ESCUELA SUPERIOR POLITÉCNICA DEL

LITORAL

Facultad de Ingeniería en Electricidad y Computación

“Implementación de Interfaz gráfica para comparación visual de métodos de segmentación y procesamiento de video usando Matlab”

TESINA DE SEMINARIO

Previa a la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentada por:

Xavier San Andrés Lascano

Vicente Julián Franco Pombo

GUAYAQUIL – ECUADOR

AÑO

2011

A G R A D E C I M I E N T O

Agradecemos:

A nuestro generoso Dios, por brindarnos esta maravillosa oportunidad de culminar nuestros estudios superiores.

A nuestras Familias por todo el apoyo y aliento para que no desmayemos en cumplir nuestros anhelos.

A la Ing. Patricia Chávez, por habernos guiado con entusiasmo en este tramo final de nuestra carrera.

D E D I C A T O R I A

A Dios,

A mí querida esposa e hijos,

A mi amada madre, a mi querido papi Vicente (†) y hermanos,

A mis amigos y compañeros de trabajo,

Por su apoyo desinteresado en la consecución

de esta gran meta.

VICENTE JULIAN FRANCO POMBO

A Dios Todopoderoso ante todo,

A mis queridos padres, principalmente a mi madre por darme ese empujón que necesitaba,

A mis adorados hermanos,

A mis familiares, amigos y compañeros de trabajo,

Por su constante apoyo y motivación para cumplir exitosamente esta meta.

XAVIER ERNESTO SAN ANDRES LASCANO

T R I B U N A L D E S U S T E N T A C I Ó N

Ing. Daniel Ochoa
DELEGADO DEL DECANO

Ing. Patricia Chávez
PROFESORA DEL SEMINARIO
DE GRADUACIÓN

D E C L A R A C I Ó N E X P R E S A

"La responsabilidad del contenido de esta Tesina de Grado, nos corresponde exclusivamente; y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITECNICA DEL LITORAL"

(Reglamento de Graduación de la ESPOL)

Vicente Julián Franco Pombo

Xavier San Andrés Lascano

R E S U M E N

El desarrollo de aplicaciones en el campo del Procesamiento de Señales Digitales (PSD) ha crecido muchísimo en los últimos años gracias al avance tecnológico de hoy. Por nuestras computadoras circulan a diario miles de imágenes y videos digitales que nos llegan desde Internet a través de interfaces como el correo electrónico, facebook o el chat. Por esta razón, consideramos muy importante que los tres métodos de segmentación de video e imágenes, desarrollados en esta Tesina de seminario, sean revisados y examinados con cuidado por el lector, con el fin de poder entender su funcionamiento para luego incorporarlos en algunas de las ramas que usan con mayor frecuencia las técnicas de PSD, como son la Medicina, laboratorios científicos, empresas de seguridad, arte y entretenimiento.

Al final de la Tesina exponemos la mayoría de problemas que se nos presentaron durante las pruebas de los métodos y las soluciones que fuimos dando a cada uno de ellos. También analizaremos los resultados obtenidos por cada método de segmentación implementado a los cuales les acompañaremos presentando el resultado de la encuesta de percepción visual de la herramienta de segmentación realizada a 50 personas.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

I N T R O D U C C I Ó N

Antes de empezar a resolver cualquier problema relacionado con la segmentación de video digital, debemos revisar los dos conceptos básicos que se manejaron durante todo el desarrollo del proyecto. Estos conceptos son, imagen Digital y Video Digital.

Una ***imagen digital*** es la representación de una imagen en códigos numéricos interpretados por un computador y mostrados en pantalla en forma de vectores o pixeles. Las imágenes representadas en forma vectorial tienen la ventaja de que estas puede ser escaladas sin perder su resolución, mientras que la representación mediante mapas de bits, formados por pixeles, al aumentarse de tamaño pierden resolución, pero pueden entregar mejores detalles de la imagen.

Un ***video digital*** en cambio es la ejecución ordenada y en secuencia de imágenes digitales de una representación o escenario. Esta sucesión de imágenes se pueden almacenar en varios tipos de formato como AVI, MPG, Real Video, etc. A las imágenes en secuencia de un video digital se las denomina Cuadro o trama, que viene de la traducción del término inglés ***Frame***.

Podemos decir entonces que la segmentación de video no es otra cosa que dividir, alterar o extraer regiones u objetos de interés de un video digital, mediante un algoritmo o programa que se va aplicando a cada uno de los cuadros o tramas que conforman el video.

CAPITULO 1:

1 ANTECEDENTES

1.1 DESCRIPCIÓN DEL PROBLEMA

Realizar un estudio comparativo de algunos de los métodos más empleados de segmentación de imágenes de video.

Existen múltiples formatos de video disponibles actualmente. Sin embargo, en este proyecto, el estudio comparativo de segmentación se lo realizó sobre archivos con formato AVI (con o sin compresión), bajo el sistema operativo Windows 7 y por el programa Matlab.

1.2 OBJETIVOS DEL PROYECTO

En este proyecto definimos tres objetivos que son los siguientes:

1. Estudio y selección de los métodos más empleados de segmentación de video en la actualidad, por su facilidad de implementación y por su utilización.
2. Implementación de los métodos seleccionados en una interface de Matlab (GUI).
3. Comparación visual entre los métodos de segmentación mostrándolos en la interfaz.

El primer objetivo de este proyecto fue investigar y estudiar cuales son los métodos más comunes para realizar segmentación de imágenes en un video digital. La investigación la realizamos en su mayoría en el Internet, en páginas de sitios como *Mathworks.com* y en otros casos a través de publicaciones, revistas y escritos de distintos autores y universidades que están detallados en la sección Bibliografía.

Luego del análisis correspondiente, los métodos seleccionados por nosotros fueron: Segmentación por Color, por Bordes y por Texturas.

El segundo objetivo fue realizar la implementación de los tres métodos seleccionados en una interface, de manera que pudiéramos observar su funcionamiento y la aplicación práctica de cada uno de ellos.

La interface visual fue diseñada con la capacidad de procesar archivos de video existentes, y archivos nuevos generados a partir de la entrada de una o varias cámaras (webcam) conectadas al computador donde corre la aplicación. El aplicativo fue desarrollado con la interfaz gráfica de *Matlab-R2010a.* En el diseño de la interfaz se consideraron un total de 11 salidas, 4 visuales por pantalla y otras 7 en formato AVI, que son archivos generados al final de cada método de segmentación. En las salidas por pantalla podemos reproducir el video original y los videos segmentados; también podemos ver el video cuadro por cuadro en formato de imágenes JPG (no comprimidos) con el fin de realizar el estudio de comparación. Mientras que en las salidas por archivos, tenemos la posibilidad de ejecutar los videos en cualquier reproductor de video estándar y verlo en pantalla completa con mayores detalles.

El tercer objetivo fue realizar una comparación visual entre los métodos de segmentación. Antes de hacer dicha comparación se hicieron pruebas con algunos videos para determinar si las segmentaciones se realizaban exitosamente, dando en ciertos casos algunos resultados no satisfactorios, los cuales se fueron solucionando (se detallan estos casos en el capítulo 5).

Para la comparación visual se procedió a encuestar a 50 personas, las cuales pudieron aprender a usar la herramienta (interfaz), reproducir algunos videos y sus respectivas segmentaciones por cada método. Cada persona encuestada tuvo su método preferido, el más útil (ya sea para su trabajo, estudios científicos, etc.), y

de mejores resultados (segmentación completa de regiones de interés), etc. Estos resultados se pueden observar en el capítulo 5 sección 8.

CAPÍTULO 2:

2. TEORÍA DE SEGMENTACIÓN DE IMÁGENES

La segmentación de imágenes digitales tiene como principal objetivo lograr extraer o aislar regiones u objetos de interés de las imágenes. Para extraer estos datos necesitamos aplicar algoritmos y fórmulas sobre las imágenes que nos permitan analizarla de forma matemática y así tomar el control de la imagen.

2.1 PROCESAMIENTO DE IMÁGENES Y SEGMENTACION DIGITAL

El procesamiento digital de imágenes es el conjunto de técnicas que se aplican a las imágenes digitales con el objetivo de mejorar la calidad o facilitar la búsqueda de información. Entre estas técnicas están el Filtrado, la Ecualización, la Transformación, e incluye también a la segmentación digital.

2.1.1 Segmentación digital

La segmentación digital de un video o imagen se logra cuando dividimos o separamos este, en las partes u objetos que lo conforman. Mientras que la manera en la que logramos esta subdivisión nos indicará el método de segmentación utilizado. Los métodos más comunes de segmentación están relacionados con las variaciones en la imagen de la intensidad de los niveles de escala de grises, las diferencias de textura o colores entre vecindades de pixeles o simplemente cambios drásticos en las intensidades de ciertas regiones de la imagen.

2.1.2 Segmentación de imágenes y segmentación de videos

La diferencia principal es que la imagen es estática, es decir que su forma inicial no cambia, mientras que el video es dinámico, al ser el resultado de una secuencia de imágenes. Esto al momento de realizar la segmentación tiene su

impacto, ya que en una imagen estática los parámetros básicos de segmentación no varían como son las formas, áreas, texturas, colores y las intensidades de la escala de grises; es decir que la vecindad de un pixel siempre es la misma sobre una imagen dada. En un video la cosa es distinta ya que de un cuadro a otro se pueden producir cambios de forma, cambios de áreas, variación de la posición de un objeto, alteración de la intensidad de una escala de grises o de colores.

En el siguiente ejemplo Figura 2.1 podemos ver como en solo 3 cuadros consecutivos de un video tenemos varias situaciones que analizar; en el primer cuadro aparece una niña sobre el lado izquierdo y luego va desapareciendo hasta salir del video. También podemos apreciar el cambio de posición del retrato de la pared entre los cuadros 1 y 3. Adicionalmente el cuadro 2 está borroso o desenfocado (debido al movimiento a gran velocidad de los objetos, en este caso de la cámara), haciendo que la detección de bordes o formas en ese cuadro no sea muy exacto. Y el área de color rojo de la camiseta del joven del cuadro 3 es diferente en cada una de las tramas.

Figura 2. 1: Cuadros consecutivos de un video AVI

2.1.3 Formatos

Entre los formatos de imágenes más empleados tenemos el *JPG.* Los archivos de formato *JPG* toman su nombre del estándar *JPEG* Grupo Conjunto de Expertos en Fotografía (Joint Photographics Expert Group) que es un algoritmo desarrollado para comprimir archivos de imágenes. La característica de los archivos

JPG permiten que se pierda un poco de calidad en la imagen, pero se gana al disminuir el tamaño final del archivo comprimido.

En cambio el formato AVI Audio-Video Intercalados (Audio Video Interleave), es un estándar creado por Microsoft a principios de los 90's para el manejo del flujo de audio y video en sus sistemas operativos.

2.1.4 Videos a color y a escala de grises

La idea principal de segmentar videos, es obtener las ventajas que brinda cada tipo de imagen. Es importante saber que una imagen a color tiene una composición diferente que una en escala de grises y que otra en formato binario. Antes de continuar revisaremos brevemente la composición de cada una para luego entender el alcance de los algoritmos de segmentación sobre ellas.

En el proyecto manejamos 3 tipos de imágenes que son guardadas en formato JPG:

1. RGB (Color Verdadero)
2. GrayScale (Escala de Grises)
3. Binarias (Blanco y Negro)

2.1.5 Estructura de imágenes RGB, Escala de grises y Binarias.

Las imágenes RGB son en realidad un arreglo tridimensional compuesto por 3 matrices Rojo, Verde y Azul; con una profundidad de 24(8x3) bits. Las imágenes RGB no usan una paleta de colores ni índices sino que el color de cada pixel es determinado por la combinación de cada uno de los valores de las intensidades en cada plano. En la figura 2.2 podemos apreciar la estructura de un arreglo RGB.

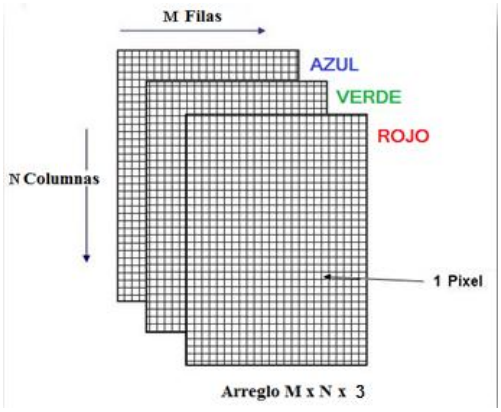


Figura 2. 2: Arreglo RGB MxNx3; http://www.mathworks.com/help/techdoc

Las imágenes en escala de grises en cambio responden a un arreglo unidimensional de tan solo 8 bits de profundidad. La escala de grises tiene un total de 256 intensidades de grises donde el valor de 1 equivale a Negro y 255 equivale a Blanco. En la Figura 2.3 podemos apreciar la escala de Grises y sus distintos valores.

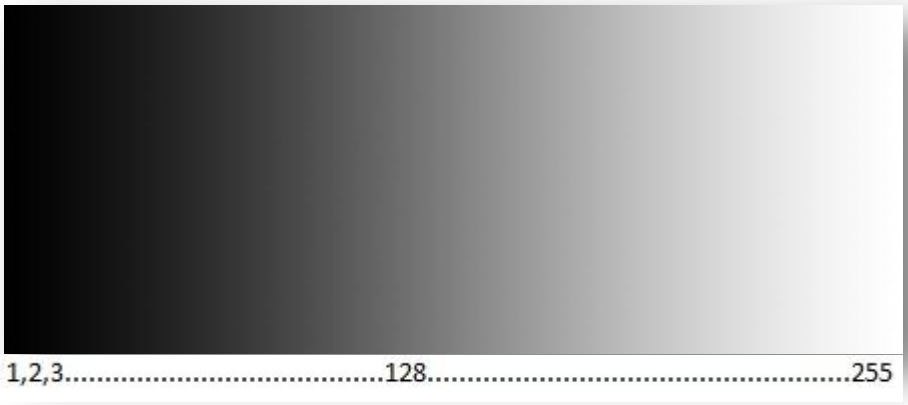


Figura 2. 3: Escala de grises (256 niveles)

Finalmente tenemos las imágenes binarias, conocidas también como blanco y negro. Estas imágenes solo puede tener 2 colores, o blanco o negro. El valor de cada pixel es comparado con un valor umbral que indica que todo lo que este abajo del umbral será negro y todo lo que este arriba de él será blanco. A continuación vemos un ejemplo de una imagen Binaria Fig. 2.4 con diferentes niveles de umbral.

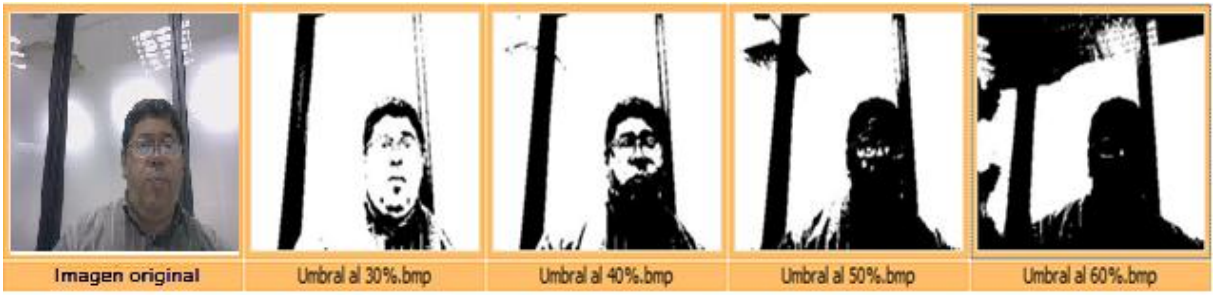


Figura 2. 4: Variación de una imagen binaria con diferentes umbrales

2.2 SEGMENTACIÓN POR COLOR

La segmentación de imágenes en color es una técnica muy utilizada en los medios de comunicación y entretenimiento visuales como son la televisión y el cine. Por segmentación de colores podemos entender los siguientes casos: apagar colores (apagar un color y dejar encendido el resto), o encender colores (encender un color y apagar el resto); estos procedimientos se explicarán más adelante. En nuestro proyecto vamos a trabajar con la segunda alternativa, es decir que de una imagen RGB apagaremos todos los colores y solo encenderemos uno, por ejemplo el color rojo. Otra decisión que tuvimos que tomar por efectos de tiempo, fue la de trabajar solo con los colores primarios RGB (Red-Green-Blue) rojo, verde, azul. Hay que tomar en cuenta que por cada color que uno desee ingresar a la segmentación, el tiempo de procesamiento del video se hará cada vez más pesado y la cantidad de cuadros o tramas que se generan por cada color se multiplican. A continuación mostramos Fig. 2.5 una trama de video RGB original y su imagen segmentada por el color azul.

