UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

Grado en Ing. Sist. de Telecom., Sonido e Imagen

“Mejora del software de detección de la continuidad en la UPVTV: Detección automática del logotipo en canales de TV.”

TRABAJO FINAL DE GRADO

Autor/a:
Gianna Lava Werner

Tutor/a:
Ignacio Bosch Roig

GANDIA, 2013

Índice

TFG: Mejora del software de detección de la continuidad en la UPVTV: Detección del logotipo

Resumen:

En este trabajo se han introducido mejoras importantes en el software de detección de continuidad en la UPVTV, desarrollado previamente en el TFC [1], de forma que la detección del logotipo de la imagen se realice de forma automática en lugar de tener que realizarse manualmente, adaptándose el procesado de búsqueda del logotipo para que lo detecte de forma independiente a la cadena emisora de TV y no únicamente para la UPVTV. Para conseguirlo se ha realizado un procesado temporal y espacial de las imágenes, en el que se analizan las imágenes píxel a píxel, y en función de la información de éstos se extraen las características a partir de las cuales se determinarán las regiones de interés de la imagen, a fin de detectar la ubicación del logotipo.

También se ha implementado la configuración mediante un archivo de parámetros tipo texto, en el cual se introducen los parámetros para ser leídos por la aplicación, y la generación de ficheros de registro del funcionamiento y alarmas generadas. De esta forma, toda la actividad de la aplicación quedará registrada, desde la fecha y hora en que se puso en funcionamiento hasta las diversas alarmas generadas por la misma.

Palabras Clave : Detección automática, procesado de imagen, logotipo emisoras TV, varianza.

Abstract:

In this work, important improvements were made in the UPVTV continuity detection software, previously developed in TFC [1], so the detection of the TV logo in the image is done automatically instead of having to be done manually, adapting the logo search processing for detecting it regardless of the broadcast network, not only for UPVTV. To achieve it, there has been developed a spatial and temporal processing of TV images, in which images are analyzed pixel by pixel, and based on the information of the pixels, drawn the characteristics from which the regions of interest of the image will be defined in order to detect the location of the logo on it.
It has also been implemented the software's configuration through a text file in which parameters are entered so as to be read by the application, and the generation of log files of execution and generated alarms. This way, the whole application activity will be recorded, including the date and time that it was put into operation and the alarms it generated.

Keywords: Automatic detection, image processing, TV logo, variance.

Introducción

En este trabajo se han realizado importantes mejoras sobre el ya existente software de detección de la continuidad de la UPVTV, *MantConTV*, desarrollado en el Proyecto Final de Carrera titulado "*Implementación de sistema de captura y detección de imágenes para el mantenimiento de la continuidad de la señal de TV en la UPV-TV* " [1].

Las modificaciones han sido realizadas sobre el mismo software, por lo cual se ha mantenido su estructura, incluida la interfaz gráfica, a la que se le han realizado los cambios pertinentes para adecuarla a las nuevas funcionalidades.

La aplicación se divide en tres etapas, la primera etapa es la encargada de la adquisición de las imágenes provenientes de la señal de vídeo, en tiempo real utilizando una tarjeta capturadora o directamente de un fichero de video almacenado tipo avi. En la segunda etapa se realiza el procesado de las imágenes adquiridas, mediante unos algoritmos específicos que detectan de forma precisa los problemas de continuidad que pueda presentar la secuencia de vídeo. La última etapa es la de presentación de la información y los resultados obtenidos, donde, de forma visual y sencilla, el usuario puede establecer los parámetros de procesado, comenzar la captura de imágenes y visualizar tanto éstas como las que se generan tras su procesado, de forma que puede comprobar el funcionamiento de los diferentes algoritmos y ver los resultados de forma inmediata.

El software previamente desarrollado aplica diferentes procesados a la señal de TV a fin de detectar problemas de continuidad tales como la imagen parada, el fundido a negro y la ausencia de la mosca. En este caso, las mejoras incorporadas se centran en el procesado de detección de la presencia de la mosca, desde la localización del logotipo en la imagen, hasta su identificación en la señal de vídeo analizada.

El algoritmo desarrollado realiza la detección automática del logotipo de la cadena emisora en la imagen, independientemente de la procedencia de las imágenes que se estén analizando. A partir del resultado obtenido, se realiza un procesado para la identificación del logotipo o mosca (como se suele denominar en TV) en la imagen, que permite determinar la presencia o ausencia del logotipo en la misma.

Tras realizar dichos cambios se obtiene una aplicación que presenta una mayor versatilidad, ya que permite analizar videos de cualquier cadena emisora. Además, se genera en tiempo real un fichero de registro diario, en el que se recoge la información obtenida durante el análisis de los videos, que incluye un seguimiento de los procesos llevados a cabo, alarmas generadas y parámetros de funcionamiento.

Objetivos

El objetivo principal del trabajo es el de mejorar el software previamente desarrollado de detección de continuidad de la UPVTV para que la detección del logotipo se realice automáticamente en lugar de tener que hacerse manualmente.

Como objetivos secundarios se establecen:

- Modificación del procesado de detección de la mosca, para que éste sea capaz de detectar el logotipo de cualquier cadena emisora.
- Configuración de parámetros por fichero de texto para que el software pueda ejecutarse remotamente sin necesidad de utilizar la interface gráfica.
- Generación de ficheros de registro del funcionamiento y alarmas generadas.
- Modificación de la interfaz gráfica para adaptarla a los nuevos procesados.

CAPITULO 1: Esquema general de la aplicación

La aplicación se estructura en tres etapas diferentes: Adquisición, Procesado y Presentación. A continuación se explicarán, de forma breve y sencilla, cuales son las tareas desarrolladas en cada una de ellas y sus principales características.

1.1 Etapa de adquisición

El objetivo principal de la aplicación es el análisis de la señal de vídeo que se está emitiendo, por lo cual la adquisición de las imágenes ha de hacerse en tiempo real, y para ello es necesario tener correctamente instalada una capturadora de vídeo.

El hardware que se ha utilizado para llevar a cabo la adquisición es la capturadora externa Sveon STV-40 – aunque la aplicación funciona con cualquier dispositivo de captura, incluida la webcam de un ordenador portátil –. Esta capturadora se conecta al ordenador mediante un puerto USB (Universal Serial Bus) y sus especificaciones técnicas con las siguientes:

- **Interfaz:** USB 2.0
- **Entrada de vídeo:**
  - Vídeo compuesto: jack RCA
  - S-Vídeo: 4 pin mini din
- **Entrada de audio:** RCA L/R estéreo
- **Resolución de vídeo:**
  - NTSC: 720x480 30fps
  - PAL: 720x576 25fps
- **Tamaño dispositivo:** 69,4x28,4x9,7 mm

Además de la captura desde un dispositivo externo, utilizamos también como método de adquisición de imágenes la lectura de un archivo de vídeo de extensión avi.

1.2 Etapa de procesado

En esta etapa se realizan, sobre las imágenes adquiridas anteriormente, los procesados para la detección de los problemas de continuidad en la señal de video en emisión.

Estos problemas son la imagen parada, el fundido a negro y la ausencia de la mosca, y sus características se describen a continuación.

- **Imagen parada:** se dice que la imagen está parada cuando no se detectan cambios en el contenido de la misma conforme pasa el tiempo, ya que un vídeo es, en esencia, una sucesión de imágenes que componen una animación. La continua sucesión de estas

imágenes –a las que llamamos frames- producen a la vista la sensación de movimiento debido a las diferencias entre ellas. Es por ello que al no haber diferencias entre los frames de un vídeo decimos que la imagen está parada.

- **Fundido a negro:** el fundido no es un error de continuidad en sí, sino un efecto que se utiliza para suavizar las transiciones - o cambios - en los vídeos, como puede ser un cambio de escena o en el caso de la emisión televisiva, un cambio de vídeo. El efecto de fundido consta en ir oscureciendo la imagen paulatinamente hasta que quede completamente negra. Es normal que cuando se ha llegado al negro total, se mantenga la imagen en ese estado durante algunos instantes, - tiempo de fundido - , que es cuando se puede confundir un fundido con el problema de la imagen parada. En estos casos, al no apreciarse diferencias entre los frames, se pierde el efecto de movimiento, y es por ello que consideraremos los fundidos como un caso más de imagen parada en el cual, además de no apreciarse movimiento, los píxeles de la imagen presentan el mismo nivel.

- **Ausencia de la mosca:** la mosca es el logotipo de la cadena emisora, y se muestra para que el espectador sepa la procedencia del material audiovisual que está mirando. La mosca es una imagen separada del video que es insertada sobre el mismo al final del montaje, y sus propiedades no cambian; mantiene la forma, ubicación y tamaño -para la misma resolución de video-. El problema de ausencia de la mosca se da cuando el logotipo desaparece de la imagen durante la emisión de contenidos (excluyendo las publicidades).

Los algoritmos de procesado analizan las imágenes adquiridas en busca de dichos problemas.

1.3 Etapa de presentación

Si bien esta etapa no es imprescindible para el adecuado funcionamiento de la aplicación, es conveniente contar con ella ya que es la forma de comprobar de manera inmediata el correcto funcionamiento de los algoritmos de procesado. Además, se ha aprovechado esta etapa para dar al usuario la opción de cambiar ciertos parámetros y configuraciones como pueden ser la dirección de correo electrónico a la que se han de enviar las alertas de funcionamiento, el procesado a llevar a cabo sin necesidad de reiniciar la aplicación.

La interfaz gráfica ha sido desarrollada en GUIDE - Graphical User Interface Development Environment- , una aplicación de MatLab que permite el diseño e implementación de interfaces de usuario de manera sencilla y cómoda.

A continuación se muestra la pantalla principal de la aplicación.

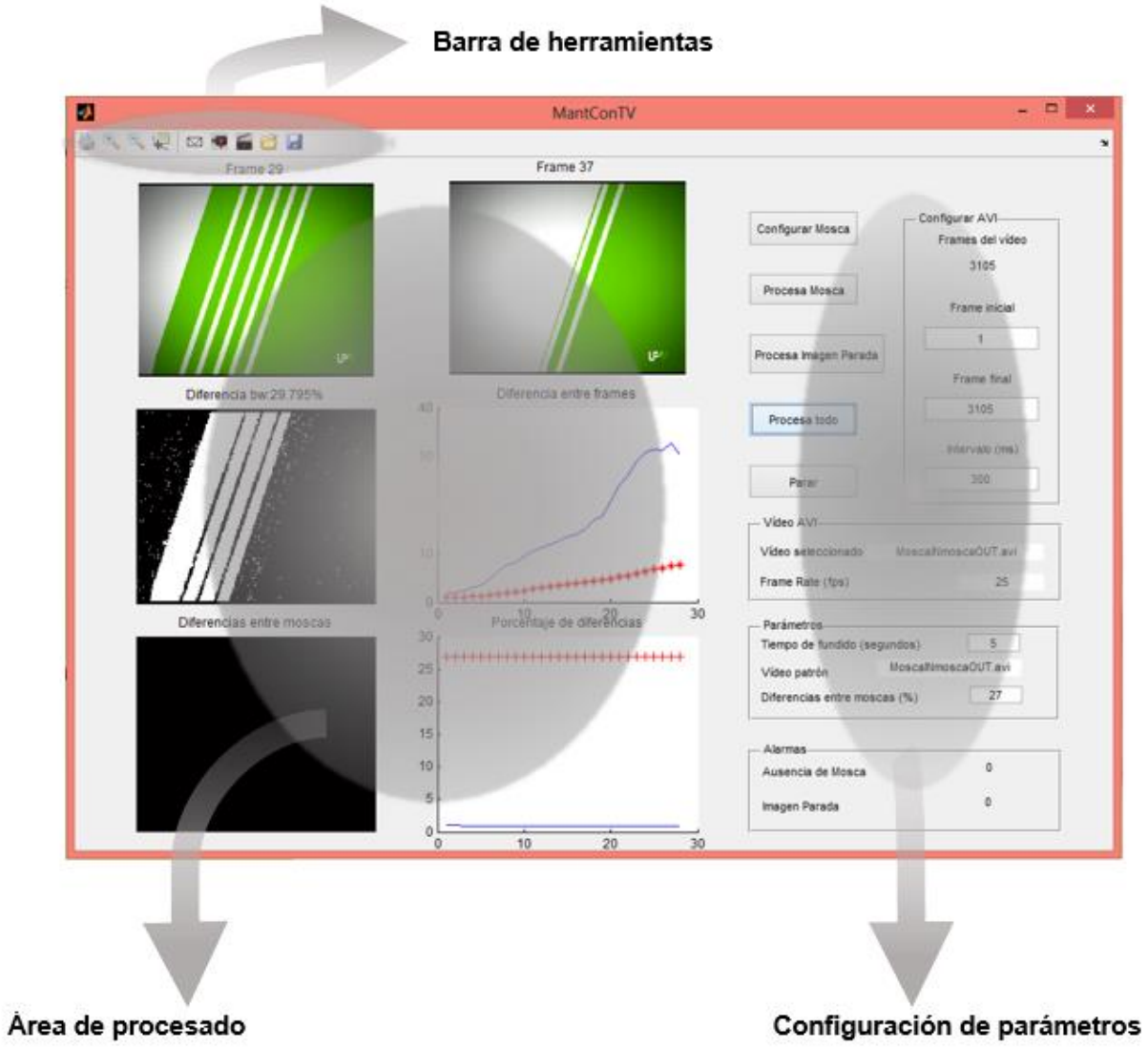

Figura 1.3.1: Interfaz gráfica de la aplicación

En la Figura 1.3.1 se puede ver el aspecto de la interfaz gráfica. Se pueden diferenciar las tres zonas que la constituyen; Barra de Herramientas, Área de Configuración de Parámetros y Área de Procesado.

Los elementos que componen tanto la zona de procesado como la de configuración de parámetros irán variando en función del proceso realizado. El funcionamiento de la interfaz gráfica se explicará más adelante.

CAPITULO 2: Algoritmos de procesado

En esta etapa se lleva a cabo el procesado de las imágenes adquiridas. Dicho procesado se realiza píxel a píxel, considerando, no sólo la imagen del instante actual, sino imágenes de instantes de tiempo anteriores.

Cuando hablamos de problemas de continuidad nos referimos, por un lado, a la ausencia de movimiento en la señal de vídeo, y por otro, a la ausencia de la mosca de la cadena emisora. Es por ello que se ha dividido el algoritmo de procesado en dos partes, una dedicada a la detección de la imagen parada -incluyendo la comprobación de que no se trate de un fundido a negro-, y la otra encargada de la búsqueda de la mosca en la imagen, ya esté ésta parada o en movimiento.

Al final de cada uno de los procesados y en el caso de que se hayan detectado problemas de continuidad, se enviará un e-mail avisando del problema detectado y si el mismo se solucionara, también se notificará por correo electrónico.

2.1 Detección de la imagen parada

El algoritmo de detección de la imagen parada no ha sido modificado, por lo que solamente se realizara una breve descripción de su funcionamiento.

La imagen parada no es más que la ausencia de diferencias entre las imágenes consecutivas de una serie, que proporcionan la sensación de movimiento que caracteriza a los videos. Este algoritmo se basa en la búsqueda de diferencias entre los sucesivos frames del video, y en función de dichas diferencias, se determinará la existencia del problema de continuidad buscado.

Al ser las diferencias entre frames la base del procesado, se define la imagen diferencia, que no es más que la resta absoluta entre el frame actual - el que se está procesando - y el anterior. De esta forma, obtenemos una imagen que presenta toda la información que necesitamos, sobre la cual se realizarán el resto de operaciones.

El siguiente paso es binarizar la imagen diferencia, de forma que se pasa de tener una imagen tridimensional - RGB - donde cada píxel está representado por sus correspondientes valores en R, G y B, a una imagen binaria en la cual cada píxel puede tomar los valores 0 o 1. Esta conversión se realiza mediante una umbralización, de modo que se establece un valor umbral a partir del cual los píxeles cuyo valor esté por encima del umbral serán blancos y los que estén por debajo serán negros. De esta forma, se elige el umbral como el nivel mínimo de diferencias a detectar, así obtendremos una imagen bidimensional que nos muestra las diferencias que se han encontrado entre una imagen y la siguiente, de forma que las partes de la imagen que se han mantenido iguales aparecen en negro y los cambios en blanco.

A partir de la imagen binaria, se calcula el porcentaje de la imagen que ha cambiado, y será este valor el que se utilizará para determinar si la imagen está parada.

Una vez se ha calculado el porcentaje de imagen que cambia, se establece un umbral a partir del cual se pueda decir con seguridad si la imagen se ha parado. Dicho umbral no puede ser elegido de forma subjetiva, ya que cada video tiene sus propias características, por lo tanto, se calculará para cada imagen un valor umbral diferente en función del valor que presenten las *n* imágenes anteriores. Si el valor de la imagen actual es mayor al valor del umbral calculado, se

entiende que la imagen está en movimiento, en cambio si dicho valor es menor al determinado como umbral, se asume que la imagen se ha parado.

En los casos en que se determine que la imagen está parada, antes de confirmarlo hemos de asegurarnos de que no se trata de un fundido. Esto se consigue de forma sencilla, ya que en los fundidos la imagen permanece parada durante algunos segundos antes de recuperar el movimiento, por lo tanto, si una vez transcurrido ampliamente dicho tiempo la imagen sigue sin presentar cambios, se puede confirmar que no se trata de un fundido. Como el tiempo de fundido se establece durante la edición del vídeo y puede variar de un vídeo a otro, se puede especificar como parámetro de procesado, de forma que no haya errores al respecto.

Por último, una vez confirmada la imagen parada, se envía una alarma mediante correo electrónico para notificar de la detección del problema. Asimismo, de reanudarse la emisión correctamente, también se notificará por el mismo medio.

2.2 Búsqueda de la mosca

El objetivo principal de este trabajo es el de mejorar los algoritmos de detección e identificación de la mosca para que la aplicación sea más versátil y efectiva. En particular, se busca un algoritmo que sea capaz de detectar automáticamente la posición de la mosca dentro de la imagen y además, que detecte la presencia o ausencia de la mosca de cualquier cadena emisora.

Para conseguirlo, se decidió realizar dos algoritmos por separado. Al igual que se hizo en el proyecto anterior [1], se realizaron dos procesados diferentes, uno para la **localización de la mosca en una imagen patrón**, y otro para la **detección de la presencia o ausencia de la mosca en el vídeo analizado**.

Dado que para realizar la búsqueda de la mosca solamente nos interesa la información presente en su zona correspondiente en la imagen, el primer paso a realizar es filtrar la imagen proveniente de la etapa de adquisición, de modo que nos quedemos únicamente con la información útil. Para ello utilizaremos una máscara en la cual toda la imagen será negra, a excepción de la zona del logotipo que será blanca, así, al multiplicar la máscara por la imagen a analizar, nos quedaremos con la zona de interés y desecharemos el resto.

El aspecto más importante de la máscara es que debe adaptarse a la mosca que vayamos a buscar, y como queremos que el algoritmo de búsqueda sea capaz de identificar el logotipo de cualquier cadena emisora, el algoritmo de detección de la mosca debe ser capaz de crear una máscara que se adaptada a cualquier mosca, independientemente de sus características. Es por ello que la máscara ha de obtenerse automáticamente, ya que es imposible establecer una máscara única que se ajuste a cualquier mosca.

Una vez tenemos la máscara, hemos de realizar un procesado de identificación de la mosca que sea capaz de determinar, mediante la información obtenida tras el filtrado, si la mosca está presente en la imagen que se está analizando.

En los siguientes apartados se explicará el funcionamiento de cada uno de los algoritmos desarrollados.

2.2.1 Algoritmo de obtención de la máscara

Para poder obtener la máscara, lo primero que necesitamos saber es qué mosca vamos a buscar, para lo cual nos haría falta una imagen que la contuviera. En este caso, como la detección se hará automáticamente, una sola imagen no es suficiente, por lo cual utilizaremos un vídeo, ya que nos proporciona más información.

Como el objetivo de la máscara es quedarnos sólo con la información útil de la imagen, aunque no sepamos que máscara en concreto estamos buscando, podemos saber a priori cuales son las regiones de interés, ya que en la gran mayoría de los casos, la mosca se encuentra en el margen derecho de la imagen sobre una esquina, lo cual nos deja dos opciones: la esquina superior o inferior derecha de la imagen.

