupos.

Fig. 29 (a) Resultado entregado por el programa sin hacer análisis por grupo de *frames* con factor de escala del umbral de 1. (b) Distinción entre los videos realizados en exteriores e interiores.

Fig. 30 (a) Resultado entregado por el programa haciendo análisis por grupo de 5 *frames* con factor de escala del umbral de 1. (b) Distinción entre los videos realizados en exteriores e interiores.

Fig. 31 (a) Resultado entregado por el programa haciendo análisis por grupo de 5 *frames* con factor de escala del umbral de 1. (b) Distinción entre los videos realizados en exteriores e interiores en el videoC6.avi.

Fig. 32 (a) Resultado entregado por el programa haciendo análisis por grupo de 7 *frames* con factor de escala del umbral de 1. (b) Distinción entre los videos realizados en exteriores e interiores.

Fig. 33 (a) Resultado original entregado por el programa con un factor de escala de umbral de 1.2. (b) Distinción entre los videos realizados en exteriores e interiores.

Fig. 34 (a) Resultado entregado por el programa haciendo análisis por grupos de 3 *frames* con un factor de escala de umbral de 1.2. (b) Distinción entre los videos realizados en exteriores e interiores para el análisis de grupos de 3 *frames* y factor de escala de 1.2. (c) Resultado entregado por el programa haciendo análisis por grupos de 5 *frames* con un factor de escala de umbral de 1.2. (d) Distinción entre los videos realizados en exteriores e interiores para el análisis de grupos de 5 *frames* y factor de escala de 1.2. (f) Resultado entregado por el programa haciendo análisis por grupos de 7 *frames* con un factor de escala de umbral de 1.2. (g) Distinción entre los videos realizados en exteriores e interiores de 7 *frames* y factor de escala de 1.2.

Fig. 35 (a) Resultado original entregado por el programa con un factor de escala de umbral de 1.4. (b) Distinción entre los videos realizados en exteriores e interiores.

Fig. 36 (a) Resultado original entregado por el programa haciendo análisis por grupos de 3 *frames* con un factor de escala de umbral de 1.4. (b) Distinción entre los videos realizados en exteriores e

interiores para el análisis de grupos de 3 *frames* y factor de escala de 1.4. (c) Resultado original entregado por el programa haciendo análisis por grupos de 5 *frames* con un factor de escala de umbral de 1.4. (d) Distinción entre los videos realizados en exteriores e interiores para el análisis de grupos de 5 *frames* y factor de escala de 1.4. (e) Resultado entregado por el programa haciendo análisis por grupos de 7 *frames* con un factor de escala de umbral de 1.4. (f) Distinción entre los videos realizados en exteriores e interiores para el análisis de grupos de 7 *frames* y factor de escala de 1.4.

Fig. 37 (a) Resultado original entregado por el programa con un factor de escala de umbral de 1.6. (b) Distinción entre los videos realizados en exteriores e interiores.

Fig. 38 (a) Resultado original entregado por el programa haciendo análisis por grupos de 3 *frames* con un factor de escala de umbral de 1.6. (b) Distinción entre los videos realizados en exteriores e interiores para el análisis de grupos de 3 *frames* y factor de escala de 1.6. (c) Resultado original entregado por el programa haciendo análisis por grupos de 5 *frames* con un factor de escala de umbral de 1.6. (d) Distinción entre los videos realizados en exteriores e interiores para el análisis de grupos de 5 *frames* y factor de escala de 1.6. (e) Resultado entregado por el programa haciendo análisis por grupos de 7 *frames* con un factor de escala de umbral de 1.6. (f) Distinción entre los videos realizados en exteriores e interiores para el análisis de grupos de 7 *frames* y factor de escala de 1.6.