Figura 2. 5: Cuadro de video segmentado por el color azul.

2.2.1 Encendido de los colores RGB

El algoritmo[1] para encender cualquiera de los colores RGB se basa en tratar de detectar el color primario de cada plano; y una vez detectado no alterarlo. Por ejemplo si la celda que estamos analizando está en el plano rojo, y el valor máximo entre los 3 valores de la celda coincide en el mismo plano, entonces no alteramos su valor y los mantenemos; ver Fig. 2.6. De igual manera procedemos con el análisis de las celdas en los otros planos. Esto es lo que se llama encendido del color.

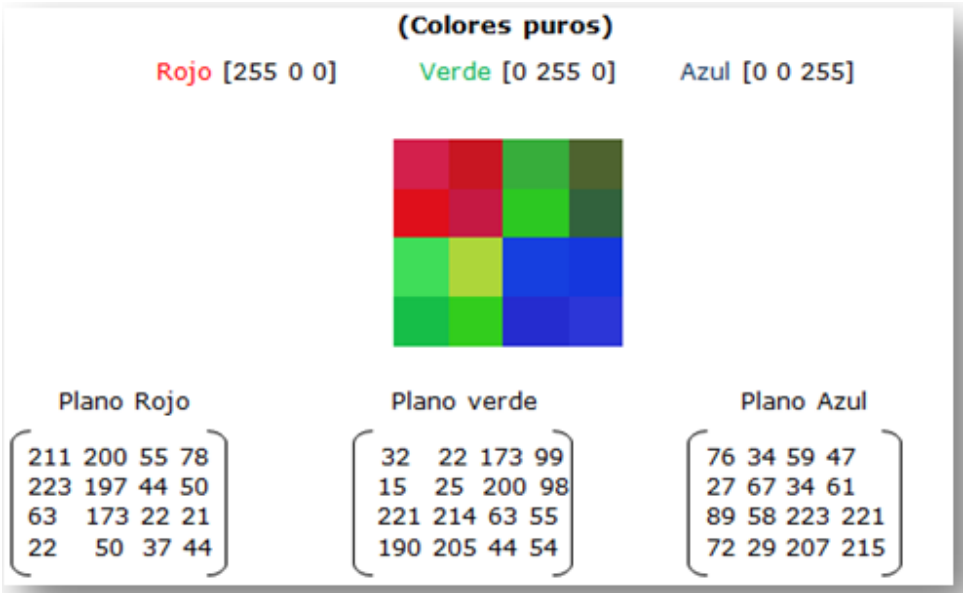


Figura 2. 6: Arreglo RGB de una imagen con sus respectivos planos.

[1] Los códigos y algoritmos usados para realizar la segmentación de video en *Matlab* se muestran completamente en el Anexo B: Código de la solución.

Para que se mantenga encendido el color azul, tomamos el máximo valor de cada celda que coincide con el plano azul y se mantienen los valores; Fig. 2.7. Al resto se aplica la segunda parte del algoritmo que es el apagado de los colores.

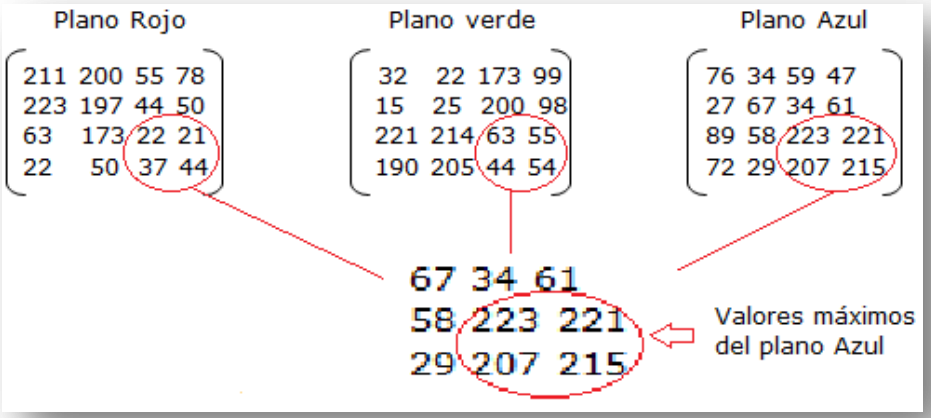


Figura 2. 7: Arreglo RGB; Encendido del color Azul.

2.2.2 Apagado de los colores RGB

Si el máximo valor escogido entre los valores de una celda pertenece a cualquiera de los otros dos planos diferente al que se está evaluando, entonces igualamos todos los valores de la celda a un solo Valor L mediante la ecuación de Brillo de Luminancia (E1) con el fin de normalizar el valor de la celda y que esta quede en escala de grises. A continuación describimos el proceso.

Ecuación de Brillo de Luminancia Normalizada

$$L = 0.30.\ R + 0.59.\ G + 0.11.\ B \quad (E1)$$

La Luminancia representa el brillo que hay en una señal de video. Al mezclarse estás en las proporciones indicadas, el resultado es un color de la escala de grises; con este efecto logramos apagar todos los colores de una imagen RGB a excepción del color original del plano filtrado con la primera parte del algoritmo. En la Fig. 2.8 vemos el resultado de la aplicación de la segmentación por color azul y como quedan las matrices del arreglo RGB.

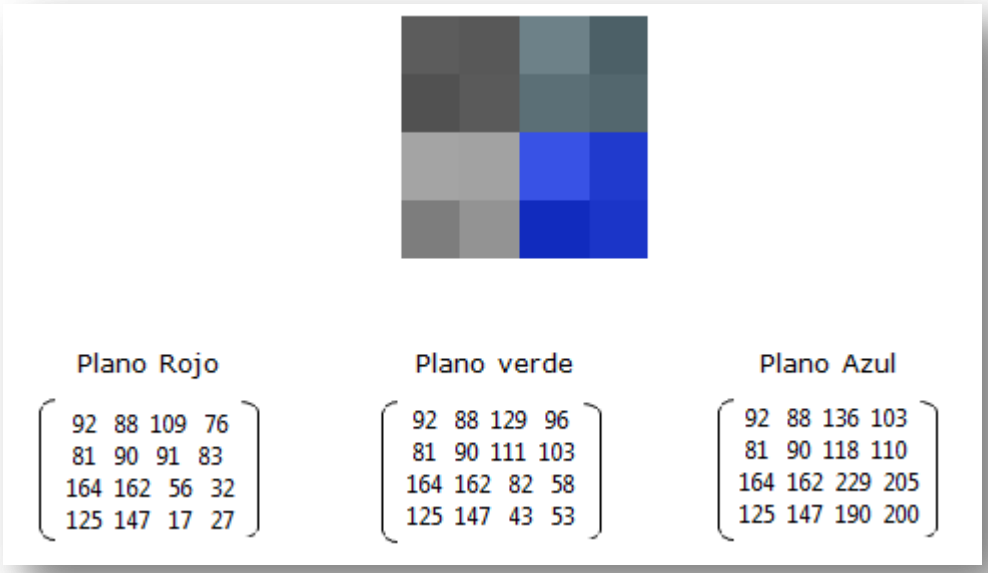


Figura 2. 8: Encendido del color Azul y apagado de los otros colores

2.2.3 Extracción de región de color

La última fase de la segmentación por color luego de efectuado el realzado, tiene que ver con la extracción final de la región de color, es decir aislar en cada cuadro del video las regiones que contengan solo color Rojo, o Verde, o Azul de manera independiente como se ilustra en la Fig. 2.9, y mostrarla como segmentación final.

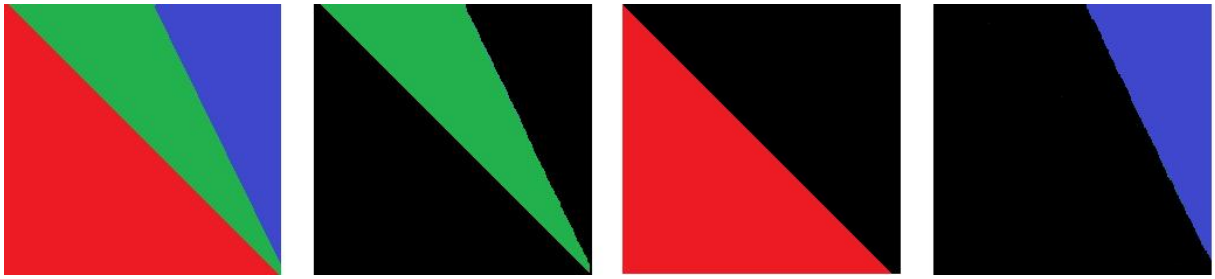


Figura 2. 9: Finalmente cada color es aislado por completo en la imagen.

Para lograr el aislamiento de las regiones en el color deseado, más un color de fondo (puede ser negro o blanco), en necesario binarizar la imagen que viene como producto del realzado.

2.3 SEGMENTACIÓN POR BORDES

Este método es uno de los más conocidos en el ámbito de la segmentación digital. Se basa en la detección de los bordes que pueden existir en una imagen, con el fin de reconocer objetos o regiones. Por borde podemos definir que son las zonas de una imagen en donde se advierte un notable cambio en los valores de la intensidad de dicha zona. Hoy en día, se han desarrollado algunos algoritmos capaces de lograr estas detecciones de bordes, entre los más usados están los algoritmos de Sobel, Prewitt y Canny.

2.3.1 Comparación algoritmos de detección de bordes

Para saber con qué algoritmo de detección de bordes trabajar para continuar con la implementación de la interfaz, fue necesario realizar un estudio y pruebas con cada uno de ellos, con la finalidad de poder escoger el algoritmo que detecte la mayor cantidad de bordes de las imágenes originales, y finalmente ir ajustando la segmentación para que la extracción de bordes sea completa. Primero se verán ciertos conceptos para tener un mayor entendimiento del funcionamiento de los algoritmos.

La forma más común de ver las discontinuidades (variaciones o cambios bruscos) en la escala de grises es pasar una máscara a través de la imagen. Las máscaras son matrices de convolución de 3x3, las cuales pueden detectar líneas en distintas orientaciones (horizontales, verticales o diagonales) (Fig. 2.10) y bordes de la imagen. Estas máscaras varían según el algoritmo que se esté utilizando.

$$\begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix} \begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix} \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$

Figura 2. 10: Máscaras para detectar líneas en distintas orientaciones

Entre las máscaras que más se utilizan está el gradiente. El gradiente es un vector (E2), cuya magnitud (E3) y dirección (E4) están dadas por la máxima variación de intensidad en la escala de grises en el punto evaluado (variación de la función f en el punto (x,y)).

$$\nabla f = \left[\frac{\partial f}{\partial x}, \quad \frac{\partial f}{\partial y}\right] = \left[G_x, G_y\right] \qquad \textbf{(E2)}$$

$$\nabla f_m = \sqrt{G_x^2 + G_y^2} \qquad \textbf{(E3)}$$

$$\alpha(x, y) = \tan^{-1}\left(\frac{G_y}{G_x}\right) \qquad \textbf{(E4)}$$

En una función bidimensional f(x,y), la derivada de dicha función da como resultado el vector gradiente. En la Fig. 2.11 podemos darnos cuenta que sucede cuando en una imagen se detectan líneas claras u obscuras.

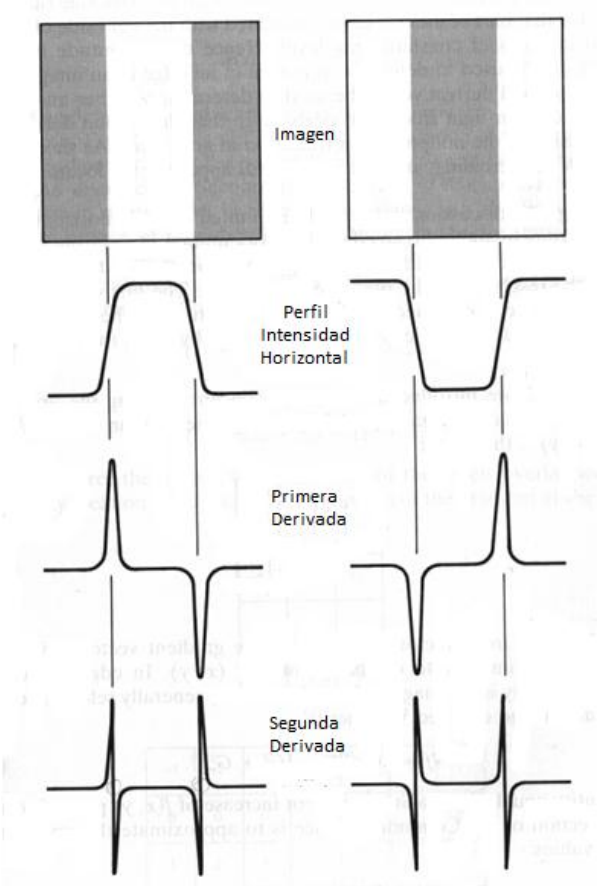


Figura 2. 11: Detección de Bordes en Imágenes

La primera derivada detecta la variación; los máximos coinciden con el punto central del borde. La segunda derivada detecta los cambios en la pendiente y los de la primera derivada; los pasos por cero coinciden con el centro del borde. El cálculo de las derivadas parciales en forma digital se puede realizar de diversas maneras, especificando la máscara más conveniente.

El algoritmo Sobel calcula el gradiente de la intensidad de brillo de cada punto (pixel) dando la dirección del mayor incremento posible (de negro a blanco). El resultado muestra qué tan abruptamente o suavemente cambia una imagen en cada punto analizado, a su vez que tanto un punto determinado representa un borde en la imagen y también la orientación a la que tiende ese borde.

Sobel utiliza dos matrices 3x3 para aplicar convolución a la imagen original, calculando aproximaciones a las derivadas (una para los cambios horizontales y la otra los verticales). En la Fig. 2.12, A es la imagen original, Gx y Gy representan para cada punto las aproximaciones horizontal y vertical de las derivadas de intensidad.

$$\mathbf{G_x} = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G_y} = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

Figura 2. 12: Máscaras que utiliza Algoritmo de Sobel

El algoritmo de Prewitt opera de la misma manera que el de Sobel, con la diferencia que utiliza diferentes matrices de convolución. En la Fig. 2.13 aparecen las matrices que utiliza este algoritmo.

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad y \quad \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

Figura 2. 13: Máscaras que utiliza Algoritmo de Prewitt.

El algoritmo de Canny combina un operador diferencial con un filtro gaussiano, el cual suaviza la imagen (reduce el ruido, detalles y texturas que no interesan); a esta imagen se calcula la gradiente para determinar los pixeles donde se produce máxima variación, se suprimen los pixeles que no son máximos locales en la dirección del gradiente, y se realiza un proceso de doble umbralización, tanto para determinar los pixeles del borde como para eliminar falsos bordes o bordes dobles.