Con esta información, el primer paso es generar una máscara que satisfaga dicha condición, para lo cual creamos la siguiente imagen:

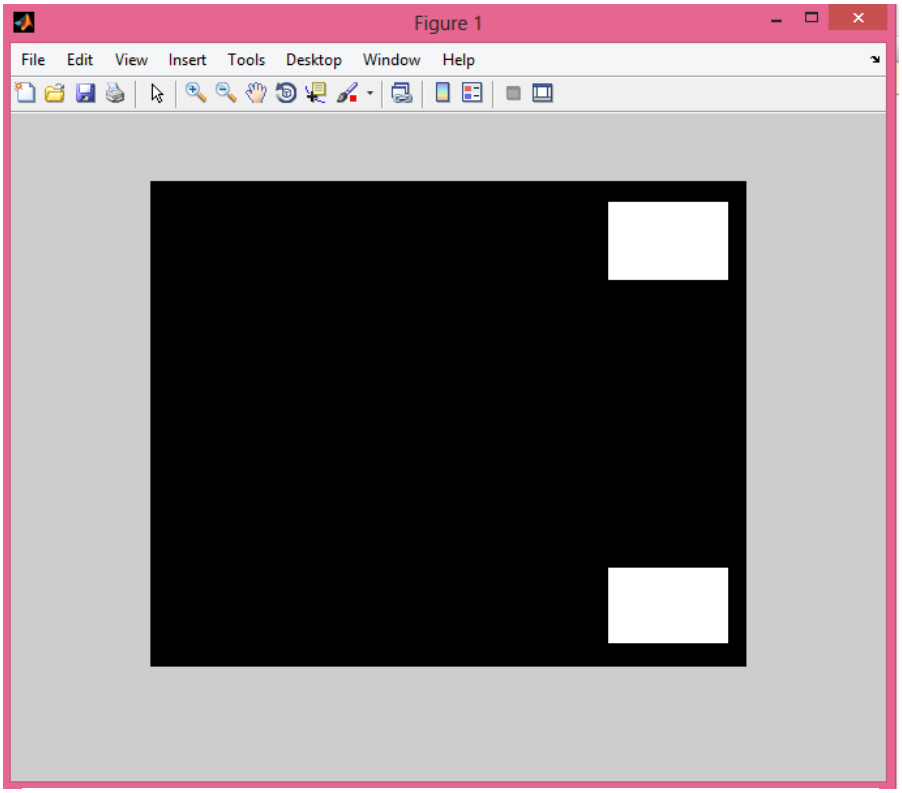

Figura 2.2.1.1: Máscara de filtrado de la imagen.

La imagen generada es de las mismas dimensiones que el vídeo patrón a partir del cual obtendremos la máscara. Como se puede ver, las zonas blancas son lo suficientemente grandes como para asegurarnos de que la mosca estará contenida en una de ellas. Una vez tenemos esta máscara, nos disponemos a analizar el vídeo, imagen a imagen, realizando un filtrado de la misma. De esta forma, tenemos dos zonas de trabajo definidas, las cuales se irán refinando hasta adaptarse lo mejor posible a la mosca.

Para crear la máscara se utilizan unos comandos básicos de MatLab para determinar, en primer

lugar, dónde queremos que estén las zonas blancas. Para ello basta con saber el número total de filas y columnas que tiene la imagen, y luego indicar el número de fila y columna donde empezarán y acabarán dichas zonas, utilizando para ello el porcentaje al que corresponden dichos números del total de las filas y columnas. En el caso de la imagen, la esquina superior empieza en el 5% y se extiende hasta el 20% de las filas totales y se indica de la siguiente forma:

```
FilaInicioSuperior=round(filas*0.05)
FilaFinSuperior=round(filas*0.2)
```

En el caso de las columnas se hace de igual manera, teniendo en cuenta el número total de columnas:

```
ColumnaInicioSuperior=round(filas*0.77)
ColumnaFinSuperior=round(filas*0.97)
```

Para la esquina inferior se repite el mismo procedimiento, en este caso el recuadro empieza en el 80% y termina en el 97% del total de filas, y las columnas son las mismas.

Como se puede ver, ninguno de los recuadros empieza o termina donde lo hace la imagen, esto es para evitar los bordes, ya que en ocasiones los videos presentan márgenes negros alrededor de la imagen que no presentan información útil, por lo cual se descartan para evitar que entorpezcan el procesado.

Una vez se tienen definidos los márgenes, sólo queda asignar a los píxeles del recuadro el color blanco:

```
for i= FilaInicioSuperior:FilaFinSuperior
    for j= ColumnaInicioSuperior:ColumnaFinSuperior
        mascara(i,j,(1:3))=1;
    end
end
```

Con este código, se recorren los píxeles de las filas y columnas indicadas y a cada uno de ellos se le asigna el valor 1 en sus tres componentes de color R, G y B. Así generamos una máscara que se ajuste a las imágenes que se van a procesar a continuación.

Como se ha dicho anteriormente, este procesado se basa en las diferencias entre imágenes, ya que en un video con logotipo, se puede observar que conforme avanza el mismo y su contenido varía, la única zona de éste que se mantiene constante todo el tiempo es la de la mosca. Ésta es creada por un generador de caracteres e insertada sobre el vídeo ya editado y listo para emisión, de forma que por más que la imagen vaya cambiando, la mosca siempre tendrá las mismas características.

Aun así, el logotipo podría presentar ciertas variaciones, no perceptibles a simple vista, pero que si se hacen evidentes al momento de procesar las imágenes, sobre todo al calcular la diferencia entre imágenes. En casos en que la mosca sea de un color muy claro, o presente algún nivel de

transparencia, la imagen que se esté emitiendo en esa zona podría interferir con el logotipo, haciendo que los valores de sus píxeles se vean afectados por los colores del fondo. De todas formas, al estar la mosca insertada encima del vídeo, por más que varíe la imagen del fondo, la zona del logotipo siempre presentará, como mínimo, el valor correspondiente a los colores de sus píxeles, de manera que aun cuando la imagen del fondo fuera completamente negra (el valor de los píxeles sería cero), en la zona de la mosca los píxeles presentarían niveles superiores, de forma que la variación en los valores de la imagen en esa zona serían menores que las variaciones en zonas sin mosca.

En este caso, a efectos de localizar el logotipo dentro de la imagen, al realizar la resta entre dos imágenes no buscamos las diferencias, sino la zona con menos diferencias de la imagen, ya que los píxeles que forman la mosca poseerán unos valores relativamente constantes a lo largo de todo el vídeo. Si bien es posible que en algunos casos los píxeles del logotipo se vean afectados por la imagen del fondo, al analizar su varianza obtenemos que ésta es menor en la zona del logotipo que en el resto de la imagen.

Cabe destacar que es necesario que el video que se utilice para este procesado sea relativamente dinámico, ya que de las diferencias entre los frames depende parte de la efectividad del procesado, por lo tanto, es recomendable evitar vídeos que presenten pocas variaciones como puede ser un noticiero o cualquier programa en que la imagen se muestre desde el mismo plano durante mucho tiempo sin que la escena cambie.

De acuerdo con lo explicado, este procesado se basa no sólo en las diferencias entre dos imágenes, sino en encontrar la zona que menos ha cambiado entre ellas. Para ello, el primer paso es realizar un filtrado de las imágenes a restar, utilizando la máscara de la Figura 2.2.1.1.

Como estamos buscando una zona que se mantenga constante conforme avanza el vídeo, no nos sirve una imagen diferencia que sea la resta de dos frames consecutivos, ya que no se registran prácticamente diferencias entre ellos y el procesado aplicado no produciría los resultados esperados. Es por ello que entre los frames a restar deberá haber transcurrido un cierto tiempo que hemos prefijado a dos segundos, de forma que haya suficientes variaciones entre ellos como para obtener una imagen diferencia que proporcione suficiente información para que el procesado sea efectivo.

En la Figura 2.2.1.2 se puede observar el resultado del filtrado de dos frames a procesar, realizado con los siguientes comandos:

*img=mascara.*frame1*

*img2=mascara.*frame2*

A partir de ahora, todos los procesados se realizarán sobre las imágenes filtradas, por lo tanto, de aquí en adelante, al referirnos a las imágenes o frames, se hará referencia a los obtenidos después del filtrado.

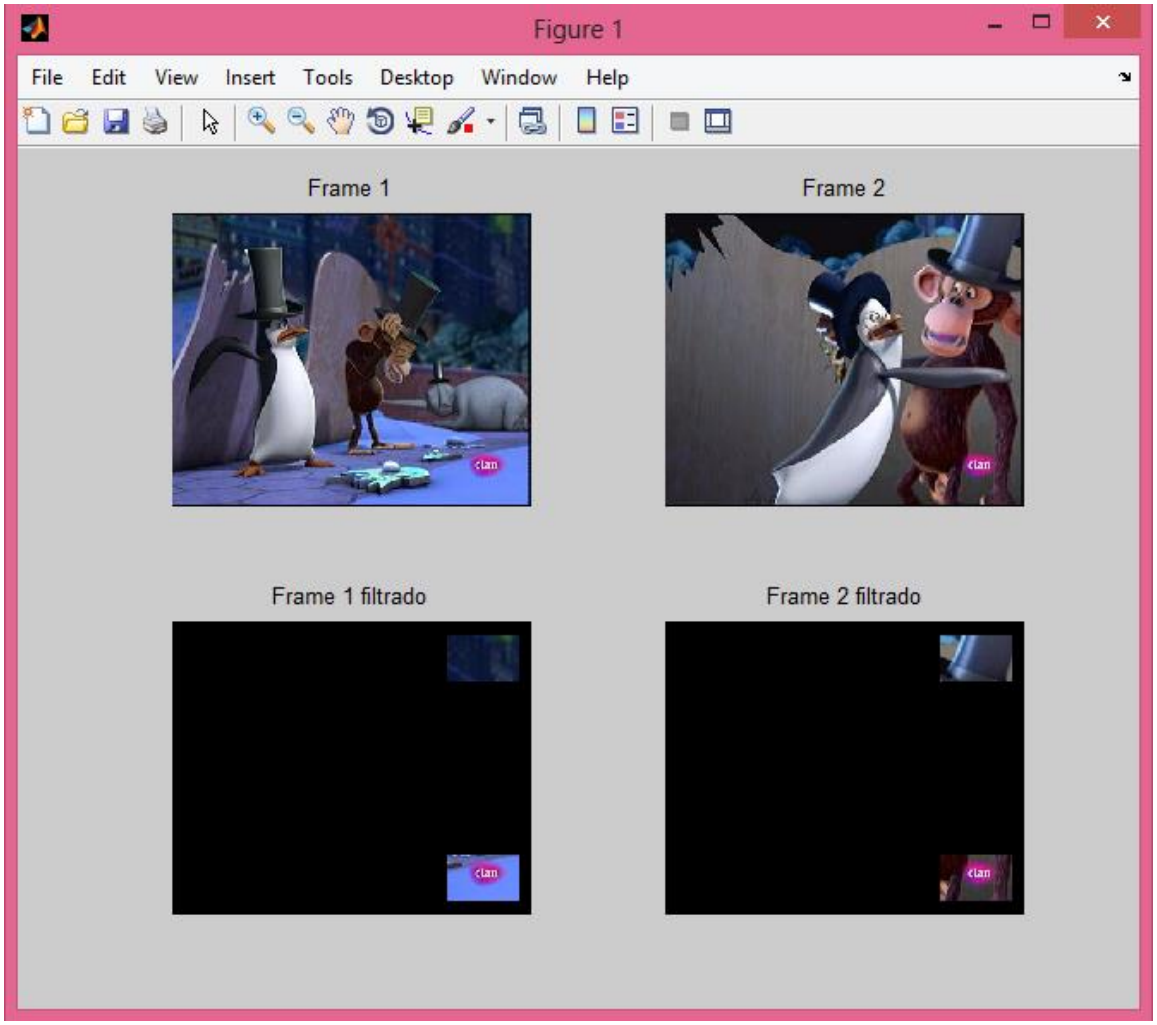

Figura 2.2.1.2: Filtrado de las imágenes a procesar.

Las imágenes a procesar están en el espacio de color RGB, por lo cual cada una de ellas posee tres dimensiones, una para cada color, de forma que cualquier operación que se realice sobre ellas se ejecutará por triplicado, incrementando de la misma manera el tiempo de procesado. Por ello y debido a que es más cómodo y sencillo trabajar con imágenes bidimensionales, se las convertirá a escala de grises, lo cual realizamos con el comando *rgb2gray* de la siguiente manera:

```
im_grises=rgb2gray(img)
im2_grises=rgb2gray(img2)
```

Llegados a este punto, disponemos de las imágenes filtradas y en escala de grises. El siguiente paso es realizar la resta de las imágenes y trabajar con los valores obtenidos.

La resta de imágenes se realiza en valor absoluto, ya que buscamos las diferencias entre las dos imágenes, independientemente del contenido de las mismas. Si no se realizara el valor absoluto, perderíamos la mitad de la información necesaria, por lo cual el comando a ejecutar es el siguiente:

```
imagen=abs(im2_grises-im_grises)
```

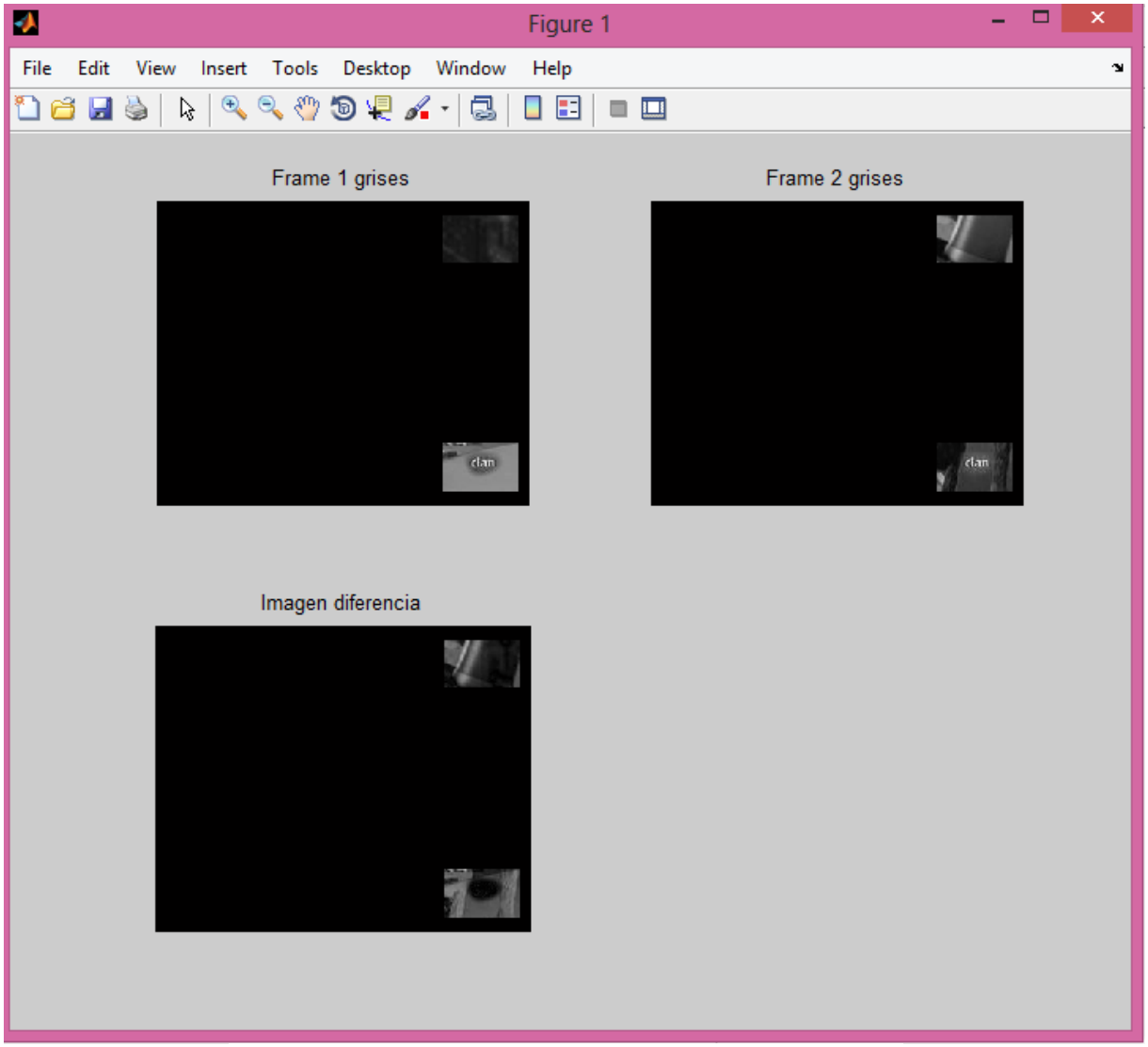

Figura 2.2.1.3: Resta de imágenes

En la Figura 2.2.1.3 se pueden ver los frames originales y el resultado de la resta de los mismos.

La Figura 2.2.1.4 es una ampliación de la imagen diferencia sobre el cuadrante donde está el logotipo. En ella se puede ver perfectamente que en la zona donde se encuentra la mosca, la gran mayoría de los píxeles está a nivel de negro, a excepción de algunos que presentan tonos más grises. Esto significa que en esas zonas de la mosca los píxeles correspondientes en ambas imágenes tenían valores diferentes, lo cual puede deberse a que la imagen de fondo esté afectando a los valores de los píxeles de la mosca.

Figura 2.2.1.4: Zoom a la imagen diferencia sobre el cuadrante inferior derecho.

A partir de ahora ya no se trabajará con la imagen entera con los dos cuadrantes, sino que se operará sobre cada uno de los cuadrantes independientemente y separados del resto de la imagen.

Para seguir con el procesado, primero hemos de tener cada uno de los cuadrantes por separado. Éstos no son más que una zona de la imagen, por lo tanto podemos crear una nueva matriz que contenga solamente la información presente en la zona de interés.

Para crear la imagen valores, el primer paso es inicializar la variable en la que se guardará la información del cuadrante, con sus dimensiones correspondientes:

```
valores=zeros(FilaFinSuperior - FilaInicioSuperior +1, ColumnaFinSuperior - ColumnaInicioSuperior +1);
```

Ejecutando el comando anterior creamos una matriz de píxeles con las dimensiones *FilaFinSuperior - FilaInicioSuperior +1* filas y *ColumnaFinSuperior - ColumnaInicioSuperior +1* columnas, en la cual todos los píxeles tienen valor cero.

Una vez tenemos la imagen valores creada, solo resta copiar el valor de cada píxel del cuadrante en su píxel correspondiente de la imagen, lo cual se hace como sigue:

```
for j=1:(ColumnaFinSuperior- ColumnaInicioSuperior +1)
    for i=1:( FilaFinSuperior- FilaInicioSuperior +1)
        valores(i,j)=imagen((i+FilaInicioSuperior-1),(j+ColumnaInicioSuperior-1));
    end
end
```

Este código recorre la imagen valores, y a cada uno de sus píxeles le asigna el valor de su píxel correspondiente en la imagen diferencia. Para ello simplemente hay que añadir a los índices de

filas y columnas de la imagen diferencia los números de fila y columna en los que comienza el cuadrante. De esta forma obtenemos una imagen en la cual solamente tenemos la información correspondiente a la zona de interés

En este caso, igual que para crear la máscara, el código es el mismo tanto para la esquina superior como para la inferior, cada una con su información de filas y columnas correspondiente.

Después de ejecutar el código, disponemos de dos matrices llamadas *valores* y *valores2* que contienen la información de los cuadrantes inferior y superior de la imagen diferencia respectivamente.

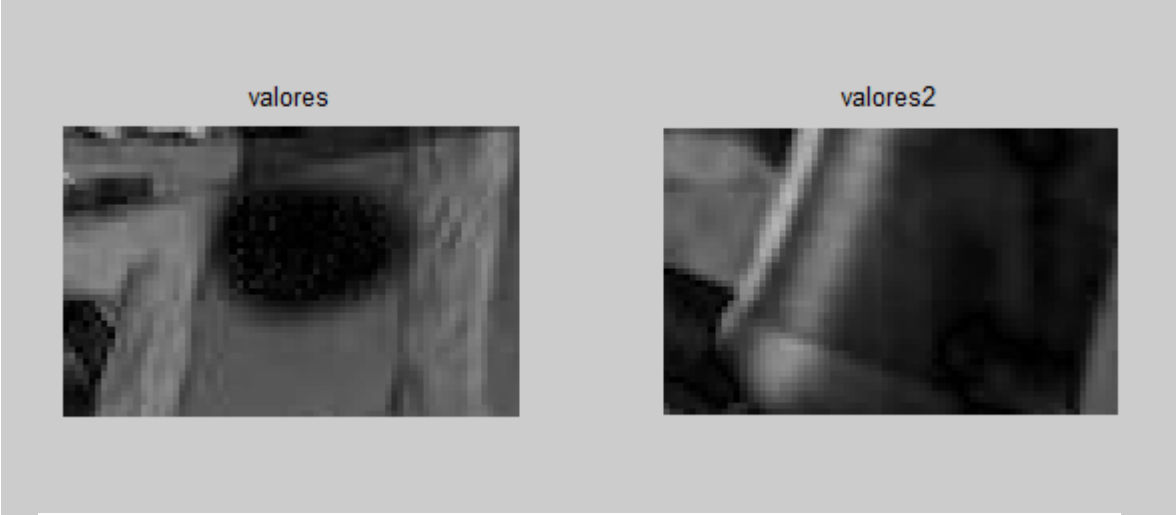

Figura 2.2.1.5: Imágenes valores y valores2, extraídos de la imagen diferencia.

A las dos imágenes - *valores* y *valores2* - se les aplicará los mismos procesados que veremos a continuación, por tanto, al referirnos a las imágenes o la imagen, se hará referencia a éstas.

Para poder identificar la mosca, debemos buscar dentro de cada una de las imágenes la zona o zonas en las que los píxeles hayan variado menos a lo largo del vídeo. Para ello, necesitamos disponer de los valores presentados por cada uno de los píxeles de la imagen en cada frame del vídeo analizado.

Normalmente, se crearía un vector para cada píxel de la imagen, en el cual se almacenaría el valor que posee el mismo en cada una de las imágenes analizadas, pero en este caso no es una opción recomendable, ya que en el caso de las imágenes utilizadas como ejemplo - Figura 2.2.1.5 - cada una de ellas es de dimensiones 45 x 71, lo que significa que cada una posee 3195 píxeles, con lo que si definiéramos un vector para cada píxel de cada imagen, deberíamos trabajar con 6390 vectores diferentes, lo cual es muy engorroso ya que deberíamos inicializar cada uno manualmente y el manejo de tantas variables es incómodo y podría llevar a errores.

Es por ello que para poder disponer de la información necesaria se ha recurrido al uso de estructuras. Dentro de una estructura se puede almacenar toda la información que haga falta, ya que dentro de ésta se pueden crear subestructuras y dentro de cada una declarar todas las variables que se desee.

A continuación se describirán las estructuras creadas, pero no se entrará en detalles sobre cómo se han creado.

Para poder trabajar cómodamente con la información de las imágenes se crearon dos

estructuras iguales, una por imagen. En ellas se declaran tantas subestructuras como píxeles posea la imagen, y en cada una de las subestructuras - una para cada píxel - se declara un vector que contendrá los valores presentados por el píxel en cada una de las imágenes analizadas. De esta forma dispondremos de un vector de valores para cada píxel de la imagen.

Como ya hemos dicho, para determinar dónde está la mosca, se buscan las zonas que se mantuvieron constantes a lo largo del vídeo, representadas por las partes negras en las imágenes.

Para ello se analiza el comportamiento de cada píxel en todas las imágenes procesadas, y así se obtiene una idea de lo que ha variado, o lo que es lo mismo, se calcula su **varianza**. De esta forma, tras analizar cada uno de los píxeles de la imagen, se crea una matriz - del mismo tamaño que la imagen - en la cual se guarda la varianza de cada píxel a lo largo del vídeo en su posición correspondiente, o dicho de otra forma, ahora tenemos una imagen de la varianza de los píxeles.

Con la imagen de varianzas ya tenemos toda la información necesaria para determinar en dónde está la mosca. El siguiente paso es establecer el nivel de varianza máximo que podrán tener los píxeles del logotipo, lo cual haremos mediante una umbralización.

Antes de realizar la umbralización normalizaremos la imagen varianza dividiendo el valor de cada píxel por el valor máximo, así los píxeles tendrán valores entre 0 y 1:

imagvar=varianza./max(max(varianza))

A continuación se pueden ver las imágenes varianza correspondientes al video del cual se extrajeron las imágenes que se han utilizado como ejemplo.

Figura 2.2.1.6: Imagen varianza

Figura 2.2.1.7: Imagen varianza2

En la Figura 2.2.1.6 se observa la varianza de los píxeles de la esquina inferior derecha del vídeo a lo largo del mismo, y en la Figura 2.2.1.7 la de la esquina superior derecha. En ambas se pueden observar los píxeles que poseen mayor varianza en tonos de gris más claros, y los que menos han variado en tonos más oscuros llegando al negro en la zona de la mosca.

Para establecer el umbral de varianza para la segmentación de las imágenes, nos valdremos del histograma de las mismas para ayudarnos a seleccionar dicho umbral de una forma más objetiva. Para la realización del histograma bastará con ejecutar la siguiente línea:

hist(imagvar(:),100)

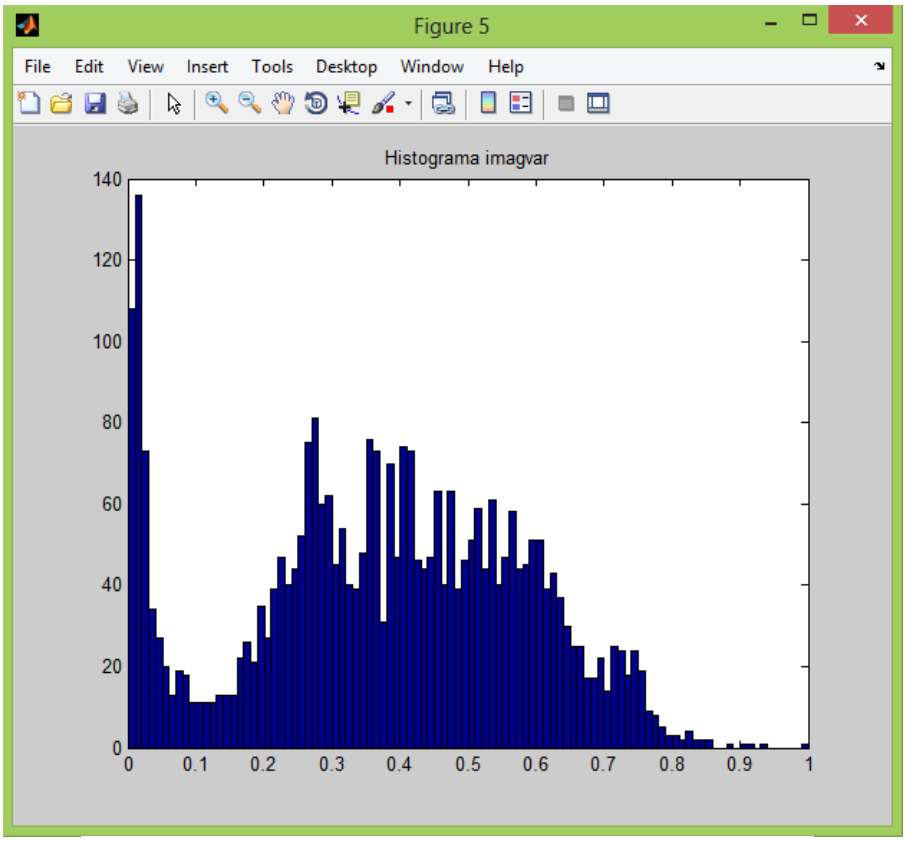

Figura 2.2.1.8: Histograma de imagvar

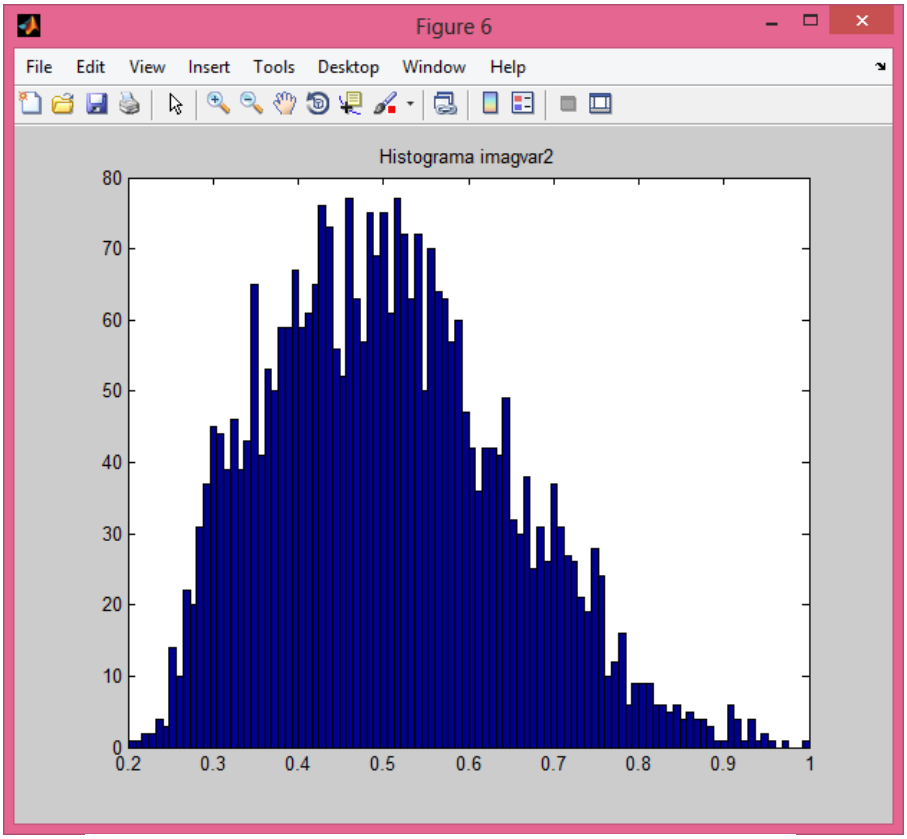

Figura 2.2.1.9: Histograma de imagvar2

El histograma muestra, para cada nivel de varianza - eje x - la cantidad de píxeles que hay en la imagen con dicho nivel.

En la Figura 2.2.1.8 se muestra el histograma de la imagen *varianza* - *imagvar* - , en el que se puede apreciar claramente cómo, en torno al nivel de varianza 0.1 hay una clara disminución de la cantidad de píxeles a dicho nivel, dejando dos grupos claramente diferenciados, uno son las varianzas menores de 0.1, donde se registra un grupo de pocos píxeles - respecto del total- y el otro grupo son las varianzas mayores de 0.1 donde se encuentra la gran mayoría de píxeles de la imagen.
Como los pixeles que pertenezcan a la mosca presentarán los niveles de varianza más bajos - debido a su constancia en el vídeo - , observando el histograma se ve claramente que el nivel umbral ha de estar en torno a 0.1. Mirando la imagen varianza - Figura 2.2.1.6 - se ve claramente la relación que mantiene con el histograma.

En la Figura 2.2.1.9 se muestra el histograma de la imagen *varianza2* - *imagvar2* -, en el que se puede observar que a diferencia del histograma de *imagvar*, no hay grupos definidos, sino que la mayoría de los píxeles presentan niveles de varianza entre 0.25 y 0.9, siendo 0.2 la varianza mínima en dicha imagen. En este caso, podemos decir con seguridad que la mosca no está presente, lo cual se corresponde con la imagen varianza2 - Figura 2.2.1.7 - De todas maneras ha de establecerse un umbral para la segmentación de la imagen, que se encontrará igual que en *imagvar*, en torno al 0.1.

A continuación se van a mostrar diferentes ejemplos con algunas de las imágenes varianza y sus correspondientes histogramas, que han sido analizadas en busca de un umbral que sea válido para las moscas de cadenas emisoras diferentes, situadas en cualquiera de las dos zonas de interés.

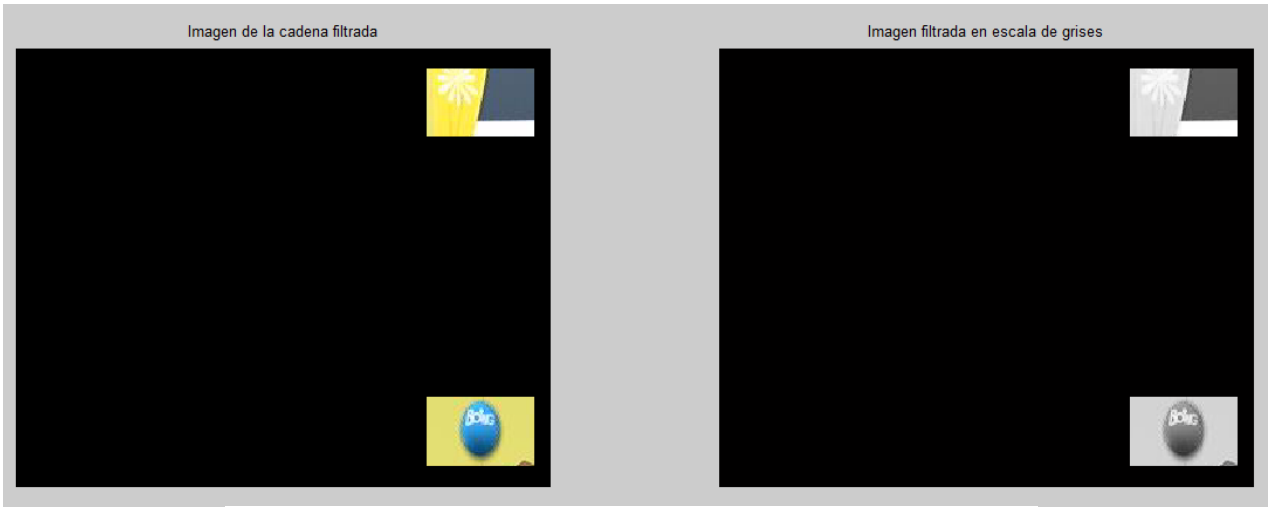

Figura 2.2.1.10: Imagen con mosca de la cadena Boing.

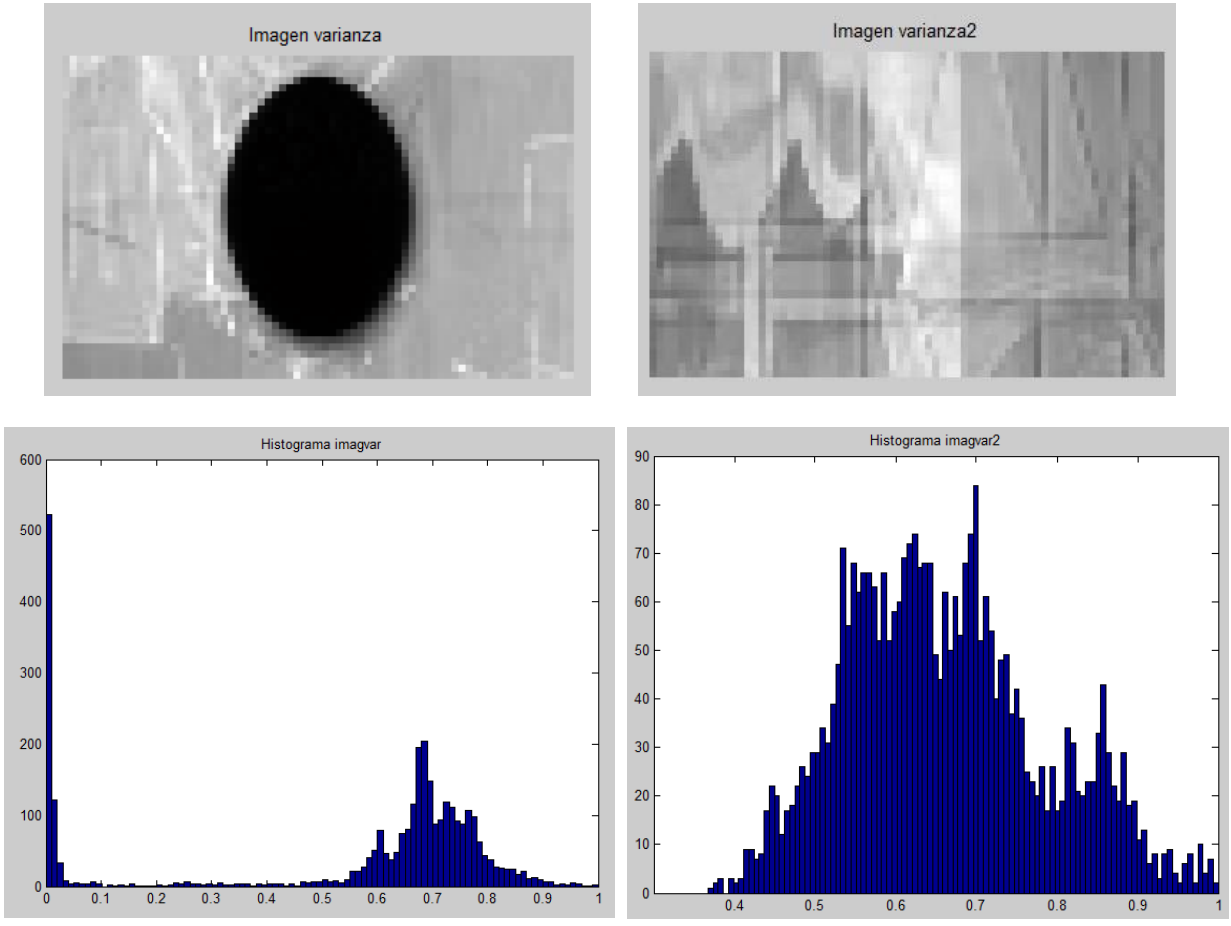

Figura 2.2.1.11: Imágenes varianza e histogramas de la cadena Boing

En la Figura 2.2.1.11 se puede observar que para la imagen *varianza*, el histograma muestra por un lado una concentración de píxeles de varianza entre 0 y 0.1, y por otro la mayoría de los píxeles de la imagen entre las varianzas 0.55 y 0.9. A partir de esta información se puede decir que un umbral óptimo de varianza estaría en torno a 0.1.

En el histograma de la imagen *varianza2* casi todos los píxeles de la imagen se encuentran entre las varianzas 0.4 y 1, lo cual nos indica que la máscara no está en la imagen.

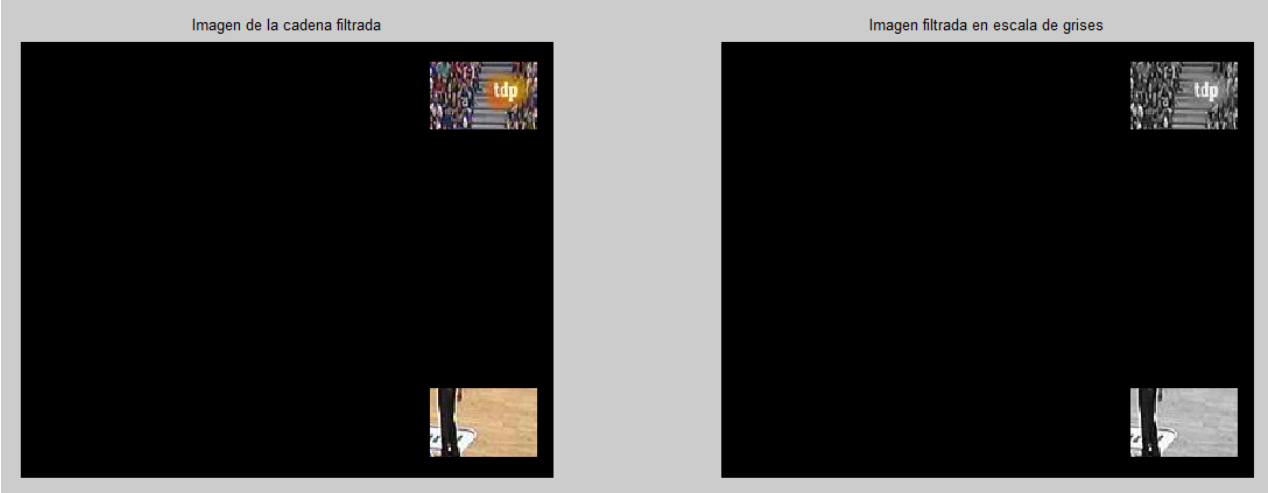

Figura 2.2.1.12: Imagen con mosca de la cadena TDP

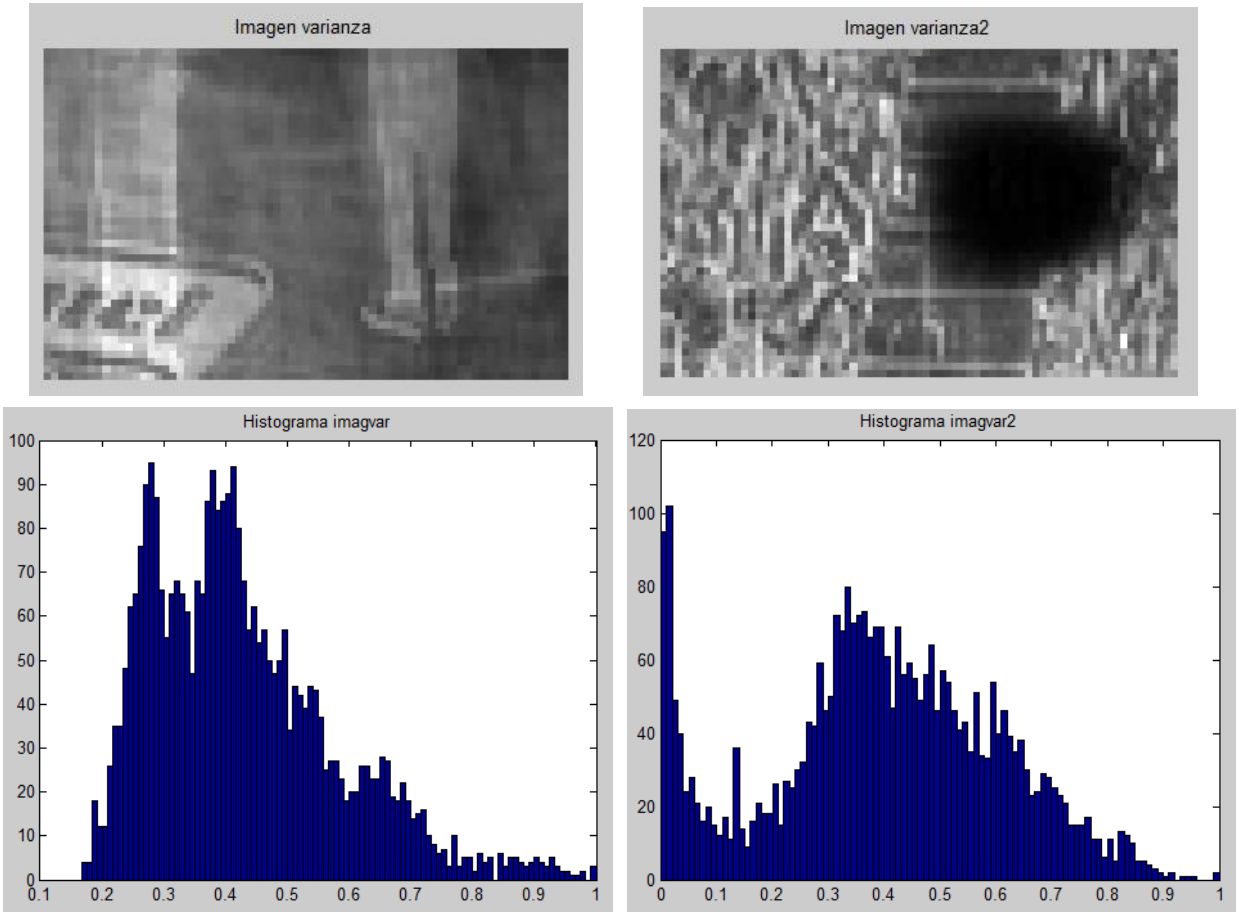

Figura 2.2.1.13: Imágenes varianza e histogramas de la cadena TDP

En la Figura 2.2.1.13 el histograma de la imagen *varianza* muestra que los píxeles de la imagen poseen valores de varianza de entre 0.2 – aproximadamente – y 1, de forma que se puede asegurar que la mosca no se encuentra en la imagen.