En las siguientes figuras podemos ver los resultados de una detección de bordes mediante los métodos de Sobel, Prewitt y Canny, empezando con la imagen original antes de la segmentación.

Figura 2. 14: Cuadro #1 de un video previo a la segmentación por bordes

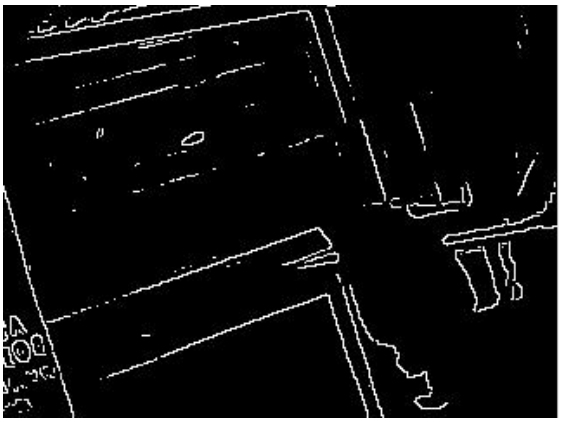


Figura 2. 15: Cuadro #1 luego de aplicado el algoritmo de Sobel

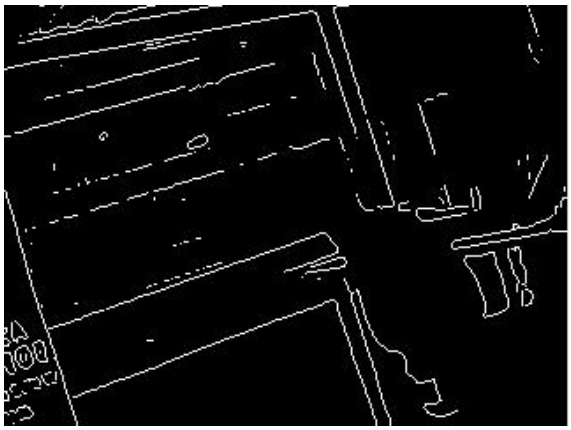


Figura 2. 16: Cuadro #1 luego de aplicado el algoritmo de Prewitt

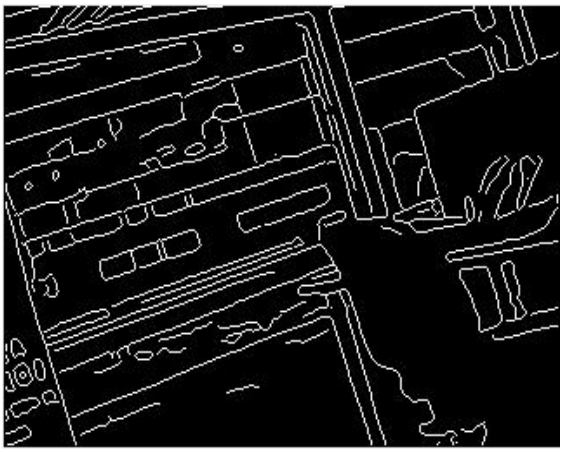


Figura 2. 17: Cuadro #1 luego de aplicado el algoritmo de Canny.

Viendo las imágenes podemos darnos cuenta que el algoritmo que detectó mayor cantidad de bordes fue el de Canny, siendo el de Prewitt el segundo mejor.

La detección de bordes por Canny demoró un poco mas que los de Prewitt y Sobel, y era de esperarse, ya que como se explicó anteriormente implica mayor procesamiento. Sin embargo, tomando en cuenta los recientes resultados, el empleo de un filtro que reduce considerablemente el ruido de la imagen original y la umbralización para desechar falsos bordes, se trabajará con el algoritmo de Canny para la detección de bordes.

2.3.2 Uso del algoritmo Canny en Matlab

La función detección de Bordes (EDGE) de *Matlab,* es la encargada de invocar a los diferentes algoritmos desarrollados para realizar la segmentación; la sintaxis de la función EDGE (E5) es como sigue:

```
BW = edge(I,'canny',thresh)     (E5)
```

Donde (I) corresponde a una imagen en escala de grises o binaria previamente cargada en el área de trabajo, y el término 'thresh' corresponde al umbral (si no se coloca este valor, EDGE elige los valores automáticamente). También podemos cargar una imagen RGB, pero esta debe ser transformada a escala de grises o a binario usando las funciones RGB2GRAY o IM2BW; luego de eso colocamos la función EDGE, el nombre del algoritmo con el cual queremos realizar la detección de bordes, para nuestro caso Canny, y el resultado se almacena en una variable (BW) de clase lógica. Ver Fig. 2.18.

Figura 2. 18: Imagen Lena en escala de grises y luego filtrada por Canny.

2.3.3 Uso de máscaras de binarias para visualizar regiones de las imágenes con detección de bordes.

Con el fin de poder visualizar una región o un determinado sector de la imagen procesada por el algoritmo de Canny, hemos considerado el uso de máscaras binarias manuales definidas directamente sobre la imagen original en el cuadro 1 del video que se esté procesando, como máscara de región. Ver fig. 2.19

Figura 2. 19: Izquierda mascara binaria, derecha máscara aplicada a la imagen.

El objetivo de usar máscaras binarias de región, que no son otra cosa que matrices del mismo tamaño que la imagen original pero llena de ceros, que representan el color negro y una sección de unos que forman el color blanco de la máscara; y que luego mediante el uso de operadores lógicos como las funciones

AND, OR, XOR, las podemos combinar con las imágenes originales y así obtener determinadas regiones de los equipos según la forma de la máscara. Ver fig. 2.20.

Figura 2. 20: Izquierda detección de bordes, derecha con máscara XOR aplicada.

El operador lógico disyunción exclusiva (XOR) o también llamado "o exclusivo", es un tipo de disyunción lógica de dos operandos que es verdad si solo un operando es verdad pero no ambos; en este caso, un operando sería la máscara binaria de una región (región del video que queremos segmentar), y el otro sería la imagen original.

2.4 SEGMENTACIÓN POR TEXTURAS

El análisis de texturas se refiere a la capacidad de poder distinguir las diversas regiones de una imagen por su contenido de textura. Por contenido de textura entendemos términos tales como áspero, sedoso, huecos, rugoso, luminoso, etc. Entendiendo un poco más podemos decir que la rugosidad o los huecos por ejemplo se refieren a las variaciones en los valores de brillo o niveles de gris de la imagen. Ver Fig. 2.21.

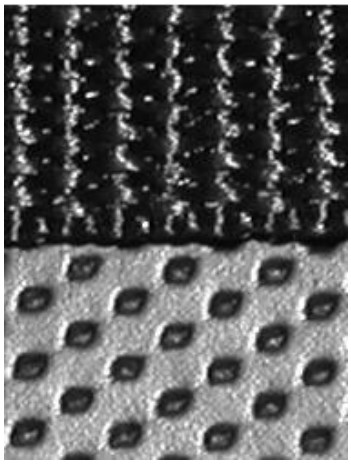


Figura 2. 21: Imagen típica para el análisis de texturas.

MATLAB incluye varios filtros para realizar la detección y filtrado de texturas. Los filtros existentes para detectar texturas son:

- STDFILT
- RANGEFILT
- ENTROPYFILT

2.4.1 Comparación de filtros para detectar texturas

El filtro STDFILT calcula la desviación estándar local de una imagen. Este filtro define la vecindad 3x3 alrededor del pixel de interés y calcula la desviación de dicha vecindad para determinar el valor del pixel en la imagen resultante.

El filtro RANGEFILT calcula el rango local de una imagen. Este filtro identifica vecindades 3x3 y diferentes formas y tamaños alrededor el pixel evaluado, determinando los valores de los pixeles en la imagen resultante.

El filtro ENTROPYFILT calcula la entropía local de una imagen en escala de grises. Este filtro define la vecindad 9x9 alrededor del pixel de interés, calcula la entropía de dicha vecindad y asigna ese valor al pixel de la imagen resultante. La entropía es una medida estadística de aleatoriedad.

En la Fig. 2.22 podemos apreciar el empleo de cada filtro sobre la imagen original.

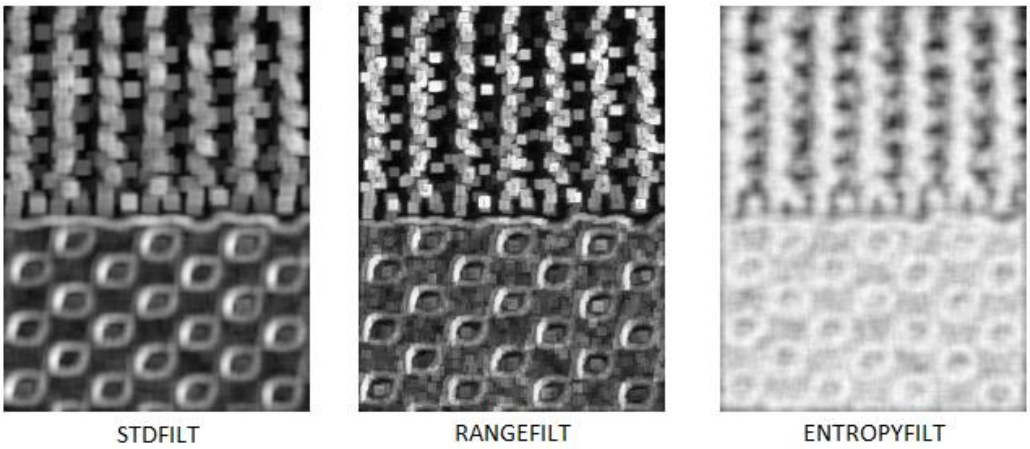


Figura 2. 22: Imagen después de diferentes filtrados de texturas.

Se consideró trabajar con el filtro STDFILT ya que pudo identificar con mayor precisión los cambios de textura en las vecindades de cada pixel de la imagen original. Como resultado, podemos notar que la imagen filtrada está más suavizada (menor cantidad de ruido) comparada a las imágenes de los otros dos filtros.

2.4.2 Uso del Filtro STDFILT (Desviación estándar local de la imagen)

Este filtro calcula la desviación estándar local de todos los valores dentro de la vecindad de un pixel que se van a segmentar en una imagen dada. Por vecindad de un pixel se entienden todos los pixeles que rodean al pixel que está siendo evaluado. Mientras más puntos de conectividad tengan la vecindad de un pixel, más exacto será el cálculo de su análisis de textura (ver Fig. 2.23). A continuación presentamos diferentes tamaños de vecindades de un pixel.

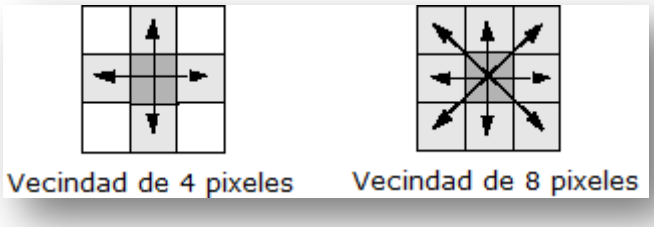


Figura 2. 23: Ejemplos de vecindades de pixeles.

La aplicación del filtro estándar es bastante sencilla (Ecuación E6). Primero debemos leer una imagen (I) en formato de escala de grises y luego le aplicamos el filtro con la siguiente sintaxis:

```
J = stdfilt(I)     (E6)
```

El resultado es un arreglo J, donde cada píxel de salida contiene la desviación típica de la zona de 3 por 3 alrededor del píxel correspondiente a la imagen de entrada I.

Con el filtro STDFILT podemos realizar dos acciones que son básicas para poder realizar una segmentación por texturas, la primera es lograr marcar los límites de las diversas regiones de texturas de la imagen y la segunda acción es poder segmentar una región determinada, a través de una máscara lógica. En la figura 2.24 podemos apreciar estos dos efectos.

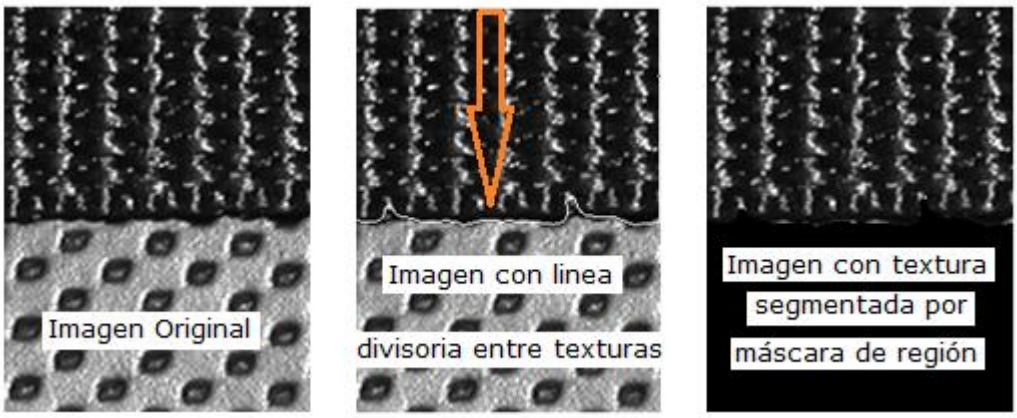


Figura 2. 24: Ejemplos de aplicación del filtro STDFILT en una imagen con texturas.

2.5 ECUALIZACION DE IMÁGENES

Un ecualizador es un dispositivo que procesa señales digitales de audio o video con el objeto de modificar su contenido. Para ello, el ecualizador puede entre otras cosas cambiar las amplitudes, la frecuencia, realzar colores, ajustar la distribución de intensidades de grises, de la señal que está siendo procesada y con eso obtener un mejor entendimiento ya sea del audio o una mejor visualización de la imagen.

Dado los 3 tipos de segmentaciones que vamos a presentar en la interfaz gráfica, consideramos 3 tipos de ecualizadores: de color, Binario y de Contraste.

Ecualizador de Color: con este ecualizador buscamos realzar los colores de la imagen RGB para que el algoritmo de segmentación logre el encendido y apagado de colores según se lo requiera.

Ecualizador Binario: Cuando el método de Canny presente gran cantidad de bordes que no permitan la correcta visualización de la imagen, usamos el ecualizador binario que no es otra cosa que la detección de bordes en una imagen binaria.

Ecualizador de contraste: nos permite variar las intensidades de los niveles de gris de la imagen para obtener una mejor segmentación de texturas. El uso de los ecualizadores se mostrará con mayor detalle en el capítulo 5.

CAPÍTULO 3:

3. IMPLEMENTACIÓN DE LA SOLUCIÓN

El segundo objetivo fue implementar una interfaz que consolide los tres métodos seleccionados para la segmentación de video digital en una sola vista. La interfaz fue planeada al estilo "Tablero de Control" (DashBoard). Ver figura 3.1.

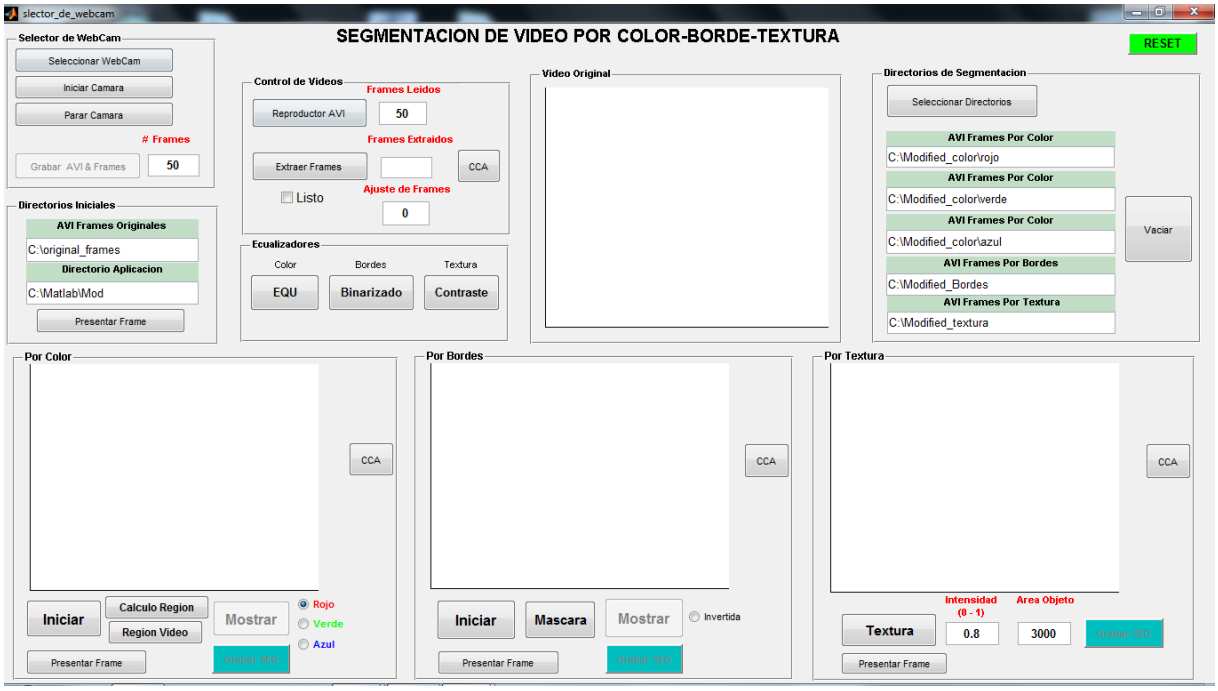


Figura 3. 1: Tablero de Control para realizar la segmentación de video

3.1 SELECCIÓN DEL AMBIENTE DE DESARROLLO

Para desarrollar la interfaz de control y los algoritmos de segmentación, escogimos el programa Matlab versión R2010a para Windows. En sistemas operativos probamos el aplicativo bajo Windows XP y 7, con plataformas de 32 y 64 Bits. En la Fig. 3.2 mostramos la selección de todo el ambiente de trabajo.

CARACTERISTICAS DEL AMBIENTE DE TRABAJO							
Versión de Matlab	√	7.10 R2010a					
Cámaras Webcam	√	HP 2MP FIXED	√	e-Messenger 112			
Requerimientos Mínimos del sistema		2 GB RAM		Core 2 Duo		20 GB Disco	
Requerimientos Recomendados	√	4 GB RAM	√	Core i3;i5	√	120 GB Disco	
Sistemas Operativos Windows	√	XP Pro 32 bits	√	XP Pro 64 bits	√	7 Pro 32 bits	√ 7 pro 64 bits

Figura 3. 2: Características de la plataforma de desarrollo

3.2 DIAGRAMA DE BLOQUES

El diagrama de Bloques Fig. 3.3 nos ayuda a visualizar y comprender mejor la interrelación entre los diferentes procesos de entrada y salida de la herramienta.

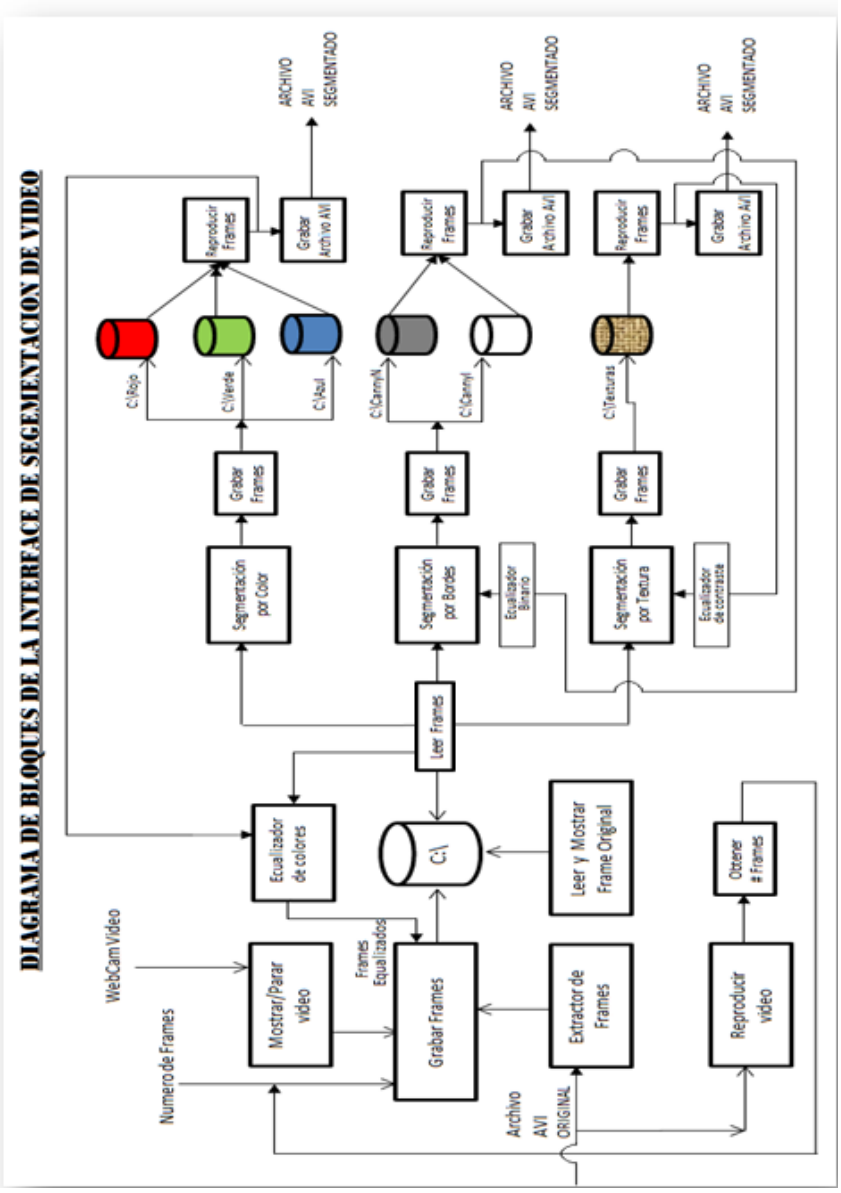


Figura 3. 3: Diagrama de Bloques del proceso de segmentación

3.3 DIAGRAMA DE FLUJO

En la Fig. 3.4 tenemos el diagrama de flujo principal de la aplicación, donde se aprecia el proceso de lectura y extracción de cuadros de un video y las diferentes segmentaciones disponibles en la aplicación.

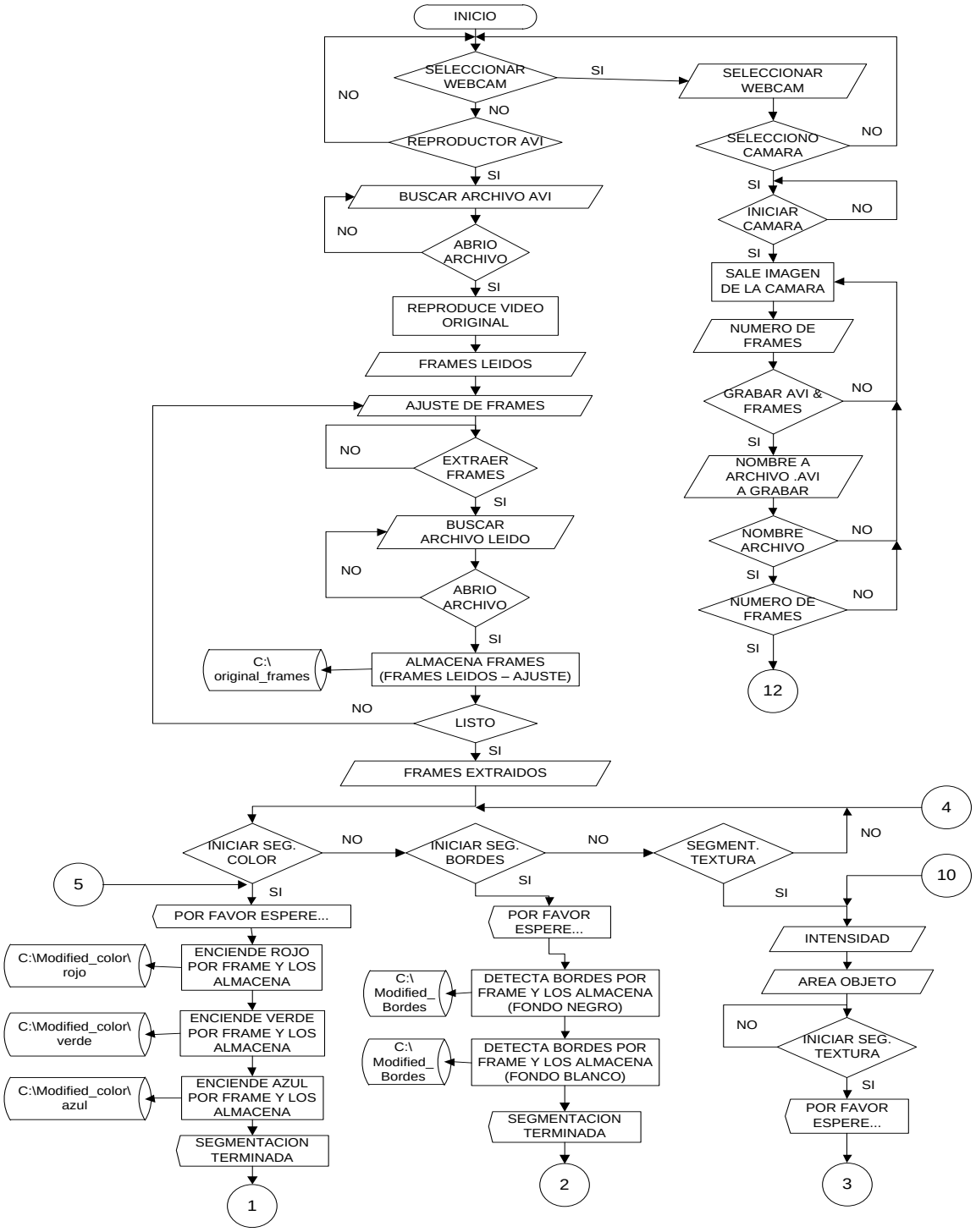


Figura 3. 4: Diagrama de Flujo - Inicio y Extracción de Cuadros

En las Fig. 3.5 y 3.6 tenemos los flujos de la segmentación por color. En la Fig. 3.5 se tiene el proceso para el encendido de los colores y en la Fig. 3.6 tenemos el proceso para la segmentación de región de colores; al final podemos a grabar 6 videos como resultado de estas segmentaciones.