En el histograma de la imagen *varianza2* se observa como va aumentando la cantidad de píxeles entre las varianzas 0 y 0.1 conforme ésta disminuye, y, por otro lado, se ve una gran concentración entre los niveles de varianza 0.2 y 0.75. Dadas las características de la imagen se podría tomar como valor umbral una varianza de 0.1 aproximadamente.

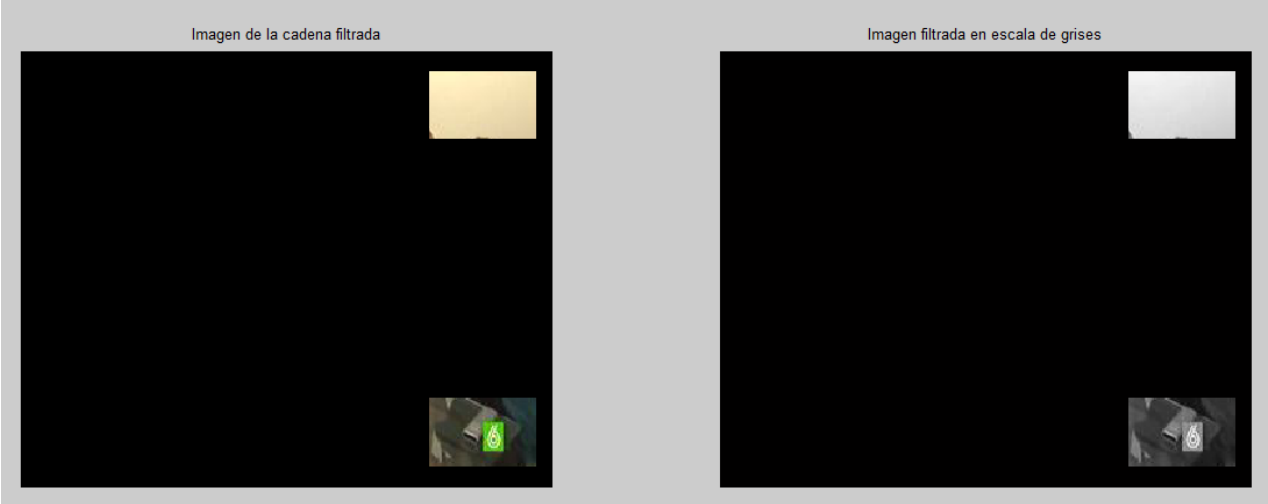

Figura 2.2.1.14: Imagen con mosca de la cadena La Sexta

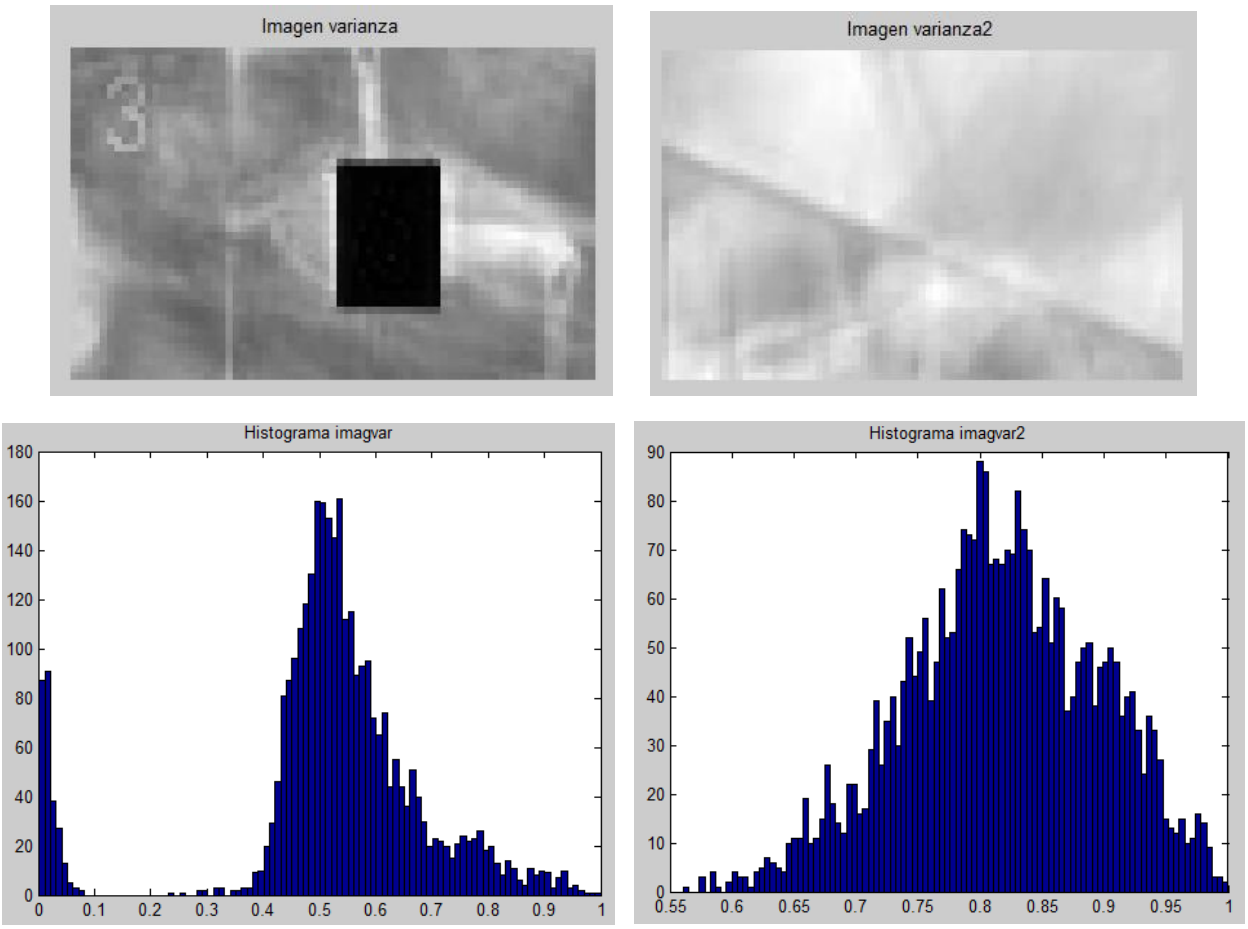

Figura 2.2.1.15: Imágenes varianza e histogramas de la cadena La Sexta

En la Figura 2.2.1.15 se puede apreciar claramente en el histograma de la imagen *varianza* que hay un grupo de píxeles entre los niveles de varianza 0 y 0.1 separados del resto de píxeles de la imagen, que se encuentran mayormente entre las varianzas 0.4 y 0.8 aproximadamente. Observando la imagen se puede determinar sin lugar a errores que un umbral de 0.9 sería perfectamente válido para esta imagen.

En el histograma de la imagen *valores2* se ve que todos los píxeles de la imagen presentan varianzas superiores a 0.55.

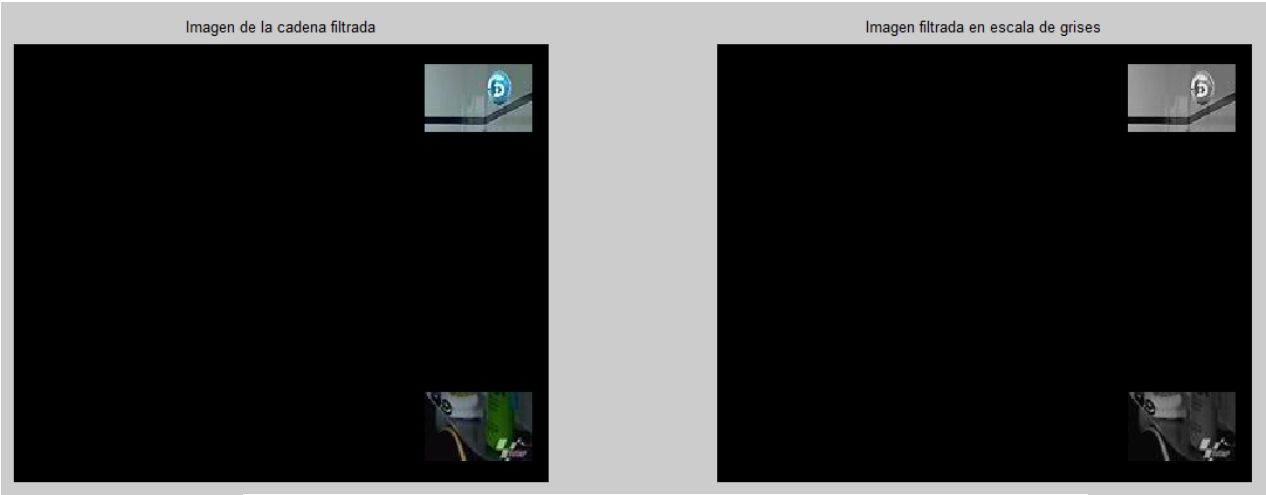

Figura 2.2.1.16: Imagen con mosca de la cadena Telecinco

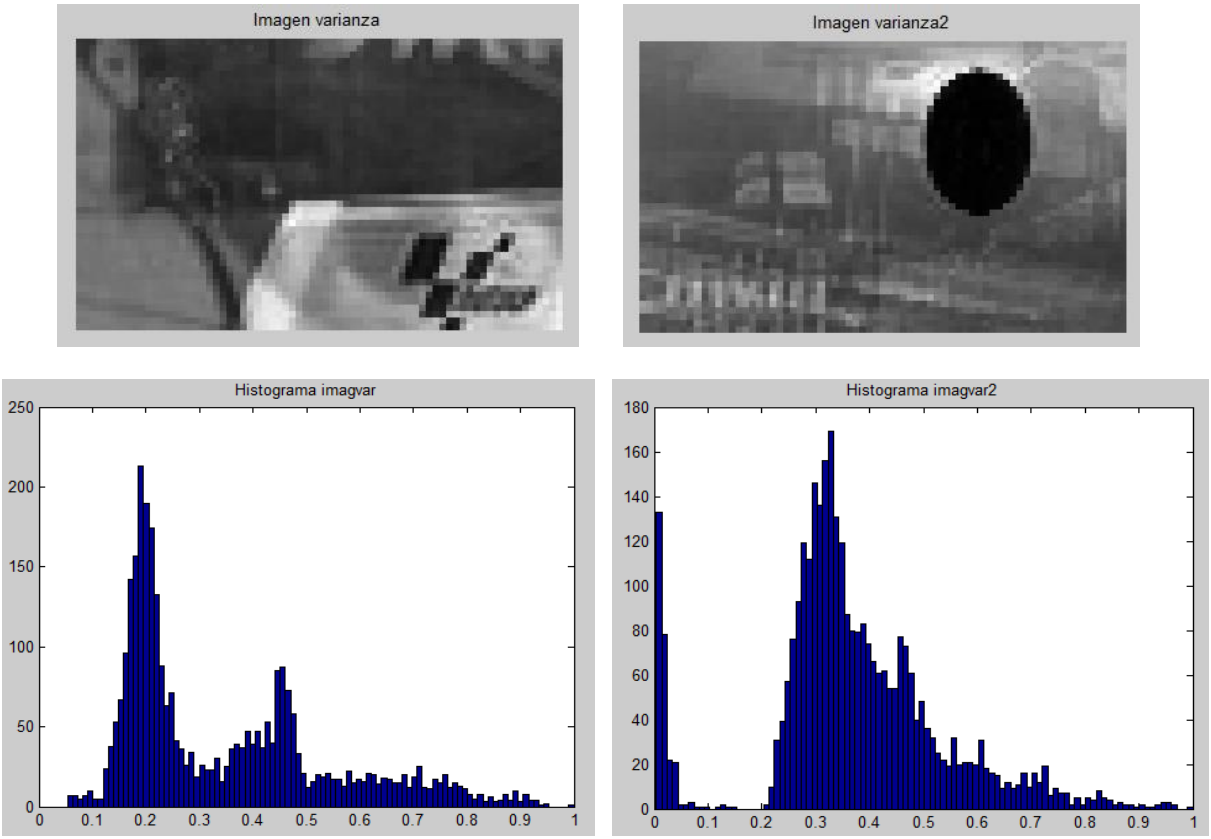

Figura 2.2.1.17: Imágenes varianza e histogramas de la cadena Telecinco

En la Figura 2.2.1.17 el histograma de la imagen *varianza* muestra que la gran mayoría de los píxeles de la imagen posee varianzas de entre 0.15 y 0.5.

En cuanto a la imagen *varianza2*, se puede observar un grupo de píxeles entre las varianzas 0 y 0.09, claramente separados del resto de píxeles de la imagen, repartidos entre las varianzas 0.2 y 1. En este caso, un valor óptimo de umbral podría ser 0.09.

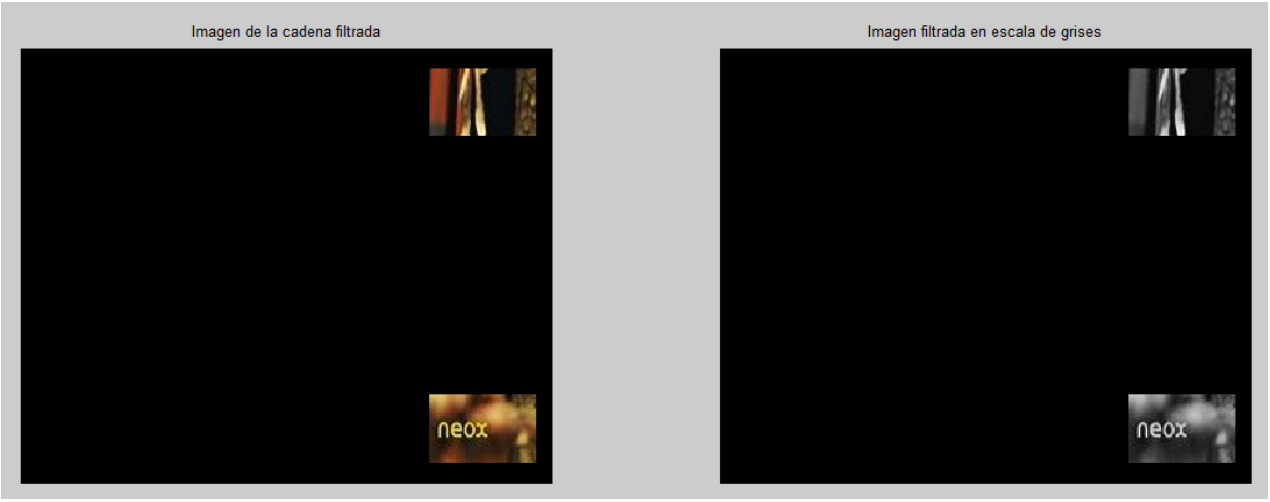

Figura 2.2.1.18: Imagen con mosca de la cadena Neox

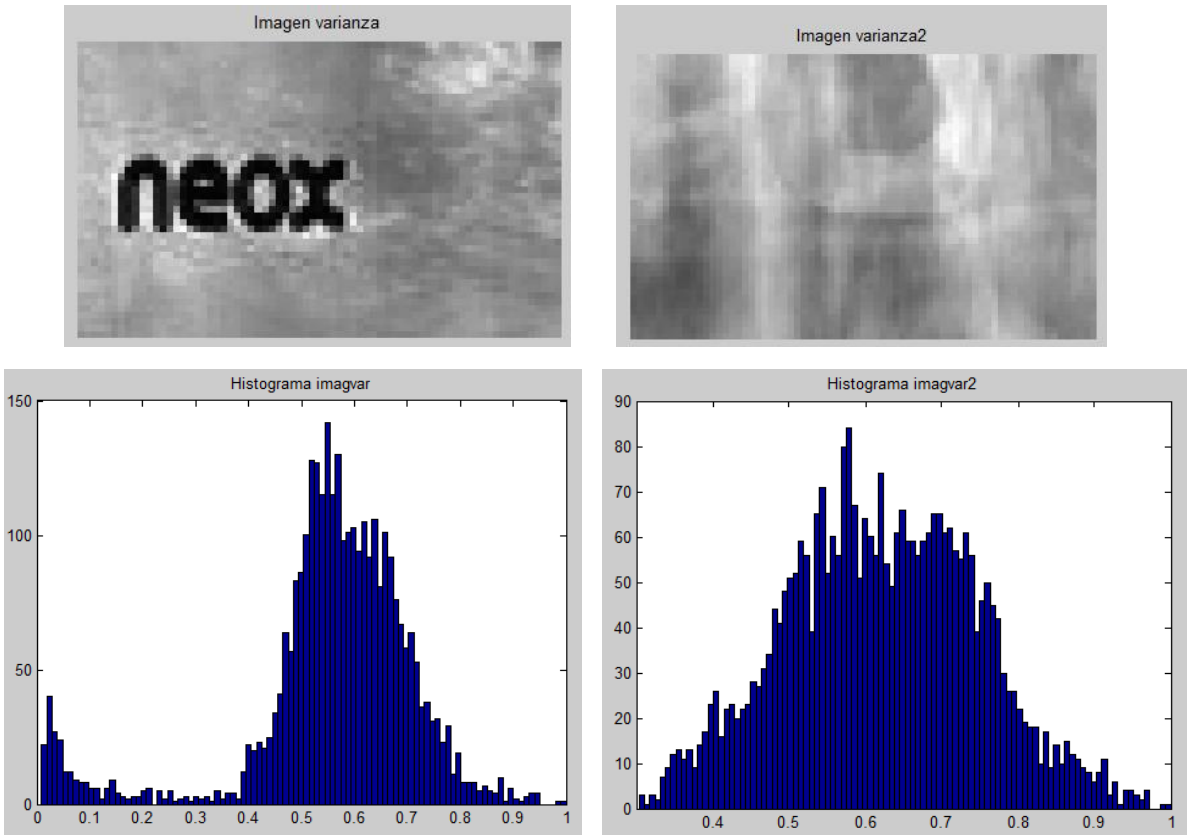

Figura 2.2.1.19: Imágenes varianza e histogramas de la cadena Neox

En la Figura 2.2.1.19 el histograma de la imagen *varianza* muestra que la mayoría de los píxeles de la imagen poseen valores de varianza de entre 0.4 y 0.9. Además, hay una concentración de píxeles de varianza inferior a 0.1, que por su distribución y comprobando en la imagen podemos decir que corresponden a la mosca.