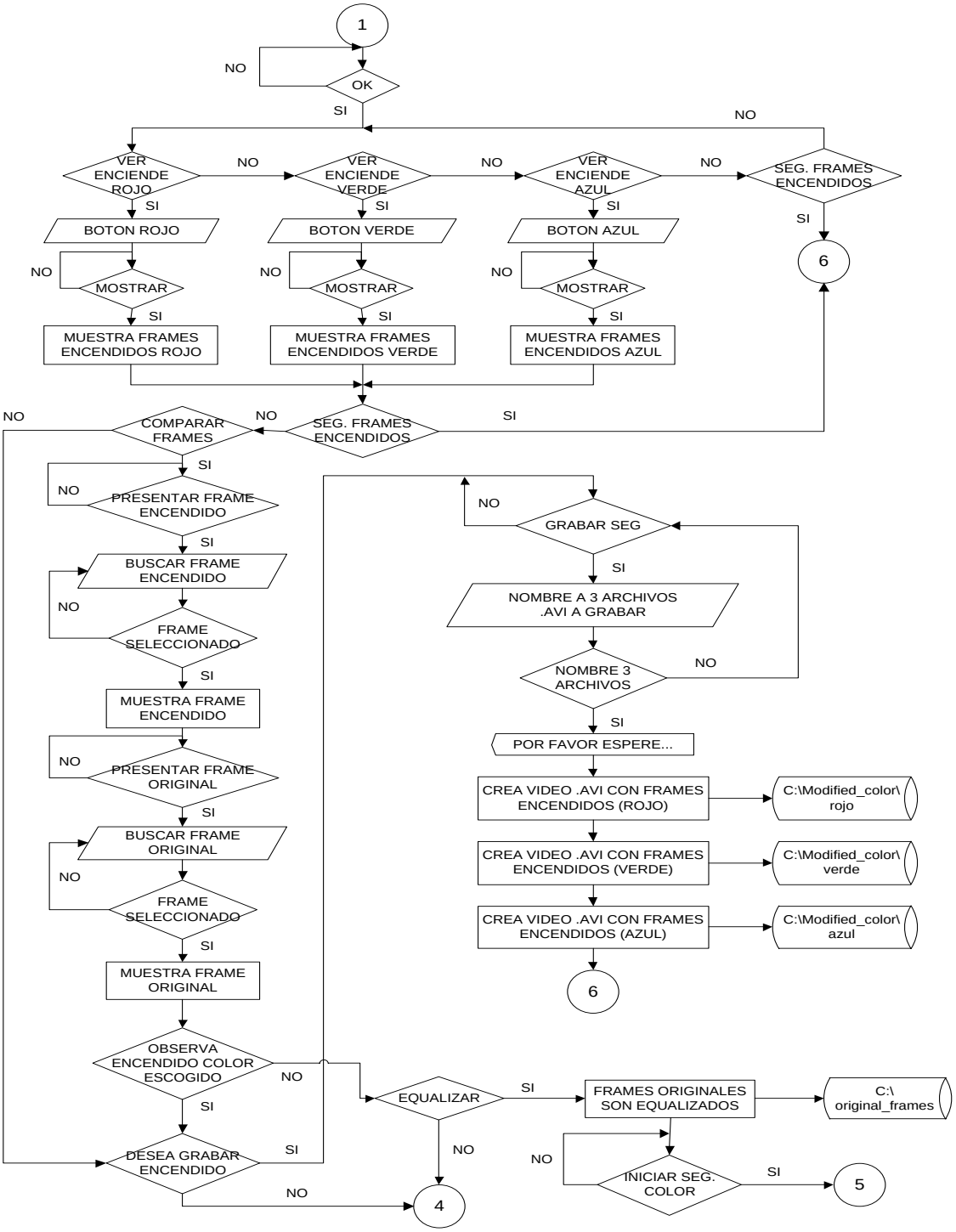


Figura 3. 5: Diagrama de Flujo - Segmentación por Color

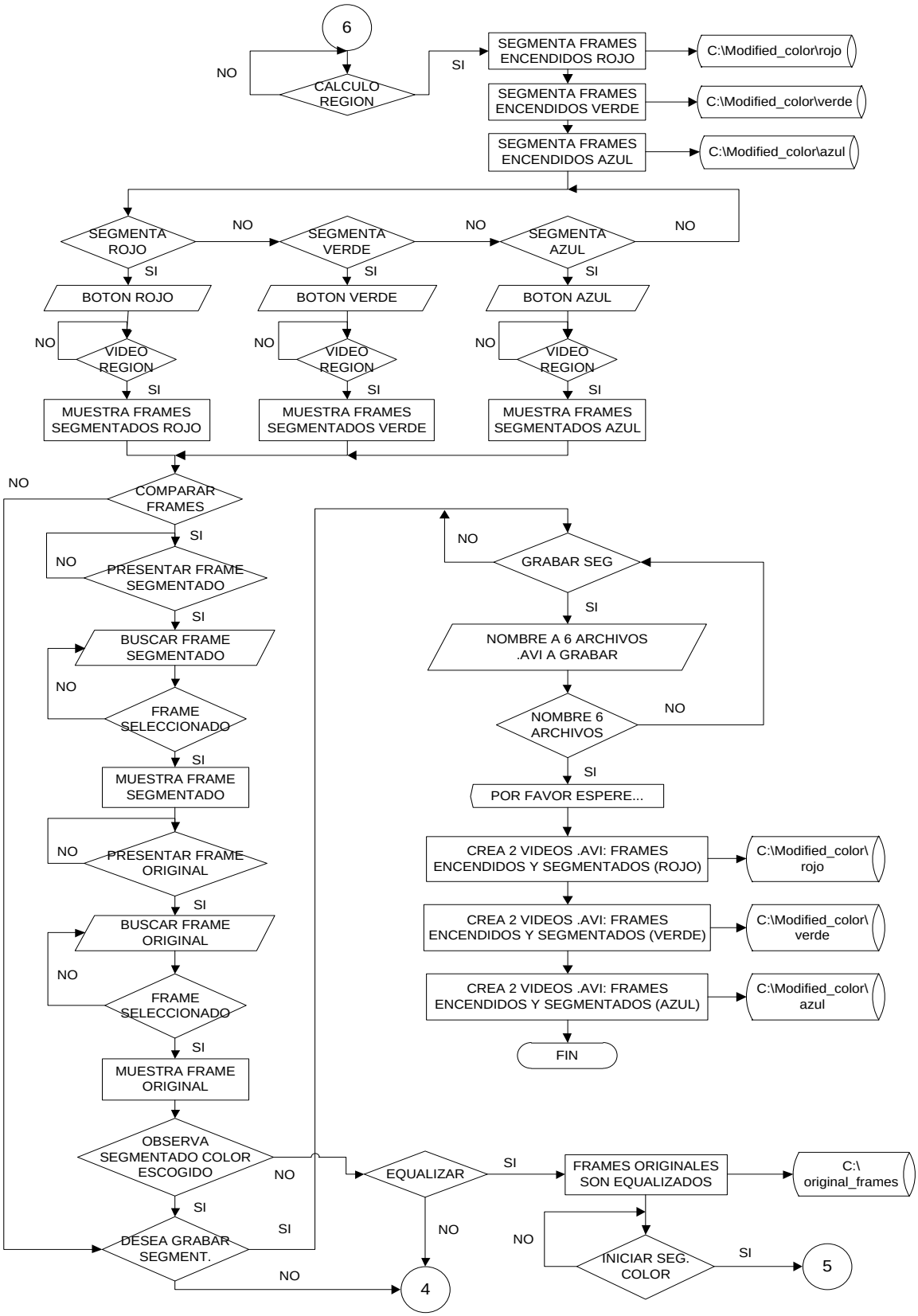


Figura 3. 6: Diagrama de Flujo - Segmentación por Región Color.

En las Fig. 3.7 y 3.8 tenemos los diagramas de la segmentación por bordes. En la primera figura se tiene el proceso de detección de bordes. En la segunda figura tenemos el proceso de segmentación de bordes en una región determinada de los cuadros que conforman el video con diferente fondo.

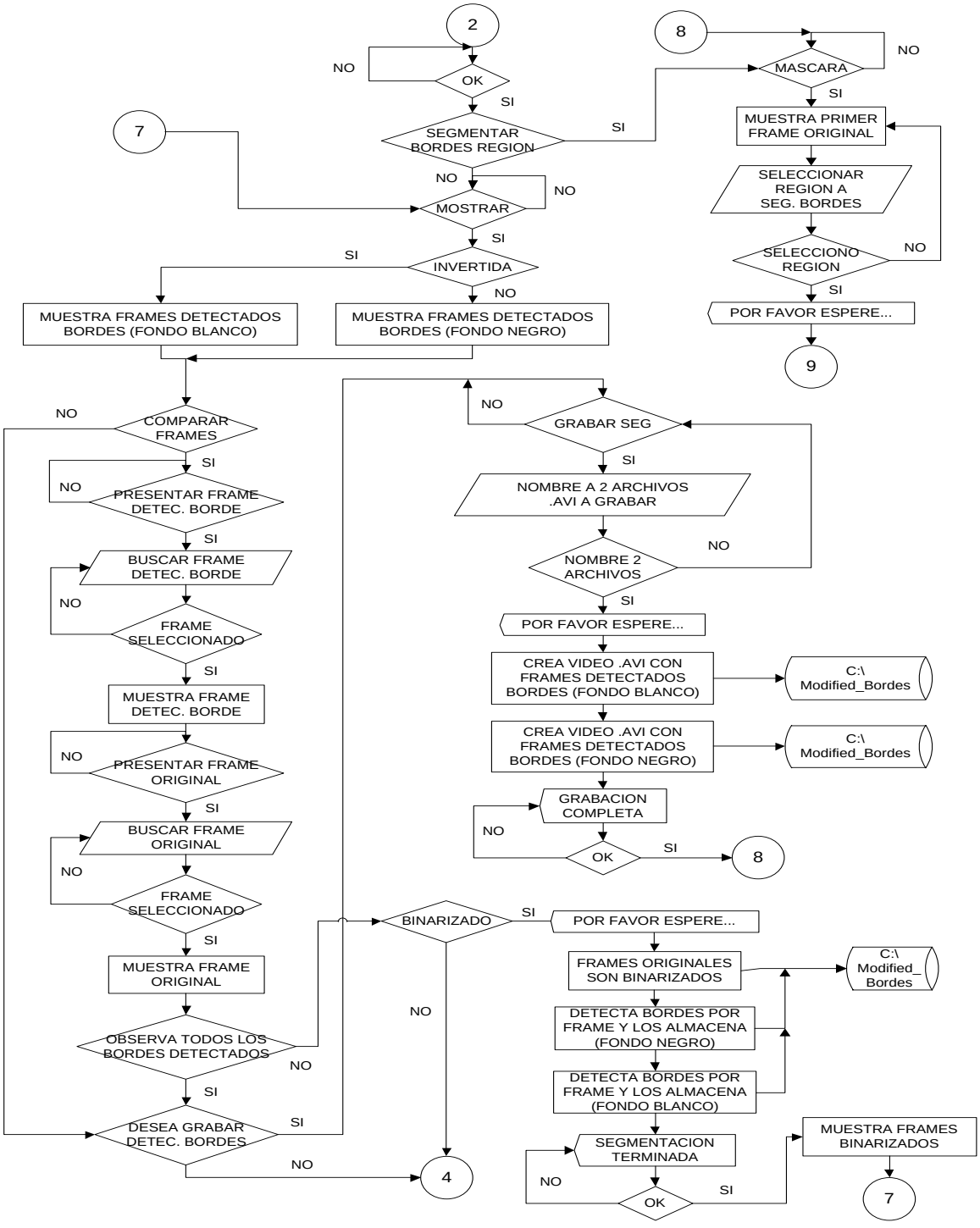


Figura 3. 7: Diagrama de Flujo - Segmentación por Bordes

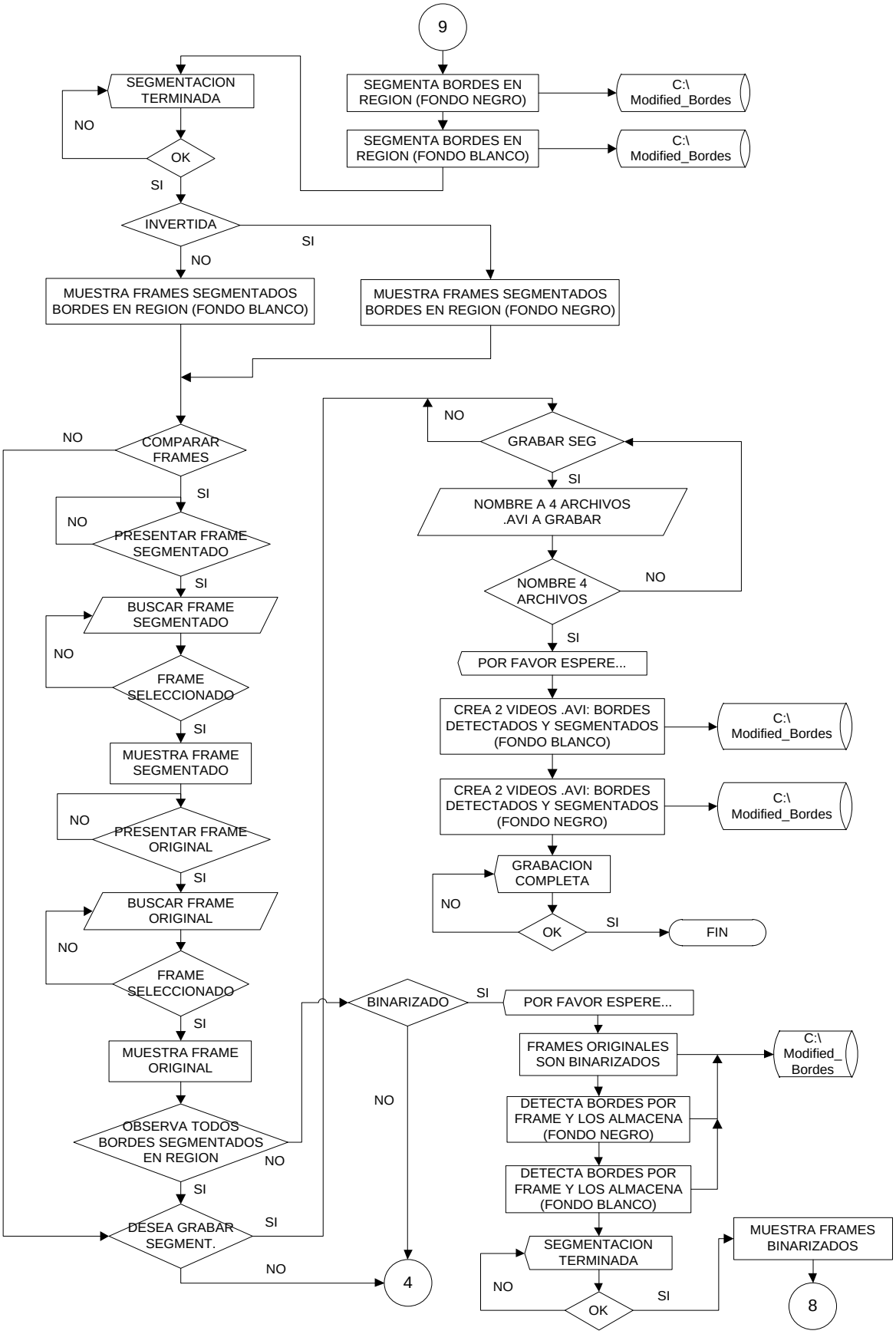


Figura 3. 8: Diagrama de Flujo - Segmentación por Bordes(Cont.)

En la Fig. 3.9 vemos el diagrama de la segmentación de textura, donde podemos modificar manualmente las áreas que queremos segmentar y la tonalidad en escala de grises de dichas áreas. Al final podemos grabar el video segmentado las texturas.

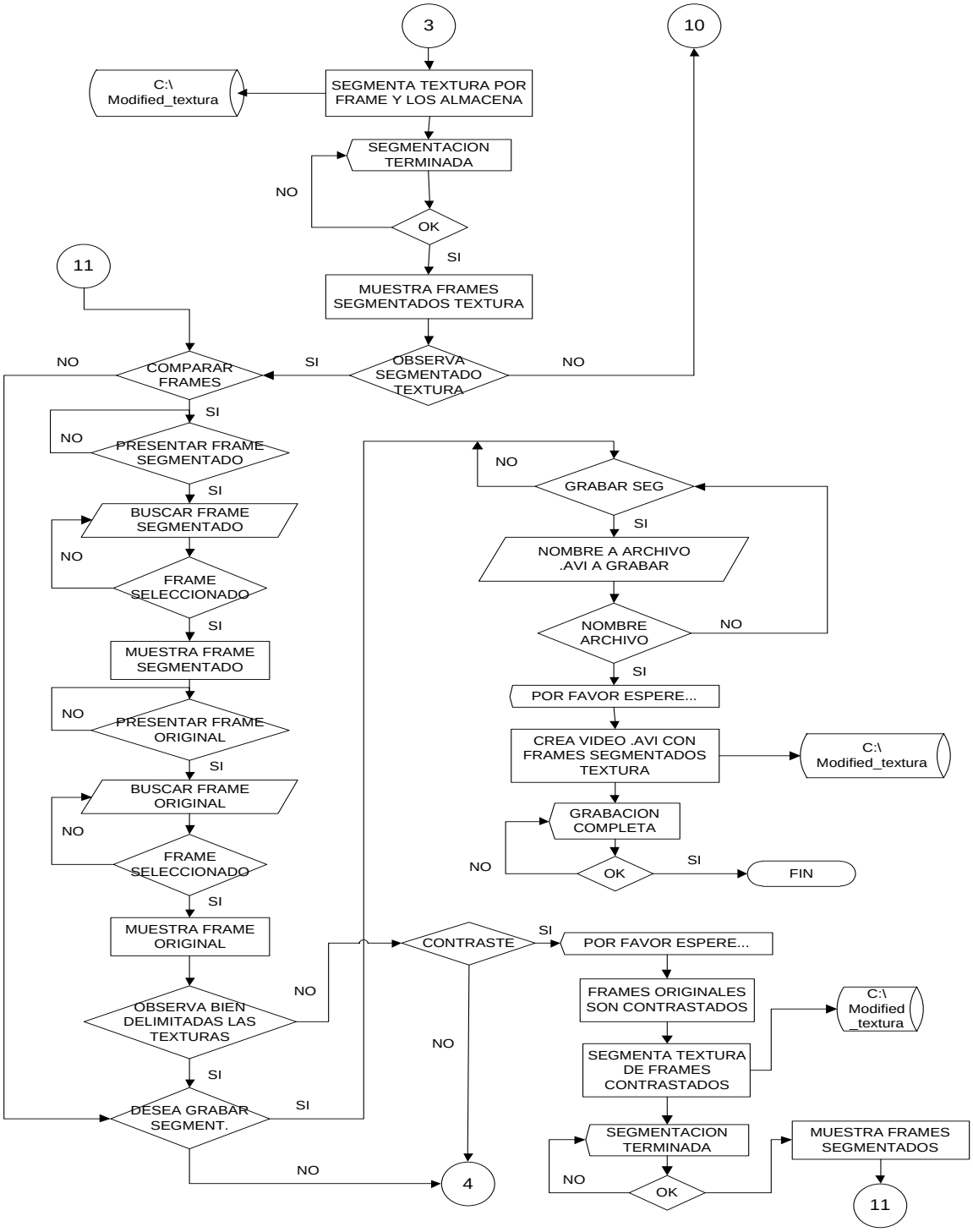


Figura 3. 9: Diagrama de Flujo - Segmentación por Texturas

En la Fig. 3.10 tenemos el diagrama del proceso de extracción de cuadros del video por medio de webcam. Este diagrama se complementa con el diagrama de flujo principal (Fig. 3.4).

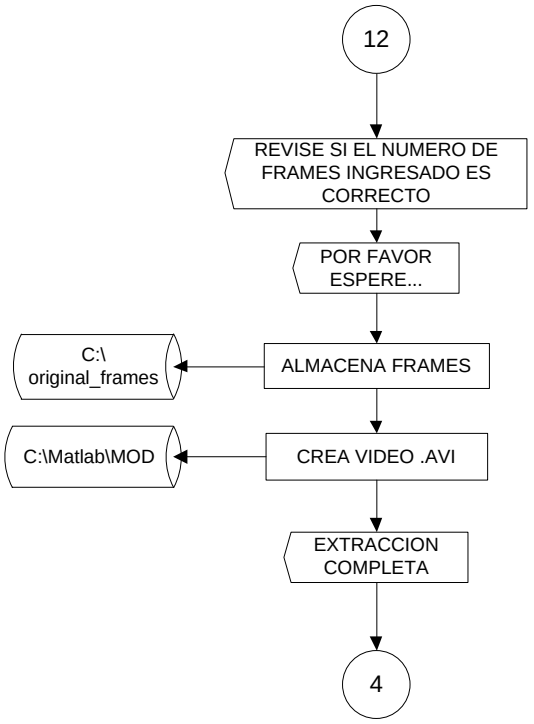


Figura 3. 10: Diagrama de Flujo - Extracción de Cuadros por Webcam

3.3 DESARROLLO E IMPLEMENTACION DE INTERFAZ GRAFICO

Uno de los retos de este proyecto, fue poder segmentar el video proveniente de archivos AVI generados en línea. Esto lo logramos haciendo entradas de video a través de cámaras WEBCAMS conectadas directamente al equipo.

3.4.1 Selector de Cámaras Webcam

Actualmente la mayoría de equipos portátiles vienen con cámara WEBCAM incluida, pero son personales y de poca movilidad. Para mejorar eso, añadimos una segunda cámara USB al equipo para poder hacer capturas de video abiertas y variadas. Luego a la interfaz se le añadió un mecanismo[2] de selección de WEBCAM, la cual

[2] El mecanismo de selección de WebCam se mostrará en detalle en el Anexo A– Manual del usuario.

detecta todas las cámaras instaladas y nos permite seleccionar una en particular como lo muestra la figura 3.11.

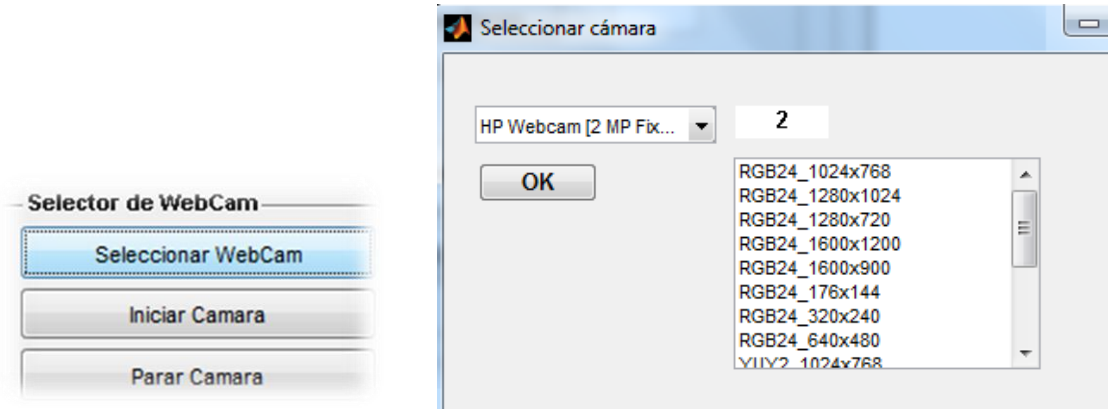


Figura 3. 11: Control de Selección de cámaras WebCam.

3.4.2 Extractor y Grabador de cuadros o tramas del video

A nuestro entender el algoritmo de extracción y grabación de las tramas de un video hacia el Disco Duro, fue el más importante de todos. Ya que una vez resuelto este problema, es decir poner el video en forma de imágenes, hizo factible que al video entero se le pueda aplicar cualquier función de MatLab para procesamiento de imágenes digitales. Fig. 3.12.

El algoritmo desarrollado durante el proyecto cuenta con la capacidad de realizar las siguientes acciones sobre un video AVI.

- Extraer uno por uno los cuadros de un video, normalmente hasta 499[3].
- Grabar los cuadros en formato JPG en una carpeta del disco duro, cada uno con su propio nombre, de forma ordenada y secuencial.
- Leer los cuadros, ponerlos dentro de un lazo y modificarlos con funciones de Matlab para procesamiento de imágenes.
- Grabar las imágenes modificadas en una carpeta del disco duro con nombre propio, de forma ordenada y secuencial.

[3] La cantidad de Cuadros que Matlab puede extraer de un video AVI depende de la cantidad de memoria RAM libre en el momento de la extracción. Se recomienda cerrar todas las aplicaciones y reiniciar la interface de Matlab antes de proceder a la extracción de los cuadros.

- Convierte los archivos JPG modificados a cuadros de video.
- Reproduce los cuadros de video modificados, modificando así el video.
- Graba el video modificado en formato AVI.

Figura 3. 12: Módulos de control de Webcam y Video.

3.4.3 Algoritmo usado para aplicar los métodos de segmentación

Para lograr la segmentación del video en formato AVI por cualquiera de los métodos seleccionados, primero reproducimos el video en la interfaz gráfica y leemos de cuantos cuadros (frames) está compuesto el video. Fig. 3.13. Nuestra interfaz es capaz de leer en promedio hasta 499 cuadros de un archivo AVI, en el caso de que un video tenga más de 500 cuadros, debemos asegurarnos que la máquina donde corre el aplicativo tenga la suficiente memoria RAM disponible para poder leerlo.

Figura 3. 13: Reproductor de video con contador de Cuadros.

Una vez leído el video, procedemos con la extracción de sus cuadros presionando el botón "Extraer Frames". Fig. 3.14. Culminada la extracción, los cuadros son almacenados en el directorio C:\Original_Frames.

Figura 3. 14: Botón de Extracción de cuadros en un video digital

Una vez que los cuadros son almacenados en formato JPG en el directorio C:\Original_Frames\, estamos listos para aplicar cualquiera de los métodos de segmentación implementados.

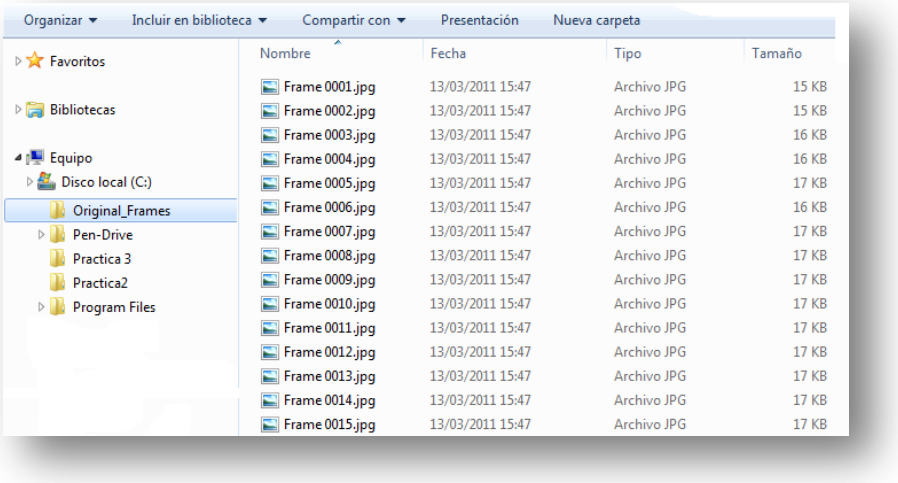


Figura 3. 15: La extracción los cuadros se almacenan en el disco duro

El proceso que realizan los métodos de segmentación para trabajar sobre las imágenes, es ir al directorio C:\Original_Frames\, leer los cuadros uno por uno e ir aplicando su algoritmo respectivo, luego los van grabando de forma ordenada en los directorios escogidos para almacenar los cuadros modificados, ver figura 3.16.

La selección y creación de estos directorios viene pre-configurada en el programa en las variables iniciales del entorno[4].

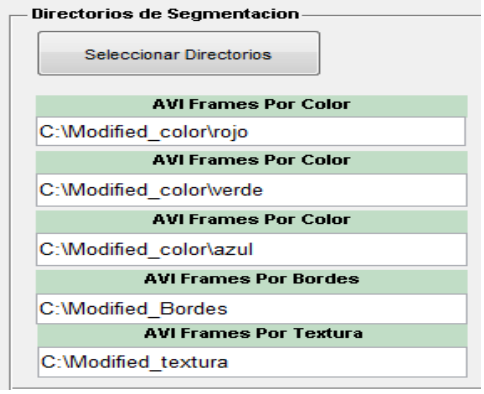


Figura 3. 16: Directorios de segmentación. Guardan los cuadros modificados

Finalmente, dentro del tablero de control de la aplicación consideramos 4 pantallas de salida donde podemos reproducir el video original y ver el resultado de cada una de las segmentaciones realizadas como se muestra en la figura 3.17.

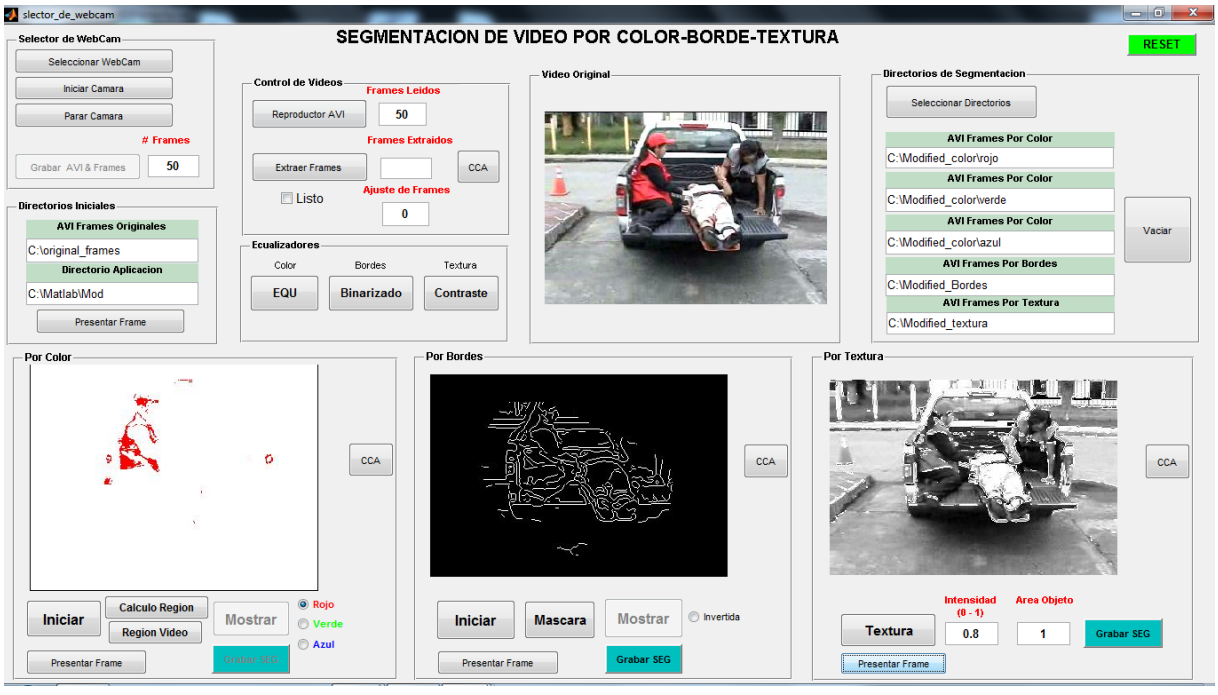


Figura 3. 17: Interfaz cargada con los 3 métodos de segmentación

[4] El código completo del programa se lo puede revisar al final del informe en el apéndice A: Manual y guía de operación del usuario.

3.4.4 Diseño de Ecualizadores

En el diseño de la interface consideramos también la implementación de 3 ecualizadores digitales para mejorar y ajustar la calidad de los videos AVI originales de tal manera que los algoritmos de segmentación sean más efectivos en sus resultados. Fig. 3.18.

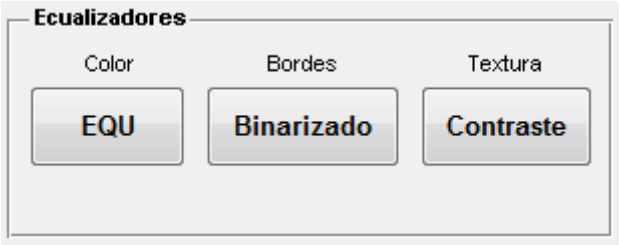


Figura 3. 18: Botonera de Ecualizadores para cada método de segmentación

El ecualizador de color tiene la función de resaltar las intensidades de los colores del video original para garantizar que el resultado de la segmentación sobre todo en áreas pequeñas, que hay múltiple diversidad de colores, sea eficaz. Fig. 3.19.

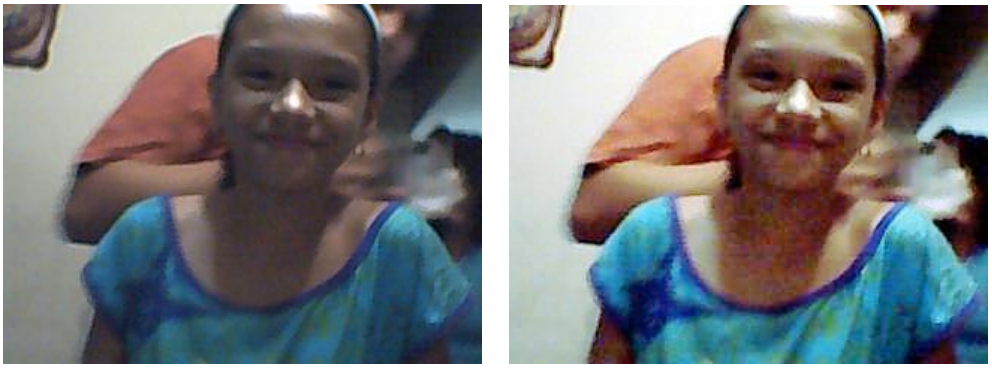


Figura 3. 19: Imagen sin ecualizar y la misma imagen ecualizada.

El ecualizador BINARIZADO en cambio tiene como función disminuir y suavizar el detalle de los bordes hechos por la segmentación de Canny. Si la segmentación por bordes genera excesos de manera que no se pueda apreciar o comprender el video original, entonces usamos el ecualizador BINARIZADO, que

genera los cuadros del video en modo Binario (blanco y negro), haciendo que el método de canny suavice y simplifique la cantidad de bordes mostrados en la imagen. Fig. 3.20.

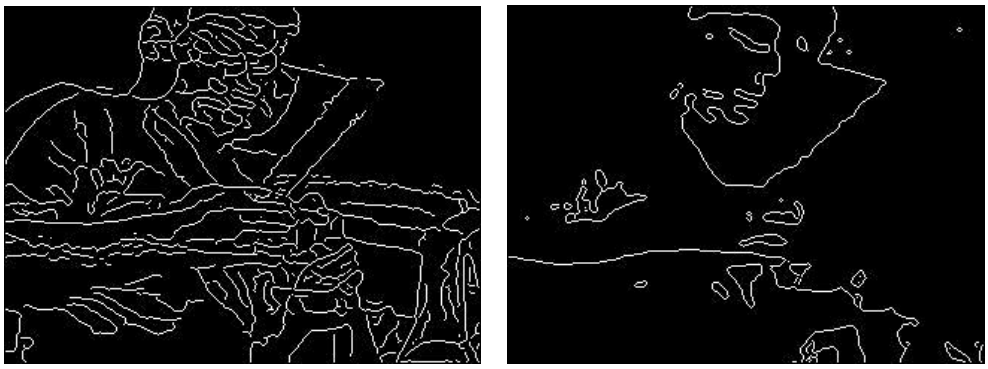


Figura 3. 20: Detección con función Canny (izq) y ecualizador Binario (der)

El ecualizador de contraste está atado en cambio a la segmentación por texturas. Este ecualizador optimiza la calidad de la imagen en escala de grises, permitiendo que el algoritmo de textura pueda identificar de mejor manera las diferentes regiones o cambios de intensidad de los cuadros del video como se muestra en la figura 3.21.