En el histograma de la imagen *varianza2* se observa que los píxeles de la imagen están repartidos entre todos los niveles de varianza, presentando una mayor cantidad entre las varianzas 0.5 y 0.8, y muy pocos píxeles en niveles inferiores a 0.2.

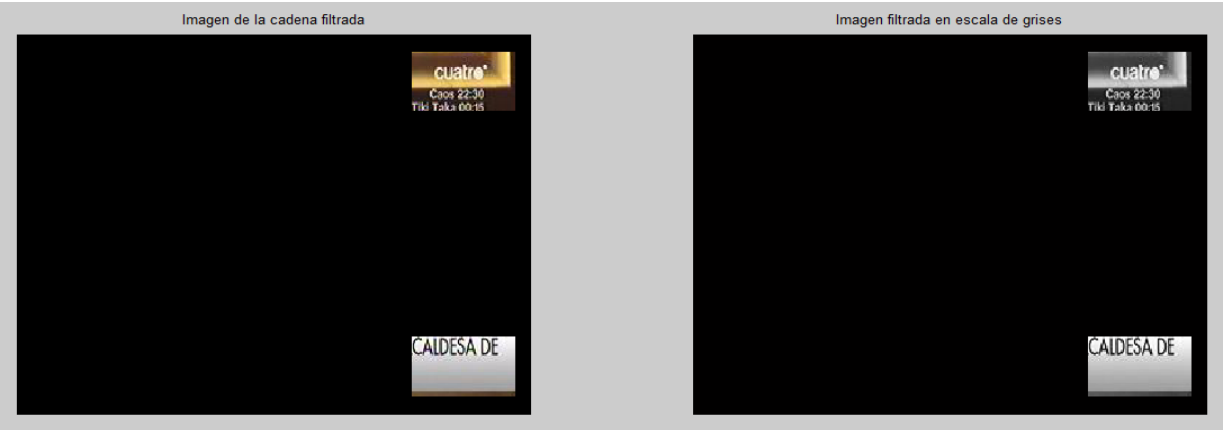

Figura 2.2.1.20: Imagen con mosca de la cadena Cuatro

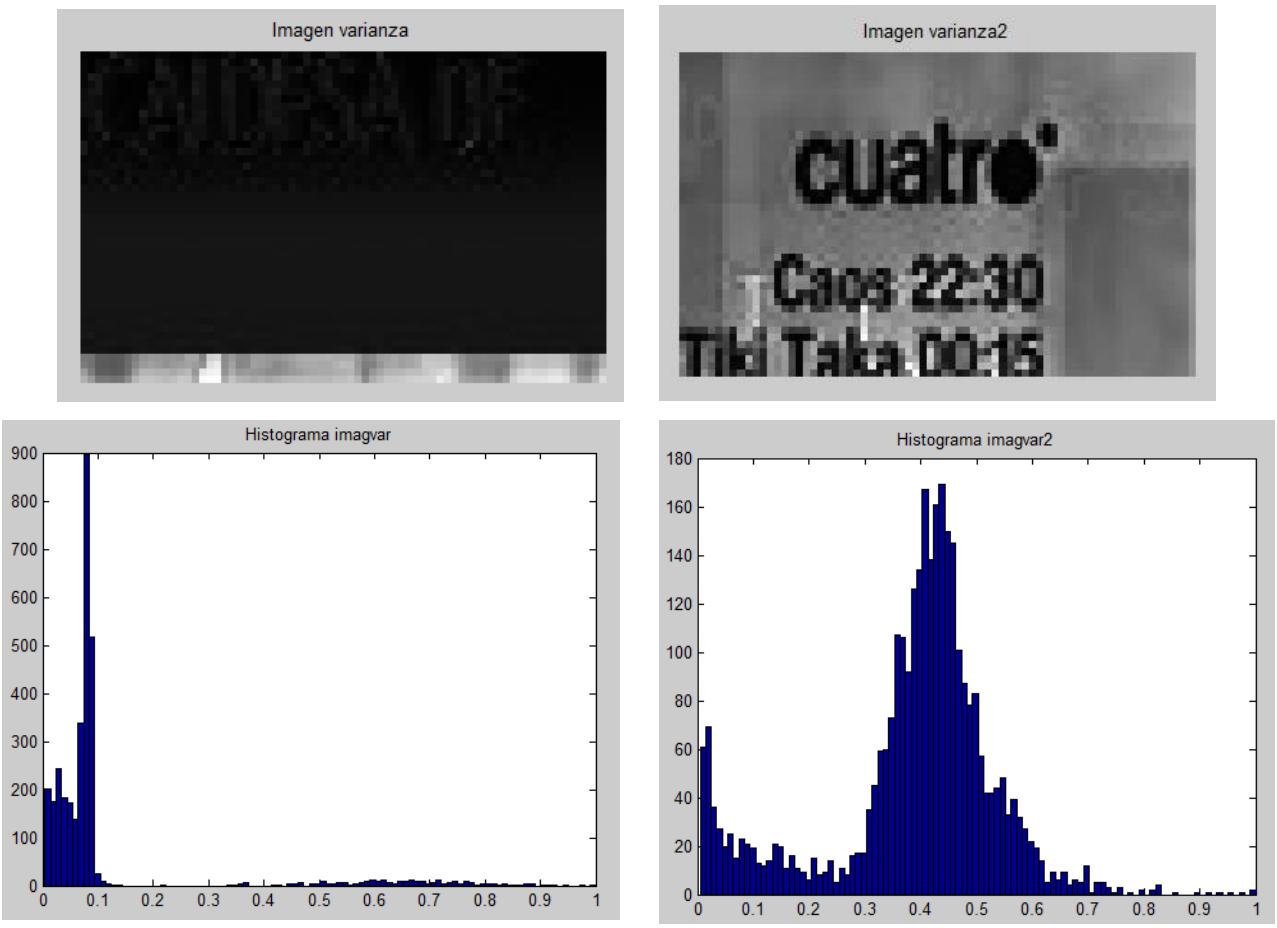

Figura 2.2.1.21: Imágenes varianza e histogramas de la cadena Cuatro

En la Figura 2.2.1.21 se puede ver que en el histograma de la imagen *varianza* la mayoría de los píxeles de la imagen tienen una varianza entre 0 y 0.1, por lo cual, teniendo en cuenta los valores de los posibles umbrales vistos anteriormente, tras la umbralización la imagen permanecerá casi igual.

En el histograma de la imagen *varianza2*, se puede ver que la mayoría de píxeles se encuentra entre las varianzas 0.3 y 0.7. También se ve que para las varianzas inferiores a 0.1 hay un grupo de píxeles que son los correspondientes a la mosca y las letras que se pueden ver en la imagen. Dichas letras forman parte de la mosca, por tanto tienen la misma varianza. En este caso, se intentará eliminarlas procesando la imagen obtenida, pero en caso de que no se puedan eliminar totalmente, el problema se solucionaría limitando el tamaño de la máscara inicial para esa zona.

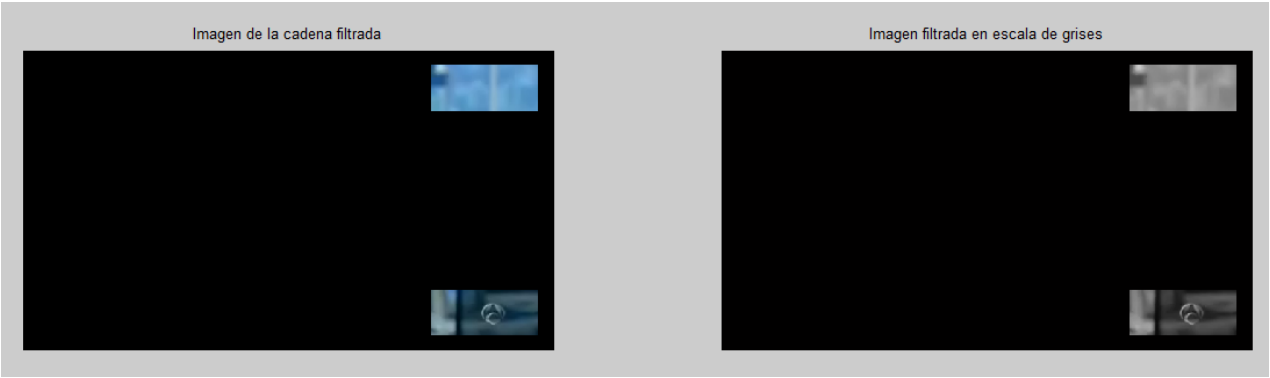

Figura 2.2.1.22: Imagen con mosca de la cadena Antena 3

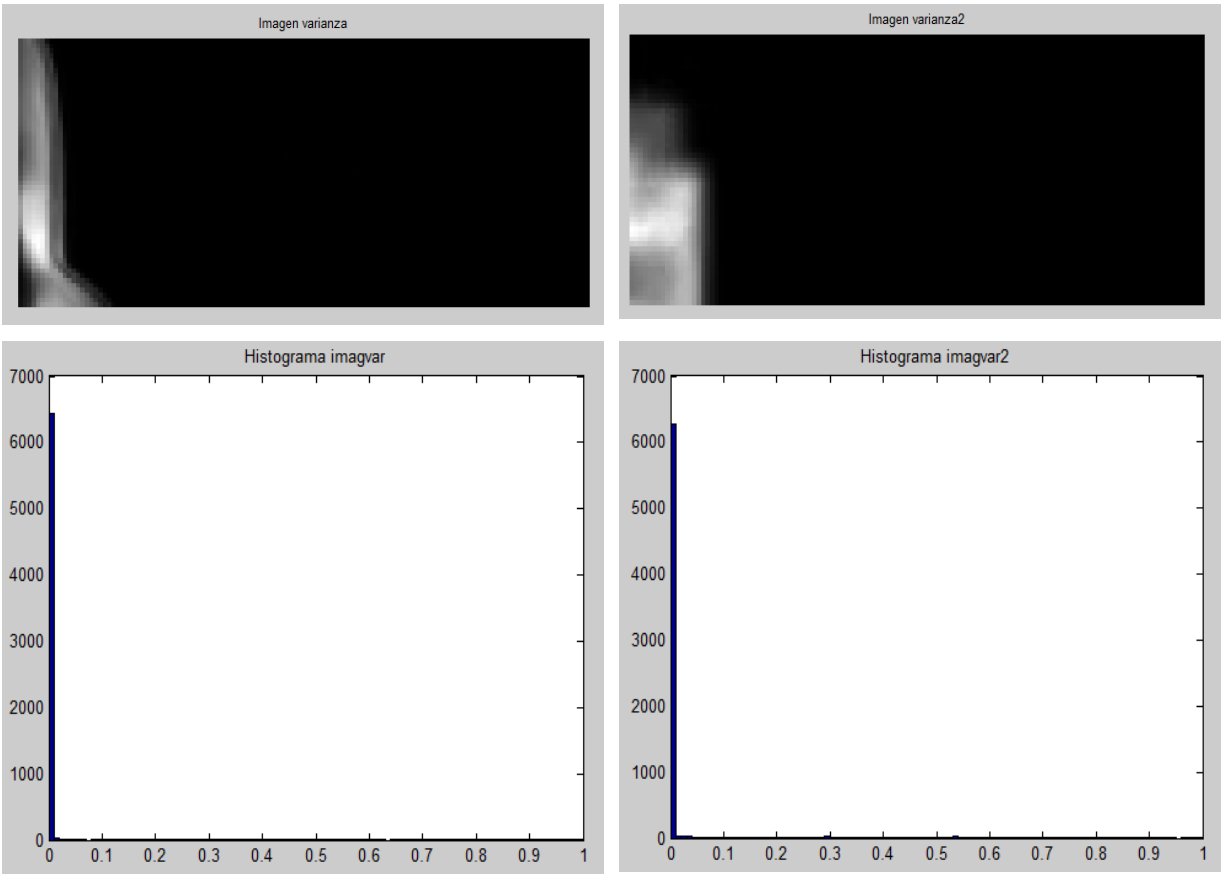

Figura 2.2.1.23: Imágenes varianza e histogramas de la cadena Antena 3

En la Figura 2.2.1.23, se puede ver que en los histogramas de ambas imágenes – *varianza* y *varianza2* – prácticamente todos los píxeles de ambas imágenes tienen varianza cero, a excepción de unos pocos que presentan varianzas mayores – correspondientes a las zonas claras de ambas imágenes –. Esto se debe a que son resultado del análisis de parte de un noticiero, en el cual sólo se muestra al presentador en un plano fijo, por ello la imagen no presenta casi variaciones.

Si se analiza el vídeo completo, en el cual, además del presentador se ve parte de un reportaje, se obtienen los siguientes resultados:

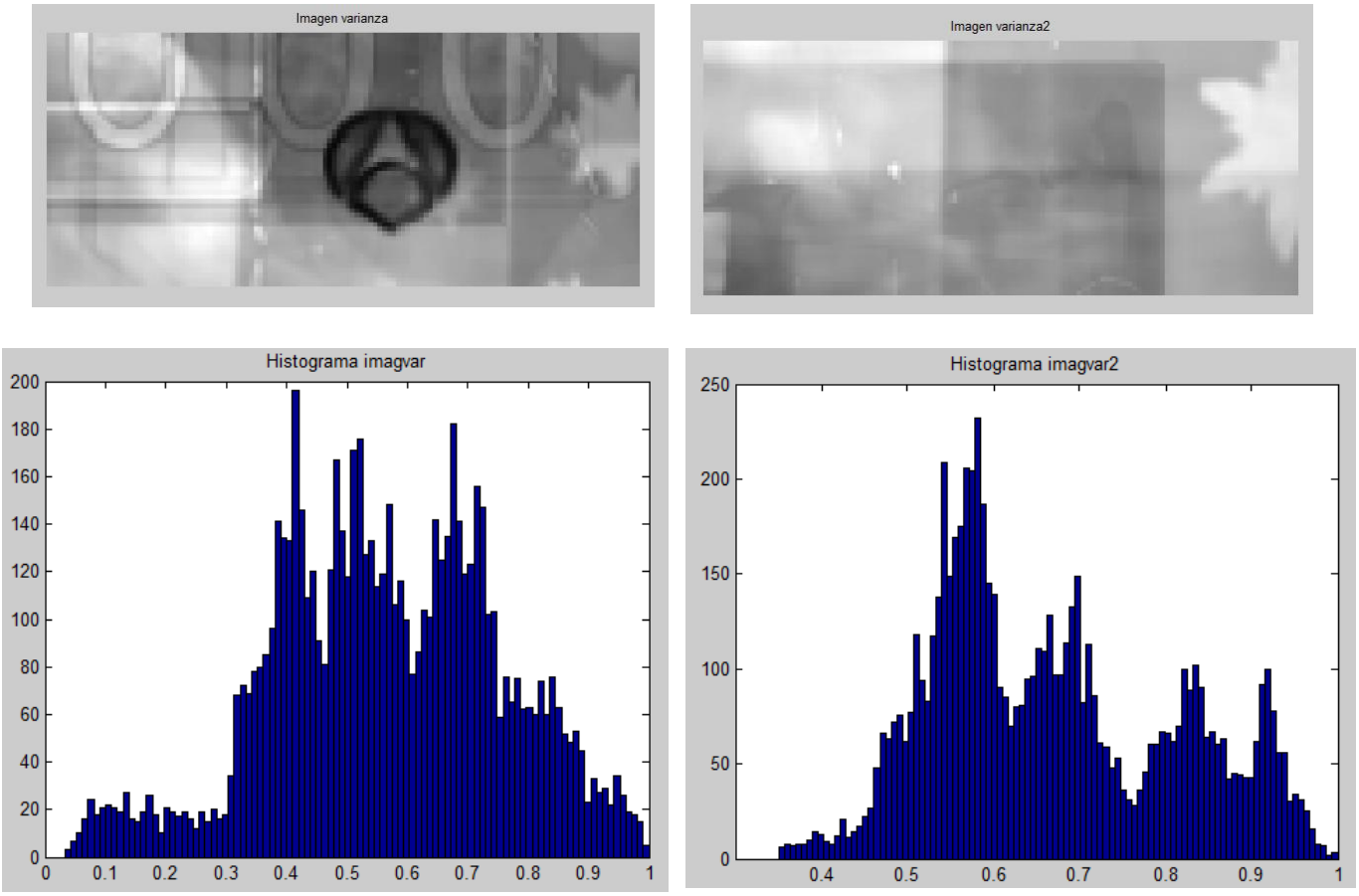

Figura 2.2.1.24: Imágenes varianza e histogramas de la cadena Antena 3

En la Figura 2.2.1.24 se muestran las imágenes varianza y sus histogramas, correspondientes al análisis del vídeo completo de la cadena Antena 3.

Observando las imágenes varianza se pueden apreciar las diferencias respecto de las mostradas en la Figura 2.2.1.23.

En la imagen *varianza* la mosca se distingue perfectamente del fondo de la imagen. En su histograma se puede ver que el valor mínimo de varianza no es cero, sino que está alrededor de 0.04, esto se debe a que este logotipo es muy claro, casi transparente del todo, por lo cual la imagen del fondo interfiere constantemente con él.

En la imagen *varianza2* también se aprecian los cambios. En su histograma se puede ver que todos los píxeles de la imagen poseen un nivel de varianza superior a 0.2, de forma que al realizar la umbralización, todos ellos se eliminarán.

Con este ejemplo se quiere demostrar que la efectividad del procesado depende de lo dinámico que sea el vídeo y de la duración del mismo, por lo cual no siempre es completamente efectivo.

Tras el análisis de varios vídeos de diferentes cadenas, haciendo uso de la información presentada en sus histogramas y realizando numerosas pruebas, se llegó a la conclusión de que el umbral con el que se consiguen buenos resultados en la gran mayoría de los vídeos es 0.09, por lo cual, éste será el valor que se utilizará.

Una vez seleccionados los umbrales, se realiza la segmentación, en la cual los píxeles que presenten un nivel inferior al umbral tendrán valor 1 y aquellos cuyo nivel sea superior se los pondrá a 0. De esta forma tendremos una nueva imagen en la cual se verá el objeto – en este caso la mosca – en color blanco sobre un fondo negro, como se muestra a continuación:

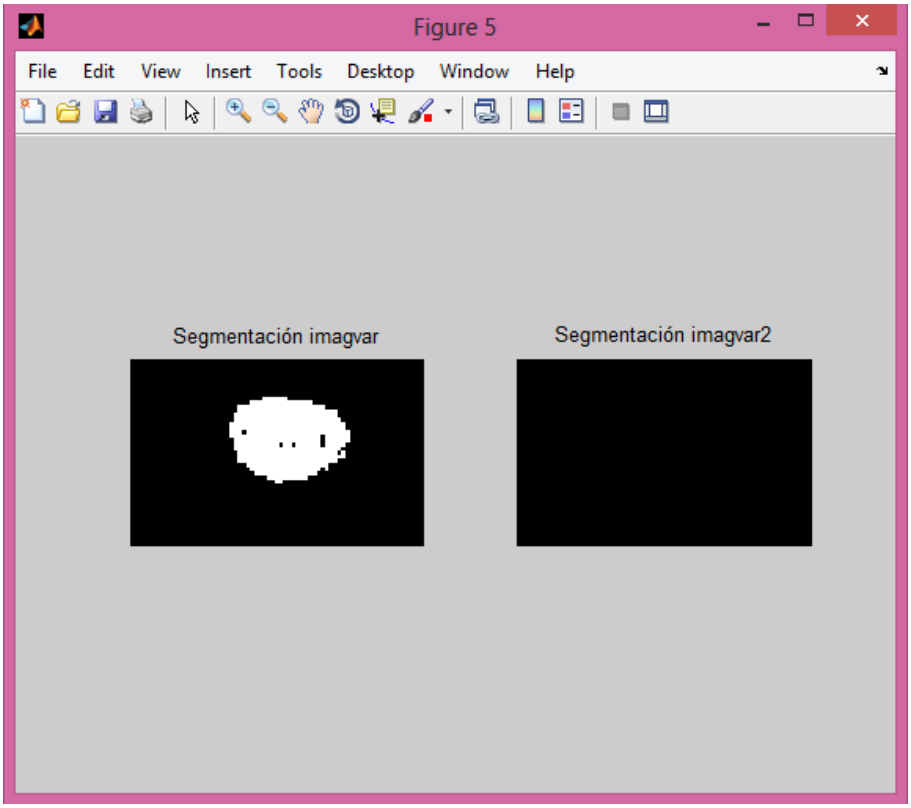

Figura 2.2.1.25: Resultado de la segmentación

En la Figura 2.2.1.25 se puede ver el resultado de la segmentación, donde se observa que en la imagen correspondiente a la esquina inferior – *imagvar* – se ha detectado gran parte de la mosca – a excepción de algunos píxeles del centro – y en la imagen correspondiente a *imagvar2* no se ha detectado ningún píxel, como debe ser, ya que allí no se encuentra el logotipo. Este resultado es el ideal, pero no siempre se obtiene, por lo cual recurrimos a determinadas funciones de MatLab incluidas en la Image Processing Toolbox para corregir los errores que pueda haber.