Figura 3. 21: Imagen con división de texturas (Izq) y ajuste de contraste (der)

CAPITULO 4:

4. OPERACIÓN Y PRUEBAS.

Antes de pasar a la fase de pruebas, vamos a aprender la operación de la herramienta realizando segmentaciones de video en cada uno de los métodos implementados. Lo haremos usando videos AVI existentes listos para segmentar y también generando videos nuevos desde las cámaras Webcam.

4.1 PRUEBAS Y OPERACIÓN DEL MÉTODO DE SEGMENTACIÓN POR COLOR

Para empezar las pruebas de la segmentación por colores, debemos tener previamente extraído los cuadros del video que queremos segmentar en la carpeta C:\Original_Frames; El video con el que realizaremos las pruebas de segmentación en los tres métodos se llama "Ambulancia.avi", tomado del sitio Youtube.com. En la Tabla I, listamos sus características.

CARACTERISTICAS DEL VIDEO DE PRUEBA	
Nombre del archivo	Ambulancia.avi
Numero de cuadros	121
Cuadros por segundo	30
Dimensiones	352 x 240
Compresión de video	MP42
Tipo de imagen	Color Verdadero (24 bits)

Tabla I

Para iniciar la segmentación por color debemos dar clic en el botón Iniciar y esperamos hasta que el proceso de segmentación termine. Fig. 4.1.

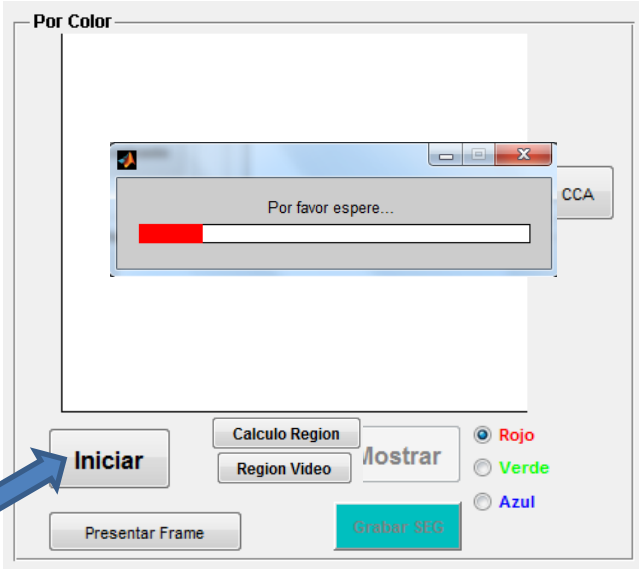


Figura 4. 1: Proceso de inicio para la segmentación por color.

Una vez que el proceso de segmentación ha terminado, nos aparece el cuadro de diálogo de confirmación que la segmentación resultó exitosa. Fig. 4.2.

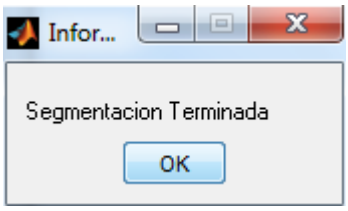


Figura 4. 2: Botón de confirmación de la segmentación terminada.

Luego damos clic en OK y notamos que el botón "Mostrar" se habilita haciendo posible visualizar el video segmentado según el color escogido en las opciones Rojo Fig. 4.3, Verde Fig. 4.4 o Azul Fig. 4.5.

Figura 4. 3: Resultado del encendido de color Rojo.

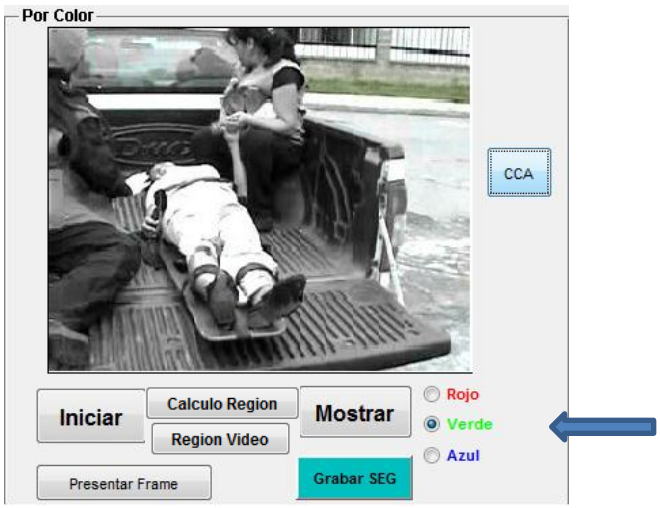


Figura 4. 4: Resultado del encendido de color Verde.

Figura 4. 5: Resultado del encendido de color Azul.

4.1.1 CALCULO DE REGION Y VIDEO DE REGION

Dentro de la sección Por Color, hay dos botones que son el cálculo de región y video de región. El primero sirve para obtener regiones de interés (ROI) del video procesado. En la segmentación por colores, las regiones de Interés son los colores. El segundo botón Video Región, es el que nos permite observar la secuencia de cuadros en video con la región de interés marcada dentro de cada uno de ellos. Fig. 4.6.

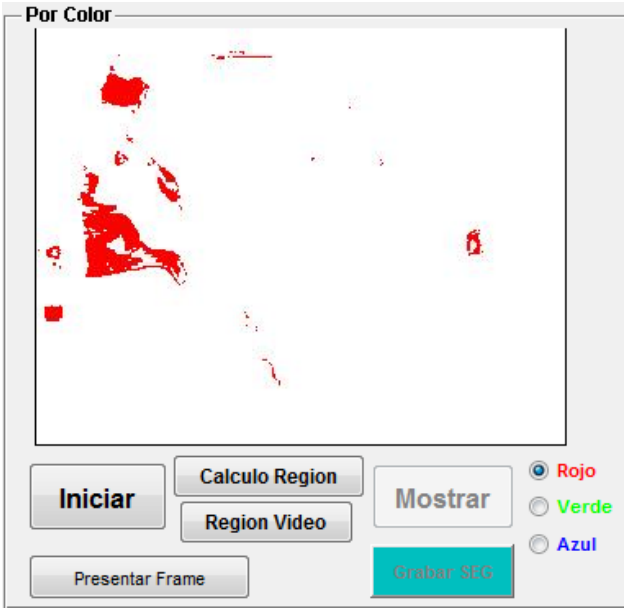


Figura 4. 6: Segmentación completa de la Región de Color Rojo.

4.2 PRUEBAS Y OPERACIÓN DEL MÉTODO DE SEGMENTACIÓN POR BORDE

Al igual que el método anterior, la segmentación por bordes necesita que los cuadros del video original estén ubicados en la carpeta C:\Original_Frames. Para iniciar la segmentación por bordes el proceso es muy similar al de segmentación por color y empieza dando clic en el botón iniciar. Fig. 4.7.

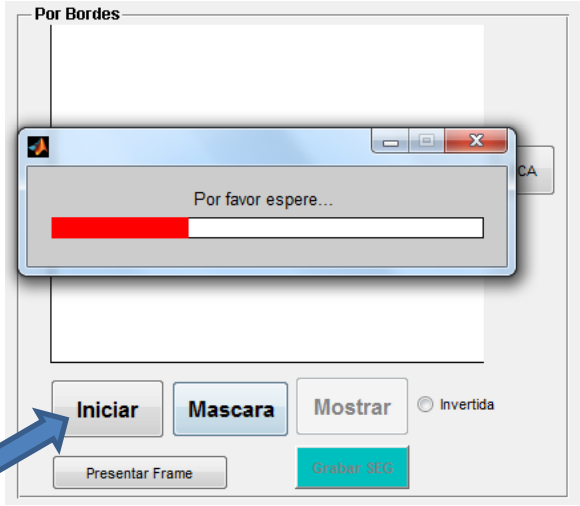


Figura 4. 7: Inicio del proceso de segmentación por Bordes.

Una vez realizada la segmentación por bordes, el botón "Mostrar" se habilita y con eso podemos ver el video segmentado mediante el algoritmo de Canny de forma normal e invertida como se muestra en las figura 4.8.

Figura 4. 8: Detección por bordes Canny Invertida (izq.) y Normal (der).

4.2.1 Máscara de región en la detección de bordes

Para obtener una determinada región en el resultado de las imágenes con detección de borde, añadimos a la interfaz un botón denominado MASCARA el cual nos permite ingresar de forma manual una máscara de región en la primera imagen

del video, la cual servirá como patrón de segmentación para todos los otros cuadros del video tal como se muestra en la figura 4.9.

Figura 4. 9: Detección por bordes Canny con máscara de región manual.

4.3 PRUEBAS Y OPERACIÓN DE LA SEGMENTACIÓN POR TEXTURAS

El último de los tres métodos implementados para las pruebas de segmentación sobre un video digital fue el de las texturas. Al igual que los métodos anteriores, antes de iniciar el proceso de texturas, debemos verificar que los cuadros extraídos del video original estén en la carpeta C:\Original_Frames.

La segmentación por texturas cuenta con dos campos de entrada para realizar ajustes sobre el resultado obtenido, Fig. 4.10. Los valores por defecto de los campos son de 0.8 para la intensidad de la Luminancia, es decir que todo lo que esté sobre 0.8 se considerará blanco y lo que esté por debajo se considerará negro. El otro valor por defecto es 3000 que sirve para remover las áreas de los objetos menores al valor ingresado en pixeles, es decir que el programa solo considerará áreas con conectividad de vecindad mayores a 3000 pixeles, las de menor cantidad no se tomarán en cuenta.

Figura 4. 10: Segmentación por Texturas. Valores por defecto.

Con estos dos valores podemos ir ajustando el resultado de la segmentación hasta llegar al resultado que queremos. Por ejemplo, Disminuyendo el valor del *Área Objeto*, podemos ir ingresando a la segmentación áreas más pequeñas dentro de la imagen, incluso de menos de 10 pixeles. Una vez que el tamaño del área ha sido ajustado, empezamos con el ajuste del valor de Intensidad que al disminuirlo va a crear una máscara de región de color negro sobre las áreas segmentadas cumpliendo de esta manera con la segmentación de la imagen.

Para iniciar el proceso de segmentación por texturas damos clic en el botón *Textura* y esperamos la confirmación de que el proceso terminó exitosamente. Fig. 4.11

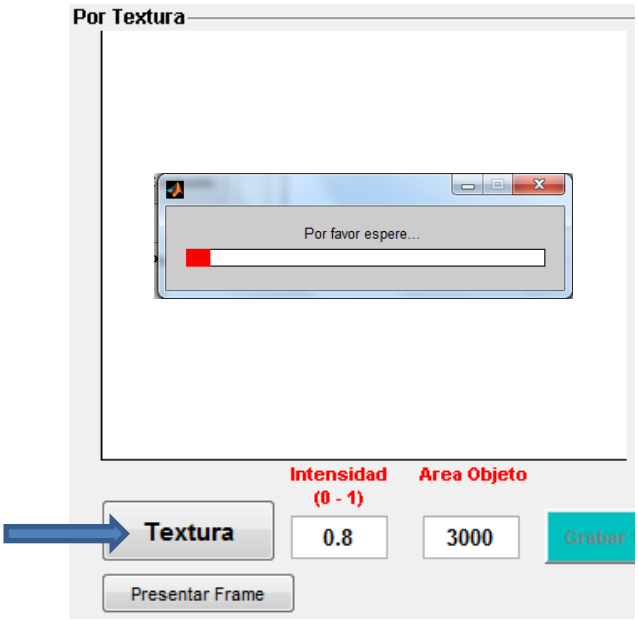


Figura 4. 11: Inicio del proceso de segmentación por Texturas.

Concluido el proceso, el video con el resultado de la segmentación se muestran en la pantalla visual, y queda listo para ser ajustado con los parámetros de Textura. Para entender los efectos de esta segmentación, vamos a trabajar con el cuadro n°27 del video "Ambulancia.avi". Fig. 4.12.

Figura 4. 12: Cuadro del video "Ambulancia.avi" antes de la segmentación.

Lo primero que debemos recordar es que el video es a color y la segmentación por texturas transforma el video a escala de grises. Esto es lo que sucede con la figura 4.13 y los valores por defecto de los campos de textura.

Figura 4. 13: Cuadro 27 segmentado por texturas con valores 0.8 y 3000.

Luego hacemos la división por texturas de la imagen, esto lo logramos disminuyendo el valor del campo "*Área Objeto*" a 10. Ver Fig. 4.14.

Figura 4. 14: Cuadro 27 segmentado por texturas con valores 0.8 y 10.

El siguiente paso luego de lograr la división de las texturas, es establecer la máscara de región de las partes que deseamos segmentar. Para ver la mascará de región, disminuimos ahora el campo de Intensidad a 0.25; Esto hará que ciertas vecindades de pixeles se oscurezcan dentro de los límites previamente divididos, ver Fig. 4.15.

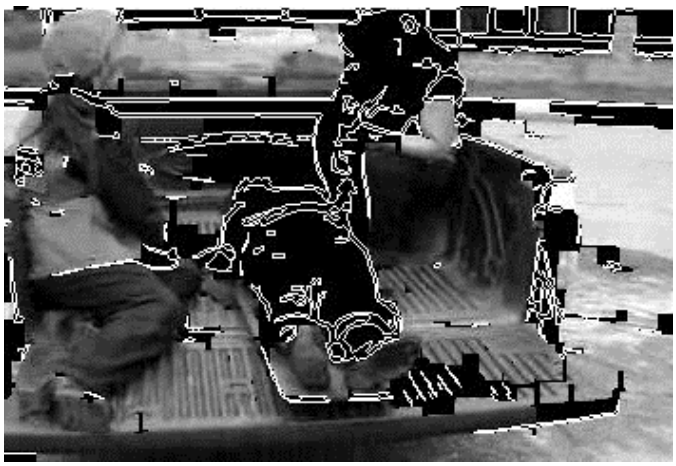


Figura 4. 15: Cuadro 27 segmentado por texturas con valores 0.25 y 10.

Finalmente quitamos las líneas divisorias de la segmentación por texturas, aumentando considerablemente el valor del Área Objeto en este caso 5000, haciendo el efecto de que estas áreas son removidas de la imagen. Fig. 4.16.

Figura 4. 16: Cuadro 27 segmentado por texturas con valores 0.28 y 5000.

En esta figura se puede observar la segmentación de áreas de igual textura mayores a 5000 pixeles; dichas áreas tienen una tonalidad obscura en escala de grises.

CAPITULO 5:

5. PRESENTACIÓN Y ANÁLISIS DE RESULTADOS

Luego de las pruebas de operación de cada uno de los métodos de segmentación, la mayoría de los resultados estuvieron dentro de lo esperado. Sin embargo, hubo algunos resultados no tan exactos que merecieron un análisis más profundo para entenderlos y esos son los que presentamos aquí. El análisis lo realizamos con los videos "Fiesta.avi" y "Vestido_colores.avi"; veamos sus características en la Tabla II.

CARACTERISTICAS DE LOS VIDEOS DE PRUEBA		
Nombre del archivo	Fiesta.avi	Vestido_colores.avi
Numero de cuadros	200	46
Cuadros por segundo	15	15
Dimensiones	320 x 240	320 x 240
Compresión de video	Ninguna	Ninguna
Tipo de imagen	Color Verdadero (24 bits)	Color Verdadero (24 bits)

Tabla II

5.1 ANALISIS VISUAL DE LA SEGMENTACIÓN POR COLOR

Uno de los problemas en la segmentación de color fue el de las tonalidades parecidas al encender ciertos colores; por ejemplo con el color turquesa. Al realizar el encendido del color verde, no sabíamos hasta que punto el algoritmo podía segmentar este color. En la figura 5.1 se muestra la imagen encendido el verde que no deja pasar el azul, pero el encendido por azul si permite el paso de ciertos tonos turquesa.