En cuanto a la mejora de las imágenes, se corrigen tres aspectos: los bordes, los píxeles interiores a la mosca que sean negros y los píxeles sueltos que hayan quedado por la imagen -en blanco –.

Para el caso de los bordes se utilizará la función *imclearborder*. Esta función elimina todos los píxeles de la imagen que estén conectados con el borde de la misma – en contacto, ya sea directo o mediante otros píxeles –. De esta forma, si hay algún grupo de píxeles sobre el borde que la umbralización no haya podido quitar, los eliminará esta función:

```
salidaCB=imclearborder(ImagvarSegmentada)
```

Para los otros dos casos, se utilizará la función *bwmorph*, la cual realiza operaciones morfológicas sobre imágenes binarias.

La operación *fill* convierte en blanco píxeles negros aislados rodeados de píxeles blancos – como el píxel central de la imagen que se muestra a continuación –.

```
1  1  1
1  0  1
1  1  1
```

```
salidaFill=bwmorph(salidaCB,  'fill')
```

La operación *clean* convierte en negro píxeles blancos aislados rodeados de píxeles negros – como el píxel central de la imagen que se muestra a continuación –.

```
0  0  0
0  1  0
0  0  0
```

```
mascaraF=bwmorph(salidaFill,  'clean')
```

Tras aplicar los tres procesos – seguidos – a las imágenes de la Figura 2.2.1.25 se obtienen los resultados que se muestran en la Figura 2.2.1.26

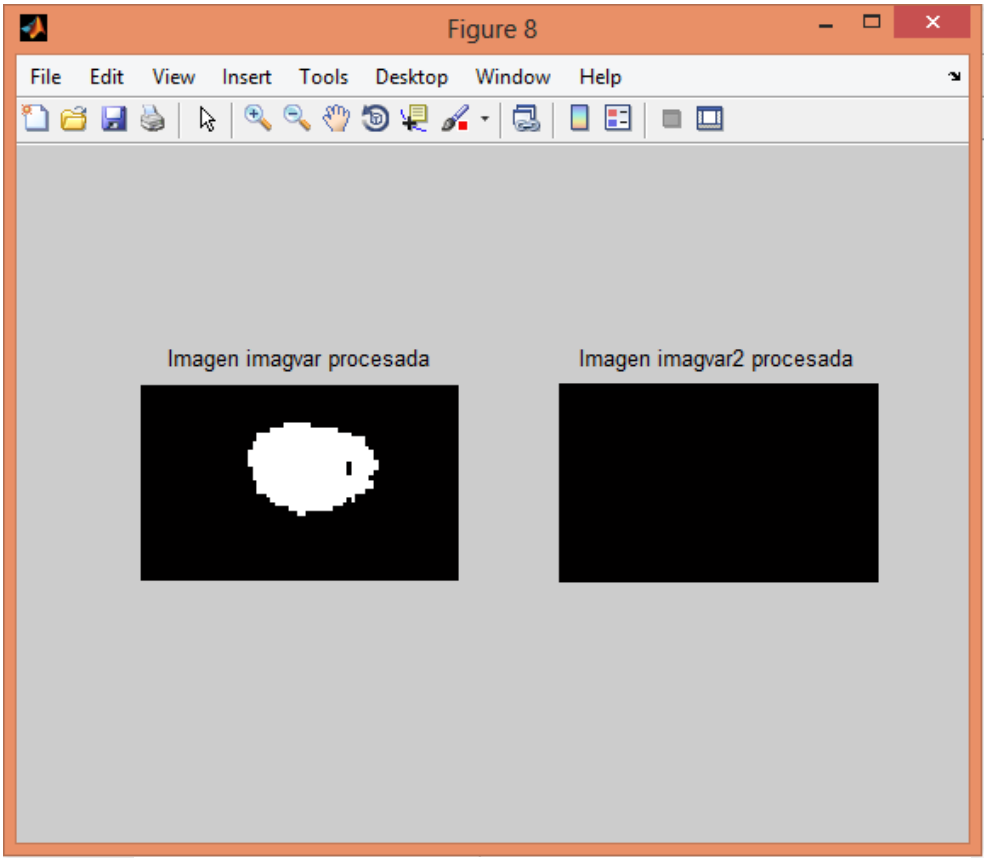

Figura 2.2.1.26: Resultado de la mejora de imágenes.

Llegados a este punto, donde ya se han realizado todas las operaciones necesarias sobre las imágenes, se crea una máscara – *mascaraFinal* – del tamaño de la imagen original, que tenga la máscara de la mosca obtenida mediante todos los procesos anteriores en su esquina correspondiente. Para ello primero se tiene que determinar en cuál de las dos imágenes resultado está la mosca.

En el ejemplo que se ha estado utilizando, como se puede ver en la Figura 2.2.1.26, en una de las imágenes se encuentra la mosca, y en la otra no hay ningún píxel blanco, por lo tanto, para saber cuál de ellas contiene el logotipo basta con saber el número de píxeles blancos en cada una y realizar una comparación entre ambos valores.

Es importante realizar esta comparación, ya que en ocasiones, después de realizar la umbralización, aún quedan píxeles blancos en la imagen que no contiene la mosca, debido a que, por las características del video de origen, dichos píxeles posean un nivel de varianza igual al del logotipo.

Cabe mencionar que puede suceder que si en el vídeo a analizar, además de la mosca hubiera otros objetos en alguna de las zonas de interés – Figura 2.2.1.1 – que se mantengan constantes durante el mismo y poseyeran un número de píxeles superior al de la mosca, al hacer la comparación dicha zona de interés sería erróneamente detectada como contenedora del logotipo. En este supuesto, habría que reducir el tamaño de la zona de interés en cuestión al momento de crear la máscara de filtrado – Figura 2.2.1.1 – de forma que los objetos que interfieren – o parte de ellos – fueran filtrados en el primer paso del procesado y no afectaran al correcto funcionamiento del mimo.

Una vez identificada la mosca, se procede a crear la máscara final, con la máscara de la mosca detectada en su zona correspondiente. Para ello primero se inicializa la variable, creando una matriz de ceros del tamaño de la imagen original, como se muestra a continuación:

```
mascaraFinal=zeros(filas,columnas,3);
```

La máscara tiene que ser tridimensional, ya que la imagen capturada también lo será y para poder multiplicar las dos imágenes, ambas deben tener las mismas dimensiones.

El paso siguiente es asignar a *mascaraFinal* los valores de *mascaraF* – resultado de la corrección de errores sobre la imagen segmentada – en su zona correspondiente, lo cual se hará de la siguiente manera:

```
mascaraFinal(FilaInicioInferior:FilaFinInferior,ColumnaInicioInferior:ColumnaFinInferior,1)=mascaraF;
mascaraFinal(FilaInicioInferior:FilaFinInferior,ColumnaInicioInferior:ColumnaFinInferior,2)=mascaraF;
mascaraFinal(FilaInicioInferior:FilaFinInferior,ColumnaInicioInferior:ColumnaFinInferior,3)=mascaraF;
```

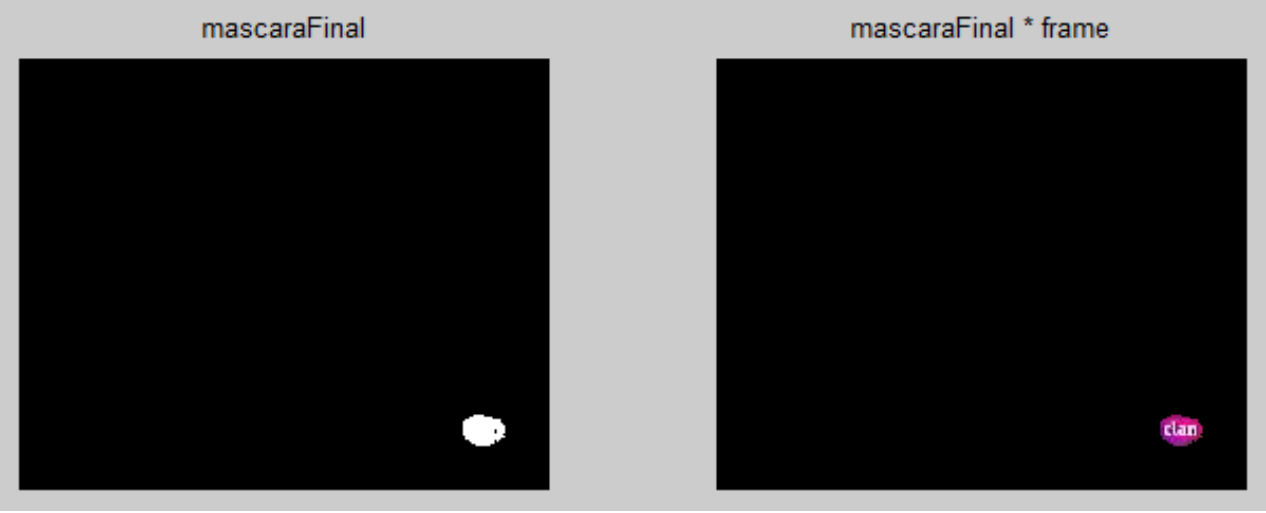

Figura 2.2.1.27: Resultado del procesado

Una vez realizada la asignación, se obtiene la *mascaraFinal* y se genera la imagen referencia multiplicando la máscara obtenida por un frame de video con la mosca – ambas se muestran en la Figura 2.2.1.27 –. Las dos imágenes se graban utilizando la función *imwrite*, de forma que están disponibles para ser utilizadas por el procesado de búsqueda de la mosca.

2.2.2 Algoritmo de detección de la mosca

El objetivo de este procesado es el de determinar si en la imagen que se está analizando se encuentra el logotipo de la cadena emisora. Para ello, nos valdremos de las dos imágenes obtenidas anteriormente: la máscara y la imagen de referencia. Con la máscara filtraremos la imagen capturada, de modo que solamente nos quedemos con la zona de interés. La imagen de referencia será utilizada para comprobar, mediante una comparación, si los píxeles en la zona de la mosca pertenecen a ésta realmente o son parte del vídeo emitido.

Este procesado se basa en las diferencias entre la imagen de referencia y la imagen capturada. Como la imagen de referencia fue obtenida del vídeo con el que se generó la máscara, podemos decir con total seguridad que en dicha imagen se encuentra la mosca, de esta forma, al calcular la diferencia entre ambas imágenes, si ésta es relativamente pequeña, podemos decir que el logotipo se encuentra en la imagen capturada.

Los procesos descritos a continuación se aplican sobre cada una de las imágenes capturadas.

En el primer paso del procesado, utilizamos la máscara adaptada al logotipo para filtrar la imagen a analizar, así solamente conservamos la información de la zona de la mosca, por lo tanto, hemos de cargar la máscara en una variable para poder acceder a ella, y convertirla al formato en que nos llegan las imágenes adquiridas:

```
mascara=imread('mascara.tiff');
mascara=double(mascara/255);
```

También cargaremos la imagen de referencia en una variable:

```
imagenMosca=imread('imXmascara.tiff');
```

En este procesado, trabajaremos con las imágenes en escala de grises, y como la imagen de referencia se encuentra en RGB, el siguiente paso es convertirla al citado espacio de color de forma que podamos operar con ella:

```
imagenMosca=rgb2gray(imagenMosca);
```

Al igual que hicimos con la máscara, hemos de convertir la imagen de referencia al formato en el cual estarán las imágenes procedentes de la etapa de adquisición:

```
imagenMosca=double(imagenMosca)/255
```

Tras realizar las operaciones descritas, obtenemos la imagen de referencia que se muestra en la Figura 2.2.2.1.

Figura 2.2.2.1: Imagen de referencia

Una vez tenemos la imagen de referencia en escala de grises podemos empezar a trabajar con las imágenes capturadas. El primer paso tras capturar una imagen es filtrarla con la máscara, de forma que solamente nos quedemos con la información presente en la zona exacta en la que debe estar la mosca, lo cual conseguimos multiplicando el frame capturado por la máscara.

*imagen=frame.*mascara*

Al multiplicar las imágenes hay que poner un punto antes del operador * para que las matrices se multipliquen elemento a elemento, o sea que cada píxel del frame capturado se multiplica por su píxel correspondiente de la máscara.

Figura 2.2.2.2: Imagen filtrada con la máscara.

Ahora que tenemos la imagen filtrada, el siguiente paso es convertirla a escala de grises igual que la imagen de referencia, ya que ambas han de estar en el mismo espacio de color para que podamos operar con ellas.

```
imagenG=rgb2gray(imagen)
```

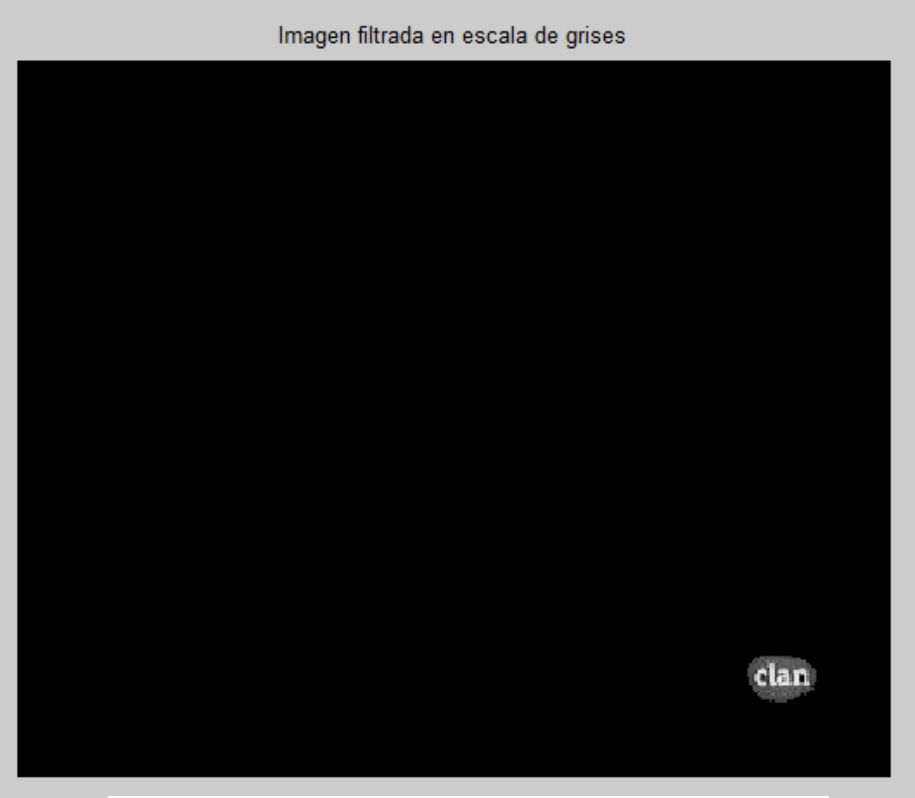

Figura 2.2.2.3: Imagen filtrada en escala de grises.

Observando la Figura 2.2.2.1 y la Figura 2.2.2.3, vemos que la imagen de referencia y la imagen filtrada, ambas en escala de grises, son muy parecidas, incluso podría decirse que son iguales, lo cual significaría que en ambas se encuentra la mosca. Para comprobarlo, realizamos una resta – en valor absoluto – entre ambas imágenes, y obtenemos el siguiente resultado:

Figura 2.2.2.4: Diferencias entre moscas.

En la Figura 2.2.2.4 se puede ver el resultado de la resta de las moscas. Mirando en la imagen *Diferencia entre imágenes* no se aprecian diferencias, en cambio si miramos en el zoom realizado sobre la zona del logotipo podemos ver algunos píxeles en tonos de gris oscuro. Estas diferencias se deben al color de la imagen que está siendo mostrada en el vídeo, que en mayor o menor medida afecta a la mosca.
Al haber obtenido la imagen de diferencias de un frame del vídeo, la claridad de los colores de la mosca se ve afectada por el color del fondo, de modo que si el fondo presenta un color claro, la mosca no se verá tan afectada, pero si el fondo es oscuro los colores del logotipo se verán menos luminosos, algo que no se ve a simple vista pero que se hace evidente en este tipo de análisis. Este fenómeno también se presenta en la mosca de la imagen capturada, de forma que cuando realizamos la resta, no sólo estamos restando los colores de la mosca sino que también se tienen en cuenta los fondos de las imágenes. Es por ello que se recomienda, antes de iniciar el procesado de detección de la mosca, capturar unos segundos de la señal a analizar - unos 35 segundos sería suficiente – y realizar el cálculo de obtención de la máscara – y por tanto de la imagen de referencia – con el vídeo capturado.

A partir de la imagen de diferencias no sólo sabemos cuántos píxeles son diferentes, sino que también sabemos cuán diferentes son, valor que se utiliza como criterio para determinar la presencia de la mosca en la imagen.

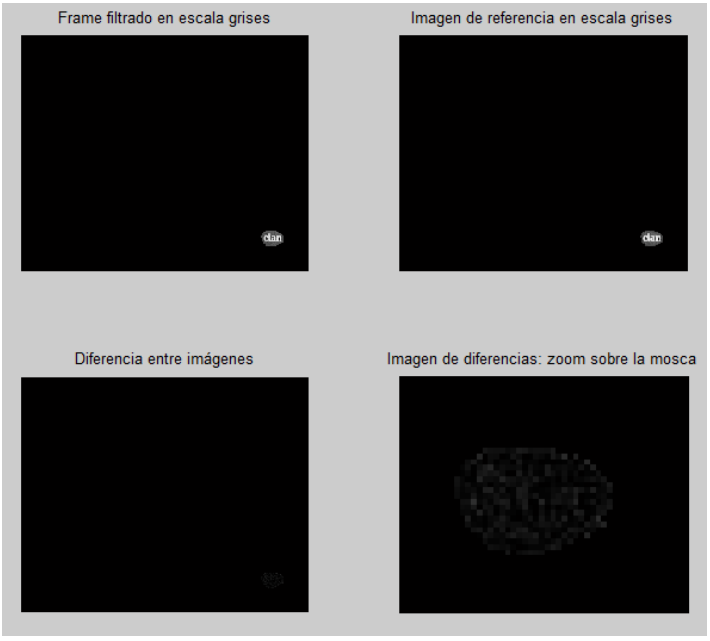

Figura 2.2.2.5: Diferencias entre imágenes con mosca.

En la Figura 2.2.2.5 se muestra una imagen de diferencias en la que a pesar de estar la mosca en las dos imágenes restadas, presenta mayor cantidad de píxeles diferentes y dichas diferencias son mayores que en la imagen de la Figura 2.2.2.4. En la Figura 2.2.2.6 se muestra la imagen de diferencias producto de restar a la imagen de referencia una imagen sin mosca, en la que se puede ver claramente que prácticamente todos los píxeles de la mosca presentan diferencias respecta a sus correspondientes en la imagen de referencia.

Figura 2.2.2.6: Diferencias entre imágenes sin mosca.

Para la detección de la mosca nos basamos en la información obtenida de las imágenes de diferencias y de referencia. En este proceso se tiene en cuenta el nivel de gris de los píxeles de ambas imágenes, ya que a partir de las variaciones de dicho niveles se determinará la presencia o ausencia del logotipo, por ello, realizaremos la suma de los valores presentados por los píxeles de ambas imágenes, con los siguientes comandos:

sumaMosca = sum(sum(imagenMosca))

sumaDif = sum(sum(imagenDiferencias))

Tras realizar dichas operaciones, obtenemos la suma de los niveles de todos los píxeles de cada imagen, a la que llamamos el "valor" de la imagen. Con estos datos se calcula qué porcentaje del valor de la imagen de referencia – *sumaMosca* – representa el valor de la imagen de diferencias – sumaDif –. En otras palabras, calculamos cuánto han variado los píxeles entre ambas moscas. Para decidir si la mosca está en la imagen se establece un umbral para dicho porcentaje, que en un principio es el 25%, el cual representa una variación significativa ya que el valor de las diferencias es de un cuarto del valor original. De todas formas, el umbral es un parámetro configurable, de modo que se puede cambiar de requerirlo las características del vídeo y/o la mosca.

Al realizarse este procesado sobre cada una de las imágenes capturadas, se lleva cuenta de la cantidad de imágenes en las que no se ha encontrado la mosca, pero antes de confirmar la ausencia de la misma, ha de transcurrir un determinado tiempo durante el cual en ninguna de las imágenes analizadas se encontrara el logotipo. Este tiempo es el doble del tiempo de fundido especificado – configurable –, una vez confirmada la ausencia de la mosca se envía una alarma mediante correo electrónico para notificar de la detección del problema. Asimismo, de confirmarse la reaparición del logotipo, se notificará por el mismo medio.

CAPÍTULO 3: Funcionamiento de la aplicación

En este capítulo se explicará el funcionamiento de la aplicación, incluyendo la configuración de los distintos parámetros de funcionamiento, ya sea desde un archivo .txt o desde la propia interfaz gráfica.

3.1 Asignación de parámetros desde un archivo de texto

En este modo de funcionamiento, los valores de los diversos parámetros que se pasan a la aplicación se escribirán en un archivo de texto llamado parametros.txt – sin acento y en minúscula –, el cual debe estar en el mismo directorio que el ejecutable de la aplicación.

Dicho archivo contendrá los siguientes parámetros:

- ***lee_parametros***: indica el modo de funcionamiento de la aplicación, y puede tomar los valores 0 o 1. Con el valor 0 la aplicación se maneja totalmente desde la interfaz gráfica, con el valor 1 los parámetros de funcionamiento se leen del archivo de texto, la interfaz gráfica se abre y se pone en funcionamiento automáticamente.

- ***captura***: indica el modo de captura. Toma el valor 1 para la lectura de un archivo de vídeo de extensión avi y el valor 2 para realizar la captura en tiempo real desde un dispositivo de captura instalado.

- ***capturadora***: indica el identificador del hardware de captura. Si el ordenador posee webcam integrada se le asigna el valor 1 a dicho dispositivo y 2 a la capturadora externa – si la hay –. Si no se dispone de webcam integrada, se asignará el valor 1 a la capturadora externa.

- ***formato***: especifica el formato en el que se realizará la captura desde el hardware especificado en *capturadora*. Ha de introducirse el nombre completo del formato, que será del tipo YUY2_???x??? o YUY2_????x???, por ejemplo YUY2_320x240 o YUY2_1280x720. El usuario debe saber de antemano el formato a utilizar de entre los disponibles en el dispositivo.

- ***procesado***: indica el procesado a llevar a cabo. Puede tomar los valores 1 para realizar el procesado de detección de la imagen parada, 2 para realizar el procesado de detección de la mosca o 3 para realizar ambos simultáneamente.

- ***tfundido***: indica el tiempo de fundido a utilizar tanto para confirmación de la imagen parada, como para la confirmación de la ausencia de la mosca. Se especifica en segundos.

- ***ninif***: indica el número del frame en el que se iniciará el procesado elegido cuando la captura se haga desde un archivo de vídeo, Debe ser siempre mayor que 1 y menor que el número total de frames del vídeo a analizar.

- ***nfinf***: indica el número del frame en el que finalizará el procesado elegido cuando la captura se haga desde un archivo de vídeo, Debe ser siempre mayor que *ninif* y como máximo igual al número total de frames del vídeo a analizar.

- ***intervalo***: indica el tiempo que transcurrirá entre la captura de dos frames – seguidos – para realizar el procesado de detección de la imagen parada, cuando la captura se haga desde un archivo de vídeo. Se especifica en milisegundos. Si no se desea indicar un intervalo – que se cojan los frames consecutivos – ha de asignársele valor 0, aunque dicha opción no se recomienda ya que disminuiría la eficacia del procesado.

- ***tam_media***: indica el tamaño del vector que se utilizará para el cálculo del umbral del procesado de detección de la imagen parada. Su valor por defecto es 70.
- ***nombre_video***: cuando la captura se haga desde un archivo de vídeo, indica el nombre del vídeo a utilizar, incluida la extensión – .avi –. Es imprescindible que el nombre del vídeo no contenga ningún espacio.

- ***ruta_video***: cuando la captura se haga desde un archivo de vídeo, indica la ruta al vídeo a analizar. Es imprescindible que la ruta no contenga ningún espacio, y que finalice con el carácter ****.

- ***direccion_origen***: indica la dirección de correo electrónico desde la cual se enviarán las diversas notificaciones. Debe ser del servidor Gmail. Si no se desea enviar las alarmas, asignar al parámetro el valor 0.

- ***contraseña***: especifica la contraseña de la cuenta *direccion_origen*. Es imprescindible para la configuración de la función que envía los correos – Gmail requiere autenticación –. Si no se desea enviar las alarmas, asignar al parámetro el valor 0.

- ***direccion_destino***: especifica la dirección de correo electrónico a la que se enviarán las notificaciones de las alarmas generadas. Si no se desea enviar las alarmas, asignar al parámetro el valor 0.

- ***video_patron***: nombre del vídeo a utilizar en el procesado de obtención de la máscara de la mosca. Se requiere de la máscara para ejecutar el procesado de detección de la mosca. Es imprescindible que el nombre del vídeo no contenga ningún espacio.

- ***ruta_vpatron***: ruta al vídeo a utilizar en el procesado de obtención de la máscara de la mosca. Es imprescindible que la ruta no contenga ningún espacio, y que finalice con el carácter ****.

- ***filas_ini***: indica la fila en la que empieza la máscara de la esquina inferior derecha de la imagen en el procesado de obtención de la máscara de la mosca. Se especifica el valor porcentual respecto del número total de filas de la imagen.

- ***filas_fin***: indica la fila en la que termina la máscara de la esquina inferior derecha de la imagen en el procesado de obtención de la máscara de la mosca. Se especifica el valor porcentual respecto del número total de filas de la imagen.

- ***col_ini***: indica la columna en la que empieza la máscara de la esquina inferior derecha de la imagen en el procesado de obtención de la máscara de la mosca. Se especifica el valor porcentual respecto del número total de columnas de la imagen.

- ***col_fin***: indica la columna en la que termina la máscara de la esquina inferior derecha de la imagen en el procesado de obtención de la máscara de la mosca. Se especifica el valor porcentual respecto del número total de columnas de la imagen.

- ***filas2_ini***: indica la fila en la que empieza la máscara de la esquina superior derecha de la imagen en el procesado de obtención de la máscara de la mosca. Se especifica el valor porcentual respecto del número total de filas de la imagen.

- ***filas2_fin***: indica la fila en la que termina la máscara de la esquina superior derecha de la imagen en el procesado de obtención de la máscara de la mosca. Se especifica el valor porcentual respecto del número total de filas de la imagen.

- ***col2_ini***: indica la columna en la que empieza la máscara de la esquina superior derecha de la imagen en el procesado de obtención de la máscara de la mosca. Se especifica el valor porcentual respecto del número total de columnas de la imagen.

- ***col2_fin***: indica la columna en la que termina la máscara de la esquina superior derecha de la imagen en el procesado de obtención de la máscara de la mosca. Se especifica el valor porcentual respecto del número total de columnas de la imagen.

- ***segundos_ini***: cuando se ejecuta el procesado de obtención de la máscara de la mosca, indica el segundo del vídeo patrón en el que se inicia el análisis del mismo.

- ***segundos_fin***: cuando se ejecuta el procesado de obtención de la máscara de la mosca, indica el segundo del vídeo patrón en el que finaliza el análisis del mismo. Este número no puede ser mayor que la duración total del vídeo – en segundos – menos 3. Por ejemplo, si el vídeo patrón tiene una duración total de 50 segundos, el valor de *segundos_fin* no debe ser mayor que 47.

Es imprescindible que todos los campos del archivo "parametros.txt" contengan un valor para su correcta lectura, así como que no existan espacios en blanco en los nombres de los vídeos a utilizar y sus rutas correspondientes, de otra forma, el archivo no se leerá correctamente y la aplicación no funcionará. Todo lo que esté escrito después de un carácter % será ignorado al momento de la lectura del archivo – se considera comentario –.

Al ejecutar la aplicación, se leerá el archivo de parámetros, y si el modo de funcionamiento – indicado en *lee_parámetros* – es el de lectura de parámetros del fichero de texto, la interfaz gráfica se abrirá y se pondrá en funcionamiento de acuerdo a los parámetros leídos en el .txt.
Si se indica que realice el procesado de detección de la mosca – ya sea solo o junto con el procesado de detección de la imagen parada –, en primer lugar se ejecutará el procesado de obtención de la máscara. El tiempo que tarde éste en ejecutarse dependerá tanto del formato del vídeo a analizar como de los parámetros configurados, tales como el tiempo de vídeo a procesar y los tamaños de las máscaras – determinados por los valores de inicio y fin de filas y columnas especificados –.

Cuando se haya terminado el procesado seleccionado o se lo haya parado intencionadamente, la aplicación se controlará totalmente desde la interfaz gráfica, de modo que si se quiere volver a ejecutar desde el fichero de parámetros deberá cerrarse, modificar los parámetros – si es necesario – y volver a ejecutar la aplicación.

A continuación, se explicará el funcionamiento de la interfaz gráfica.

3.2 Funcionamiento de la interfaz gráfica

Al ejecutar la aplicación para trabajar en la interfaz gráfica – *leer_parametros* vale 0 –, ésta se abre y nos encontramos con la pantalla que se muestra en la Figura 3.2.1

Como ya se enseñó en la Figura 1.3.1, la interfaz está compuesta por tres zonas: Barra de Herramientas, Área de Configuración de Parámetros y Área de Procesado.

La barra de herramientas cuenta con las herramientas básicas de MatLab – *Print Figure*, *Zoom In*, *Zoom Out* y *Data Cursor* – y además algunas utilidades propias de la aplicación, que de describen a continuación – en orden –.

- **Configurar correo electrónico:** abre una ventana de diálogo en la que se introducen la dirección de origen, contraseña y dirección de destino. Se mantiene igual que en la versión anterior de la aplicación.

- **Capturar vídeo:** elige la captura como método de adquisición y aparece en el área de configuración un panel donde se puede elegir el dispositivo de captura de entre los instalados en el equipo. Una vez seleccionado el hardware, aparece un panel que muestra los diferentes formatos disponibles en dicho dispositivo, y se habilita el botón para cargar el vídeo patrón – antes deshabilitado – y el botón de inicio del procesado de imagen parada. Si se va a realizar el procesado de detección de la imagen parada, se debe seleccionar el formato y ya se puede iniciar el procesado. Si se quiere realizar otro procesado, primero ha de generarse la máscara de la mosca. Para ello, primero ha de cargarse el vídeo patrón. El resto del procedimiento se explicará más adelante.

- **Cargar video .avi:** elige la lectura de un vídeo de extensión avi como método de captura. Se abre el explorador de archivos para seleccionar el vídeo deseado y aparecen en el área de configuración los diferentes parámetros a configurar y la información del vídeo seleccionado. Al pulsar este botón y seleccionar el vídeo se activan los botones de para cargar el vídeo patrón e iniciar el procesado de detección de la imagen parada.

- **Cargar vídeo patrón:** al pulsar este botón se abre el explorador de archivos para seleccionar el vídeo patrón a utilizar. Al cargar el vídeo, su nombre aparece en el panel “Parámetros” del área de configuración. Inicialmente este botón está inhabilitado hasta que se seleccione el método de adquisición deseado. Tras seleccionar el vídeo patrón se habilita el botón “Configurar mosca”.

- **Guardar parámetros:** genera un archivo de texto en el que se guardan todos los parámetros de configuración utilizados – con el mismo formato que *parametros.txt* –. Al pulsar este botón se abre el explorador de archivos, en el cual habrá que seleccionar un nombre y una ubicación para el archivo que se va a crear. Si se le asigna un nombre de archivo ya existente en esa ubicación, pide confirmación para reemplazarlo, por lo tanto si se dice que si el archivo anterior desaparecerá. Es recomendable utilizar esta función una vez se haya iniciado el procesado elegido, de forma que todas los parámetros posean su valor correspondiente. El documento generado puede utilizarse de igual manera que *parametros.txt*, aunque para ello debe estar en la misma ubicación que el ejecutable de la aplicación. En la versión anterior de la aplicación no se contaba con esta función.

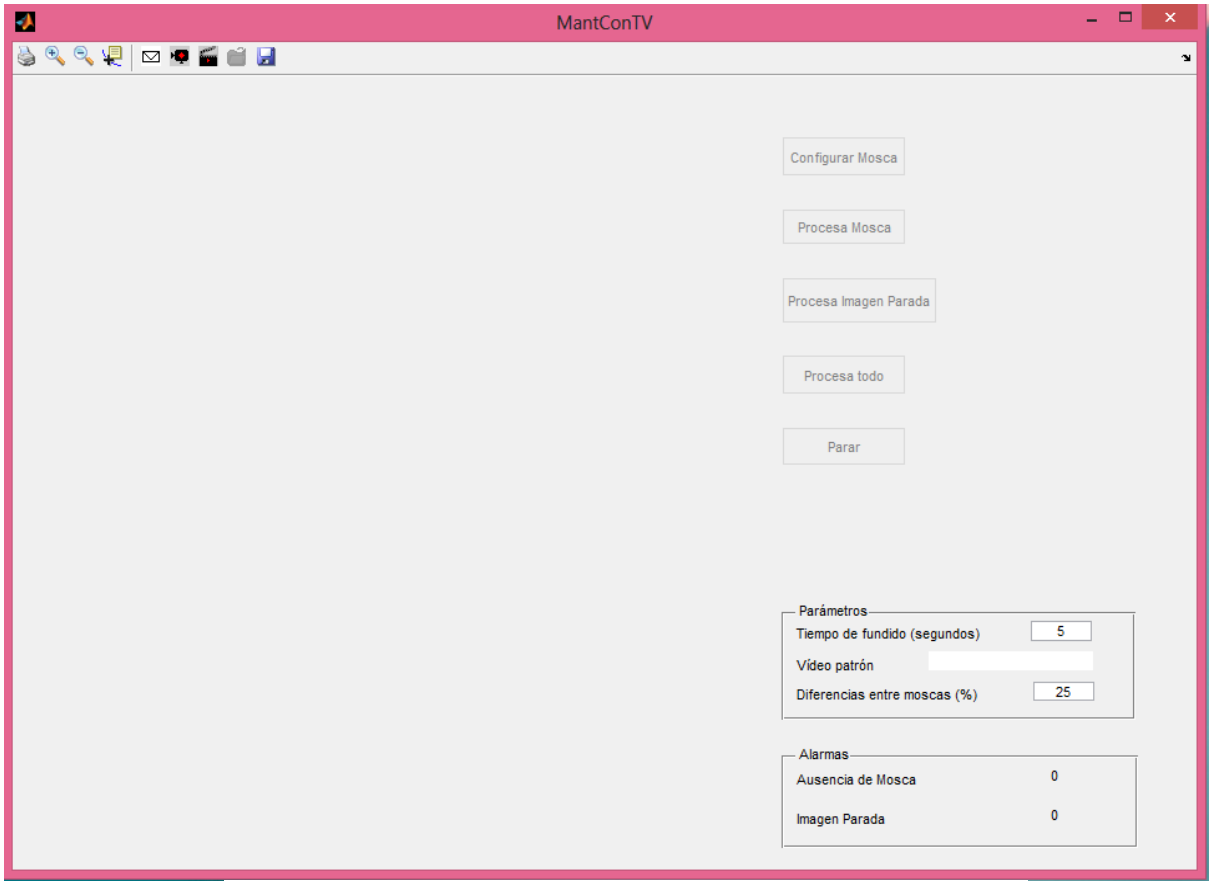

Figura 3.2.1: Imagen de la interfaz gráfica.

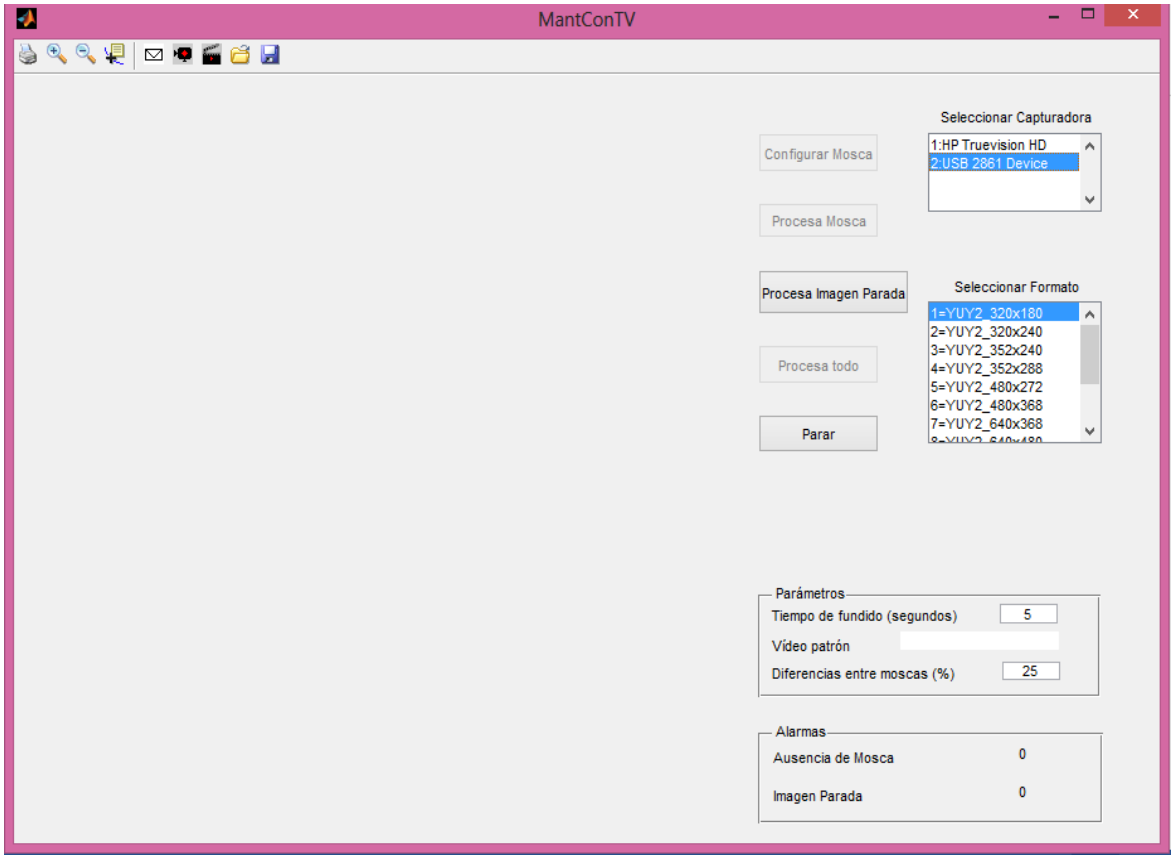

Figura 3.2.2: Configuración de la capturadora

En la Figura 3.2.2 se pueden ver los paneles de configuración de la capturadora, donde se selecciona el hardware a utilizar y el formato. También se pueden ver los paneles *Parámetros*, donde se selecciona el tiempo de fundido y el umbral de diferencias – para el procesado de detección de la mosca – y se muestra el nombre del vídeo patrón a partir del cual se obtiene la máscara del logotipo – en este caso aún no se ha seleccionado –. En el panel alarmas se muestran las alarmas generadas para cada uno de los problemas detectados. La alarma *Ausencia de Mosca* muestra el número de imágenes en las que no se encontró la mosca, y la alarma Imagen Parada indica el número de veces que se ha confirmado la imagen parada. Ambos paneles *Parámetros* y *Alarmas* son comunes a los dos métodos de adquisición de imágenes.

A continuación, en la Figura 3.2.3, se ven los paneles de configuración *Configurar AVI* y *Vídeo AVI*. En el primero se eligen los frames en los que inicia y finaliza el procesado y el intervalo – para el procesado de imagen parada –, mientras que en el segundo se muestra el nombre y el frame rate del vídeo seleccionado.

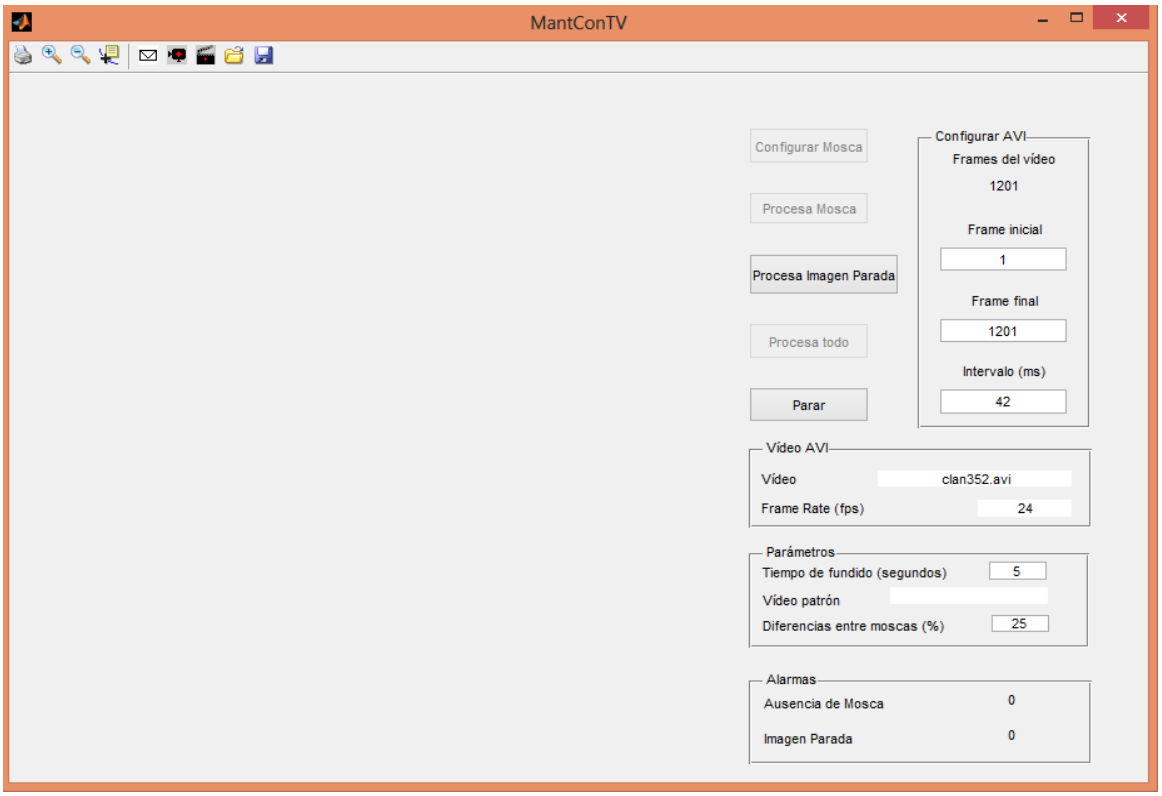

Figura 3.2.3: Configuración de los parámetros del vídeo seleccionado

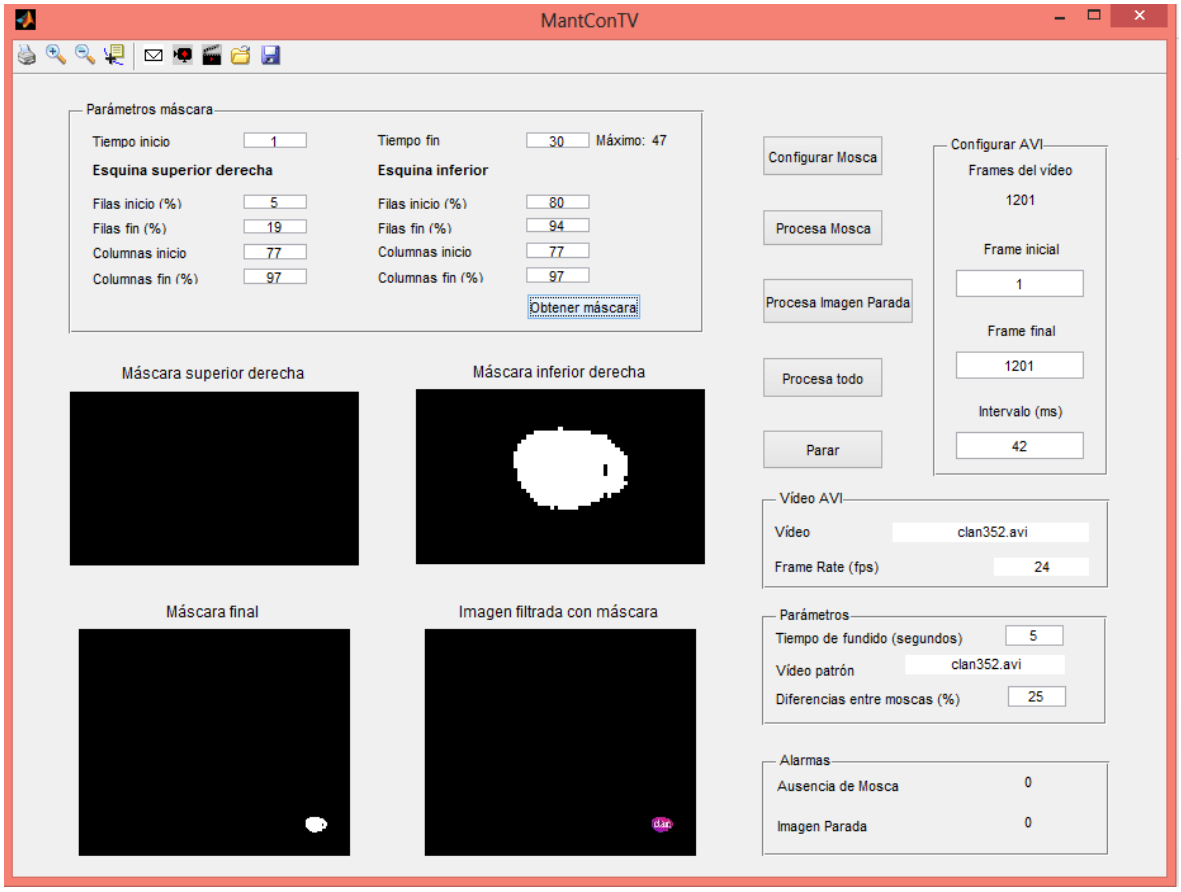

Figura 3.2.4: Obtención de la máscara.

En la Figura 3.2.4, se ve en el área de procesado, la pantalla de configuración de los parámetros para la obtención de la máscara de la mosca. En el panel *Parámetros máscara* se puede seleccionar el tiempo de video a analizar en *Tiempo inicio* y *Tiempo fin* – ambos en segundos – y los parámetros de tamaño de las máscaras del filtrado inicial. Al pulsar *Obtener máscara* se ejecuta el procesado, e inmediatamente se ven los resultados, de forma que si la máscara obtenida no es la esperada se puede ajustar los parámetros y volver a calcularla.

En cuanto al Tiempo fin, se puede ver que a su costado se indica el valor máximo que puede tomar dicho parámetro.

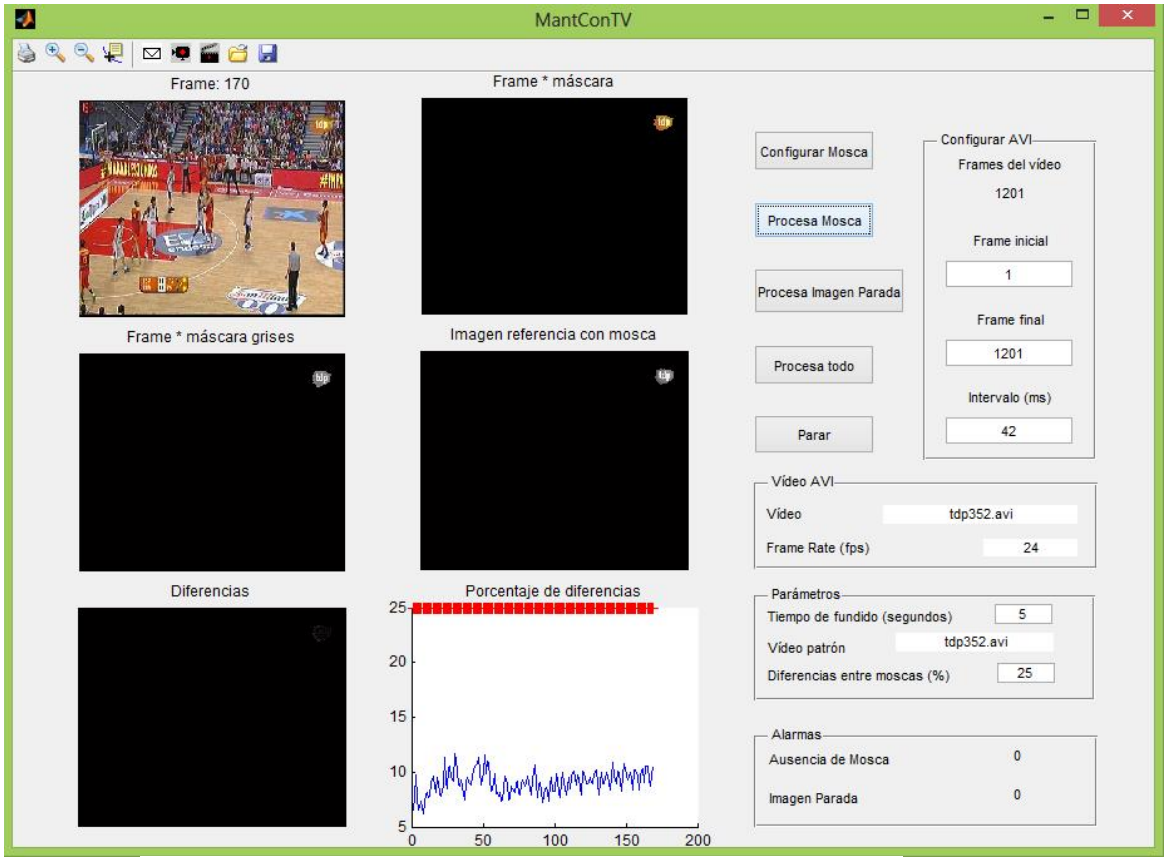

Figura 3.2.5: Procesado de detección de la mosca.

En la Figura 3.2.5 podemos observar el funcionamiento del procesado de detección de la mosca. En el área de procesado se muestra el resultado de las diferentes etapas del proceso: el frame adquirido, éste mismo filtrado con la máscara, tanto en color como en escala de grises, la imagen referencia – en escala de grises también –, la imagen diferencia y la gráfica en la que se observa el resultado de la comparación con el umbral en todos los frames procesados.

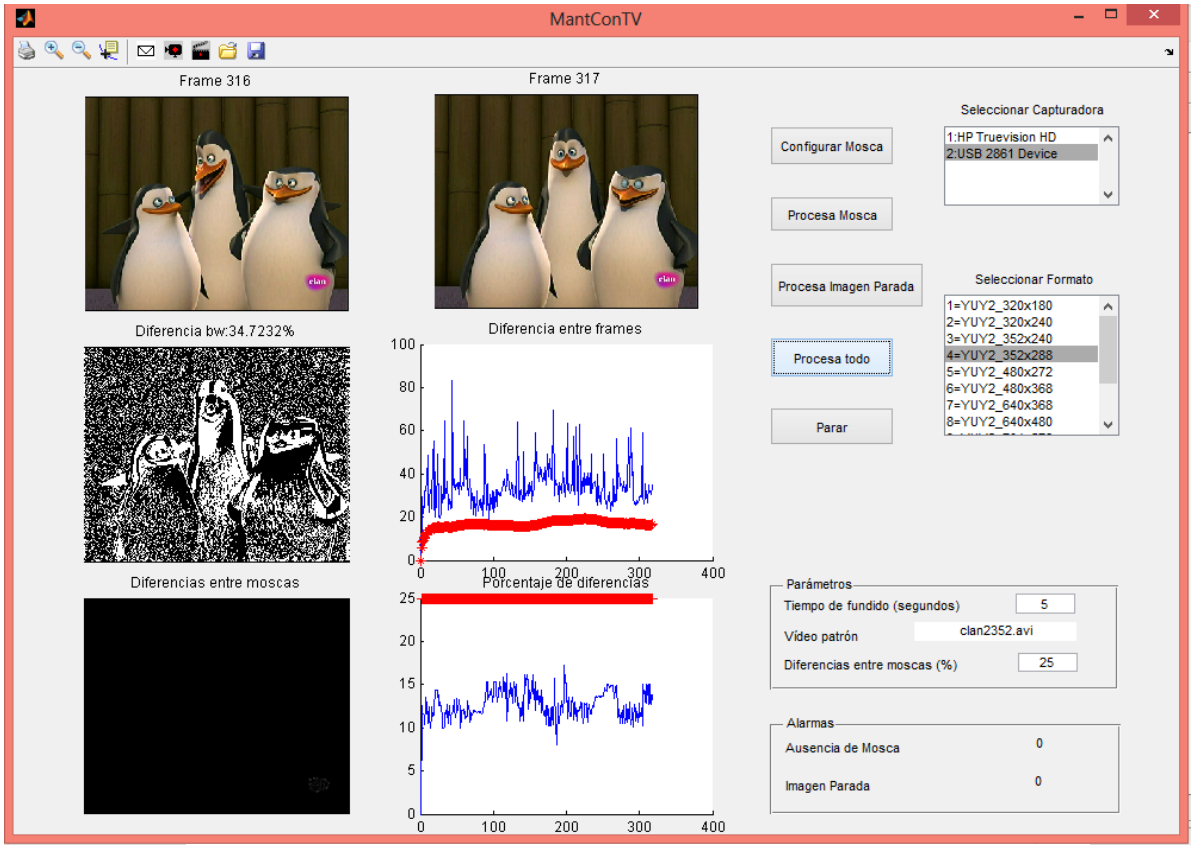

Figura 3.2.6: Procesa todo: imagen parada y detección de la mosca.

En la Figura 3.2.6 se puede ver la ejecución de la función Procesa todo, en la cual se ejecutan ambos procesados uno a continuación del otro. En este caso, los parámetro de funcionamiento fueron configurados en la propia interfaz, por ello se ven en el área de configuración los paneles para la selección de la capturadora y el formato a utilizar. Hay que destacar que, cuando la configuración de la capturadora se realiza desde la interfaz, el formato a utilizar ha de seleccionarse justo antes de iniciar el procesado elegido. En caso de que se quiera detener el procesado y volver a iniciarlo, ha de volver a seleccionarse el formato ya que al hacerlo se envía dicha información a la función que realiza la captura.

En el área de procesado se pueden ver los resultados que se obtienen del mismo, donde se muestran los frames capturados, la imagen de diferencias entre ellos y la gráfica donde se muestra el nivel de diferencias presentado por los frames y el umbral correspondientes al procesado de detección de la imagen parada. También se puede ver la imagen de diferencias entre las moscas y la gráfica a partir de la cual de determinará la presencia o ausencia del logotipo.

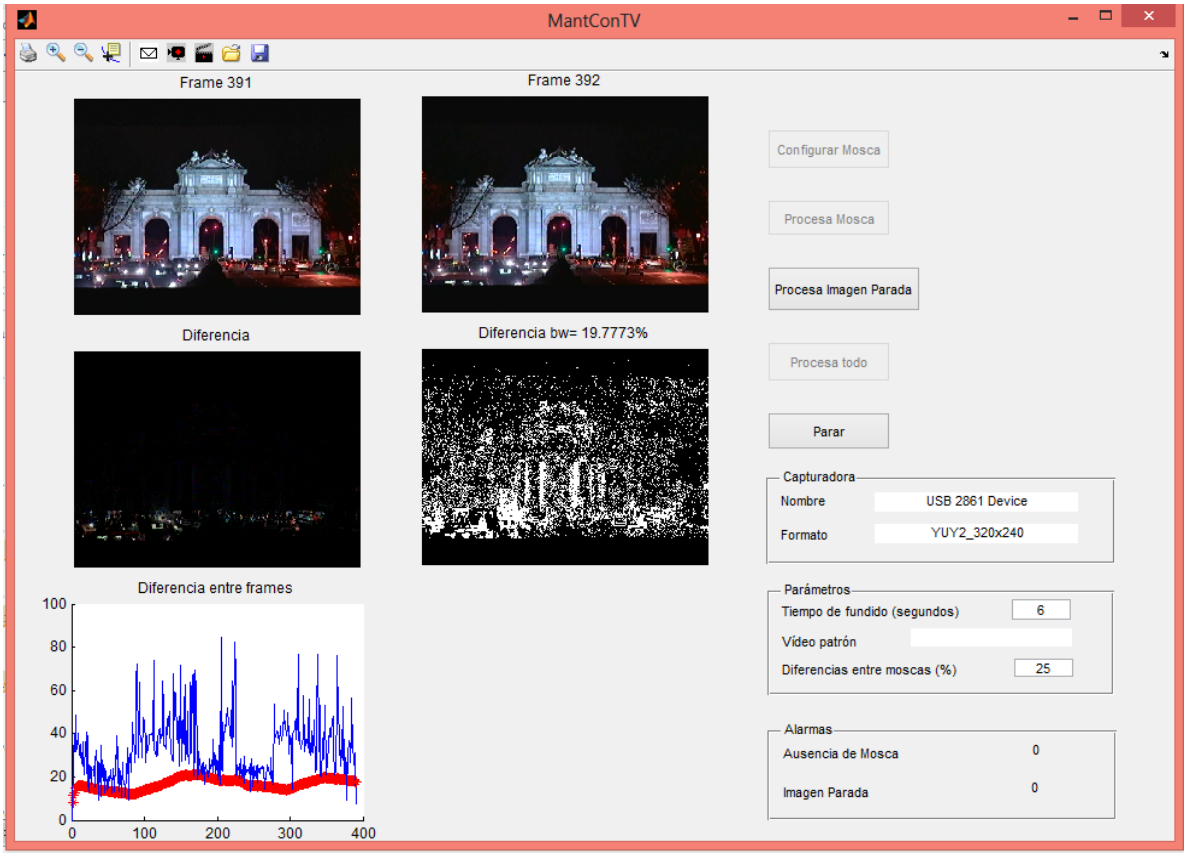

Figura 3.2.7: Procesado de detección de la imagen parada.

En la Figura 3.2.7 se puede ver, en el área de procesado el funcionamiento del procesado de detección de la imagen parada mediante la captura en tiempo real. Para realizar este procesado, se ejecutó la aplicación para que utilizara los parámetros especificados en el archivo de texto, por lo cual ya no aparecen los paneles para seleccionar la capturadora y el formato, sino que se muestra el panel *Capturadora*, en el que se indica el nombre del dispositivo utilizado para la captura y el formato en el cual se está realizando la adquisición de imágenes.

CONCLUSIONES Y LINEAS FUTURAS:

En este trabajo se han desarrollado dos algoritmos de procesado digital de imágenes que realizan las tareas de detección e identificación del logotipo de la cadena emisora a la cual pertenece la señal de vídeo analizada.

En el análisis de un vídeo con dichos algoritmos, primero se realiza la detección de este logotipo o mosca dentro de la imagen, a partir de un vídeo patrón, de forma que se pueda determinar su ubicación y forma. Este aspecto es muy importante, ya que realiza la detección de la mosca de cualquier cadena emisora, independientemente de su forma, tamaño y ubicación, al disponer el procesado de parámetros configurables que permiten adaptar el funcionamiento del algoritmo a las características particulares de cada mosca.

Una vez realizada la detección de la mosca, se analiza la señal de vídeo en busca de la misma. Para este procesado nos valdremos de la información obtenida en el procesado anterior, de forma que se adapta totalmente a las características del logotipo obtenido anteriormente.

Si bien tras realizar dichas modificaciones al software existente [1] se obtiene una aplicación que presenta una mayor versatilidad, se pierde en exactitud respecto del mismo, ya que en aquél los resultados del procesado de identificación de la mosca son del todo eficientes – aunque limitados a una única cadena emisora –, mientras que en el nuevo procesado se contempla la posibilidad de error.

Además, se ha incorporado un nuevo modo de funcionamiento; la configuración de los parámetros a utilizar desde un archivo de texto, lo cual permite ejecutar la aplicación sin necesidad de manipular la interfaz gráfica, a la cual se le añadieron nuevas características adaptadas a los procesados desarrollados.

Por último, se ha implementado la generación de ficheros de registro, tanto del funcionamiento de la aplicación – donde queda registrado cada vez que se inicia un procesado, se detecta algún problema de continuidad y se soluciona el mismo – como de los parámetros utilizados. De esta manera se puede llevar un seguimiento de toda la actividad de la aplicación y su configuración.

Se puede concluir que el trabajo realizado ha dotado a la aplicación de nuevas funcionalidades que la hacen más flexible y por tanto más útil.

Si bien los algoritmos desarrollados cumplen su cometido, y la aplicación funciona correctamente, se proponen las siguientes **líneas futuras**:

- Mejora del algoritmo de identificación de la mosca, de forma que se aprovechen las características específicas del logotipo, como sus componentes de color, a fin de conseguir una identificación más precisa.

- Modificación del algoritmo de detección de la mosca en la imagen y generación de la máscara, para que pueda funcionar en tiempo real, sin necesidad de un vídeo patrón.

- Desarrollo de un procesado que recalcule la máscara cada cierto tiempo, a fin de evitar que el fondo de la imagen interfiera en la identificación de la mosca – o reducir al mínimo las consecuencias dichas interferencias –

- Implementación de un sistema de control del funcionamiento de la aplicación, para que ésta sea reiniciada automáticamente cada cierto tiempo a fin de evitar un colapso del sistema.

Bibliografía:

[1] Proyecto Final de Carrera "*Implementación de sistema de captura y detección de imágenes para el mantenimiento de la continuidad de la señal de TV en la UPV-TV* ". Autora: Gianna Lava Werner

[2] A. Domingo. "Tratamiento Digital de Imágenes". Ed. Anaya, 1994.

[3] R. C González y R. E Woods, "Tratamiento digital de imágenes", Ed. Prentice Hall, 2008.

[4] González R.C y Wintz P, "Procesamiento digital de imágenes", Addison-Wesley, 1996.

[5] Apuntes y transparencias de clase de Tratamiento Digital de Imágenes, impartida por Ignacio Bosch Roig.

Páginas web consultadas

[1] www.mathworks.com

[2] www.youtube.com