Figura 5. 1: Cuadro n°4 de "Fiesta.avi" encendido en verde y azul

Otro problema similar se dio con el encendido por color Rojo, ya que el algoritmo dejó pasar casi todas las tonalidades rojas y derivadas como por ejemplo el color de la piel, tonos cafés rojizos, colores rosados, anaranjados, e incluso hasta tonos morado. En la figura 5.2 podemos apreciar estos efectos.

Figura 5. 2: Encendido por color rojo, dejando pasar tonalidades rojas.

Otro problema más que detectamos, fue la dificultad del algoritmo en encender apropiadamente los colores cuando están concentrados en áreas muy pequeñas como en el video "Vestido_colores.avi". Ver Fig. 5.3.

Figura 5. 3: Cuadro original y las encendidas por rojo, verde y azul.

Viendo las imágenes podemos notar que entre el cuadro original (IZQ) y la encendida por color rojo casi no hay diferencias, lo único que se ha logrado filtrar es el fondo de la imagen. El cuadro que corresponde a la encendida por verde es el que mejor resultados dio en este video y la encendida por azul también presentó problemas con los colores excepto con el fondo azul.

5.1.1 Uso del Ecualizador EQU para mejorar el encendido de Color

Para ayudarnos a corregir los problemas presentados por el encendido del color, implementamos un Ecualizador de color cuya función es separar la matriz RGB en tres matrices independientes de dos dimensiones, de tal manera que a cada matriz le podamos aplicar la función *Histeq* para ecualizar la imagen de cada plano, realzando el contraste de la imagen a través de un histograma. Finalmente las matrices son concatenadas en un arreglo multidimensional otra vez usando la función *CAT.* En la figura 5.4 mostramos el cuadro n°4 del video "Fiesta.avi" que contiene la imagen original, la imagen encendida por azul y por verde, y abajo en la Fig. 5.5 están las mismas imágenes luego de aplicado el Ecualizador de color.

Figura 5. 4: Cuadro n°4 de "Fiesta.avi"; original y encendido en verde y azul

Figura 5. 5: Cuadro n°4 de "Fiesta.avi"; ecualizados.

El análisis visual luego de aplicado el ecualizador, es que el resultado del encendido es más eficiente en las imágenes ecualizadas que las originales y esto se debe a que el ecualizador hace una mejor redistribución de los 256 niveles de grises por toda la escala y evita que estos se concentren en un solo rango que normalmente ocasiona que la imagen aparezca muy oscura o muy brillante.

Veamos ahora los resultados del ecualizador sobre las tonalidades rojas. A simple vista notamos que no han ocurrido cambios significativos y que los problemas de las tonalidades rojas se mantienen como se aprecia en la figura 5.6. A pesar del uso del ecualizador, se mantiene el paso de tonalidades cafés, naranja y morado.

Figura 5. 6: Cuadros encendidos por color rojo.

Finalmente hicimos la prueba de realizar el encendido de color sobre imágenes con zonas multicolores de áreas muy pequeñas y con gran variedad de tonos en el video "Vestido_colores.avi". En la prueba anterior no habíamos tenido buenos resultados y quisimos mejorarlo usando el ecualizador. Ver Fig. 5.7.

Figura 5. 7: Cuadro original y encendidos Rojo, Verde y Azul ecualizadas

Ahora podemos apreciar que el encendido por color funciona mucho mejor para áreas con mucho colorido usando el ecualizador que sin él. Con la imagen ecualizada la encendida por colores rojo, verde y azul funcionó sin problemas. Fig. 5.7.

5.2 ANALISIS VISUAL DE LA SEGMENTACIÓN POR BORDES

La función EDGE (Bordes) es la que realiza la detección de bordes en una imagen a través de varios métodos existentes en Matlab (capítulo 2 sección 3). El método Canny funciona muy bien para la detección de contornos, siendo sus únicas

variables los umbrales (THRESH) y la desviación del filtro Gaussiano (SIGMA). En nuestro caso tuvimos que ir ajustando estos valores de tal manera que se disminuya el ruido en la imagen pero sin difuminar los contornos. Ver Figuras. 5.8 y 5.9.

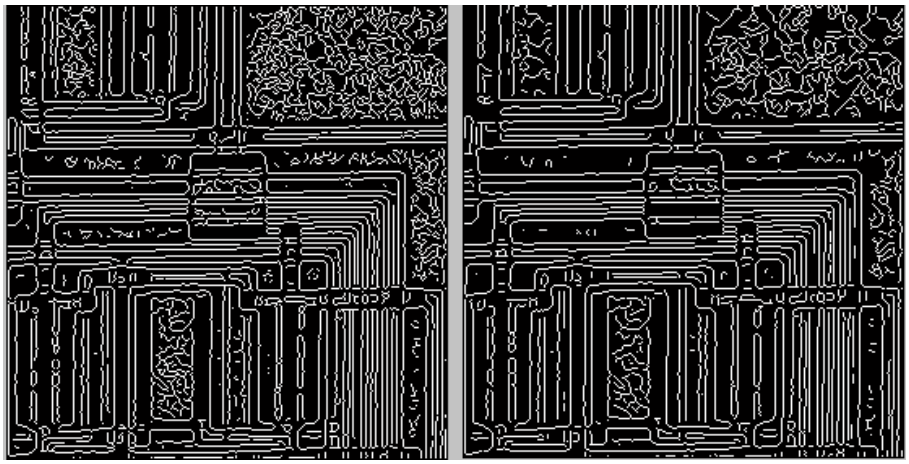


Figura 5. 8: Detección de bordes con Canny; Umbral=0.1 Desviación=0.8

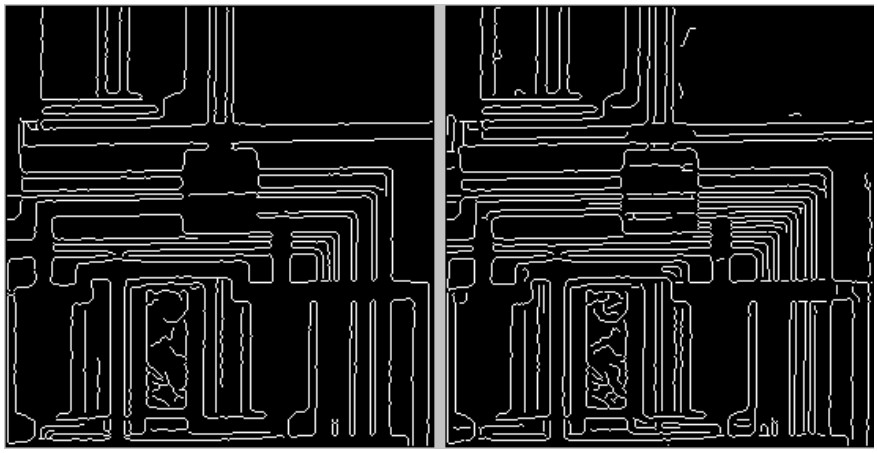


Figura 5. 9: Detección de bordes con Canny; Umbral=0.4 Desviación=0.8

Para tener otra perspectiva de los resultados que se obtienen por este método de segmentación, hemos añadido a la interfaz gráfica, la capacidad de mostrar las imágenes segmentadas por Canny pero invertidas, algo así como el

negativo de una foto. La inversión se logra con el operador lógico negación (~). Ver Fig. 5.10.

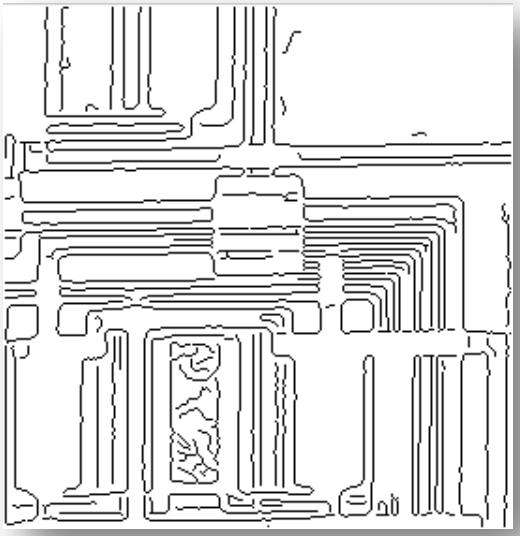


Figura 5. 10: Imagen con detección de bordes por Canny invertida.

5.2.1 Uso del Ecualizador BINARIO en la Detección de Bordes

En algunos casos la detección de bordes, dependiendo de la calidad del video, presenta mucho ruido y un exceso de bordes en la imagen resultante, lo que dificulta mucho la identificación o apreciación de la imagen central. Para atenuar esto, implementamos un ecualizador binario, que no hace otra cosa que binarizar las imágenes (transformarlas a blanco y negro) para luego de eso proceder con la detección de bordes usando el algoritmo de Canny tanto en la forma normal como la invertida.

El resultado es una figura con menos detalle que las que se tienen con la imagen en escala de grises, pero con lo suficiente como para identificar los bordes de la figura original. Ver Fig. 5.11. Esto ayuda mucho en la percepción visual de la persona que ve el video segmentado.

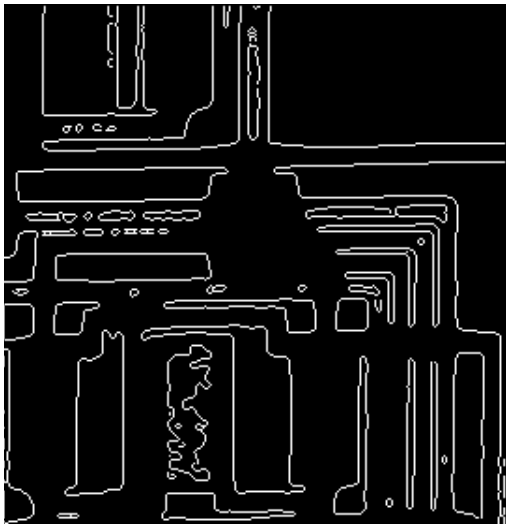


Figura 5. 11: Imagen Binarizada con detección de bordes por Canny.

5.3 ANALISIS VISUAL DE LA SEGMENTACIÓN POR TEXTURAS

El análisis visual de la segmentación por texturas fue el más complejo de todos al tener que hacerlo sobre un video completo y no sobre una imagen determinada. Cabe destacar que el algoritmo usado para la segmentación de texturas, opera sobre imágenes en escala de grises y binarias; y las imágenes en escala de grises presentan muchas variantes debido al cambio de posición, forma o intensidad de color de los pixeles en una vecindad.

Para este análisis vamos a cargar el video "Banner.avi". Las características del video las vemos en la Tabla III.

CARACTERISTICAS DEL VIDEO DE PRUEBA	
Nombre del archivo	Banner.avi
Numero de cuadros	50
Cuadros por segundo	15
Dimensiones	352 x 240
Compresión de video	Ninguna
Tipo de imagen	Color Verdadero (24 bits)

Tabla III

Cargamos el video y le aplicamos la segmentación por texturas. Luego tomamos un cuadro del video para ver el efecto de la segmentación. Fig. 5.12.

Figura 5. 12: Video Banner.avi segmentado por texturas; Int=0.8; Area=3000.

En la figura 5.12 apreciamos que en líneas rectas y bien definidas, las divisiones hechas por la segmentación de textura se marcan bien, pero en cambio en la parte del escudo que está sobre el banner de tela que tiene movimientos ligeros, la demarcación de las texturas no es totalmente completa. Haciendo ajustes a los parámetros de entrada de la segmentación por texturas, 0.1 para la intensidad y 10 para el área, vemos el resultado en la figura 5.13. Los valores en los campos "Area de Objeto" e "Intensidad" pueden ingresarse manualmente en la interfaz, y fueron creados con la finalidad que el usuario pueda segmentar regiones de diferente tamaño (que compartan una misma textura), y poner diferente tonalidad en escala de grises a dichas regiones.

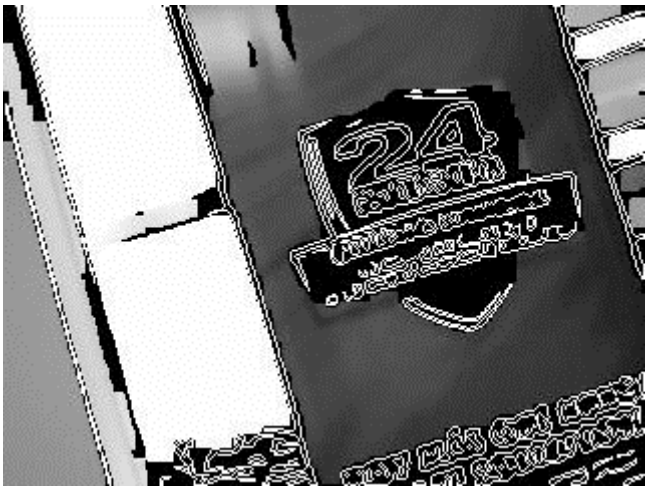


Figura 5. 13: Video Banner.avi segmentado por texturas; Int=0.1; Area=10.

Para lograr la segmentación total del escudo, aumentamos el valor del área de la máscara de región al valor original 3000 y vemos el resultado en la figura 5.14.

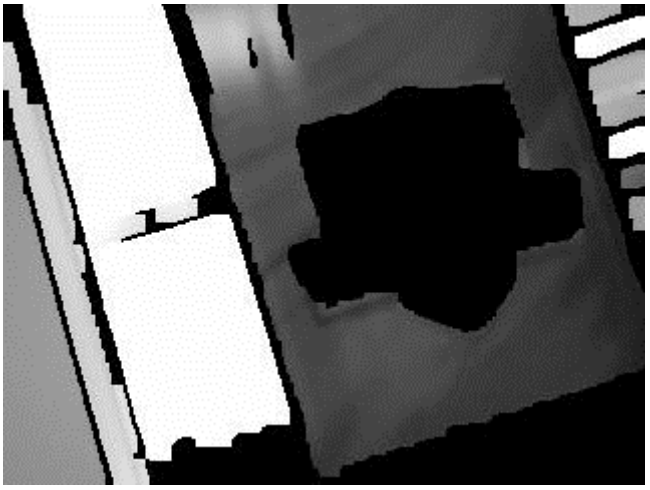


Figura 5. 14: Video Banner.avi segmentado por texturas; Int=0.1; Area=3000.

Sin embargo como hemos dicho al inicio de este análisis, el valor del área de la máscara de región debe calcularse adecuadamente ya que un valor de 2000 por ejemplo, provoca un efecto de intermitencia debido a que a pesar de ser un video estático, se suceden cambios en las intensidades de grises en los cuadros

consecutivos del video, haciendo que la línea de división entre texturas aparezca y desaparezca entre cuadro y cuadro causando el efecto mencionado y que podemos apreciar en la siguiente figura 5.15.

Figura 5. 15: Intermitencias en Banner.avi con valores de Int=0.1; Area=2000.

5.3.1 Uso del Ecualizador de contrastes

El ecualizador de contrastes está basado en la función *Imadjust* (E4), la cual sirve para realizar transformaciones en las intensidades de los niveles de grises de una imagen; su sintaxis es la siguiente:

J = IMADJUST(I,[ENT_INF; ENT_SUP],[SAL_INF; SAL_SUP],GAMMA) (E4)

La relación entre las variables y el coeficiente gamma lo vemos en la Fig. 5.16.

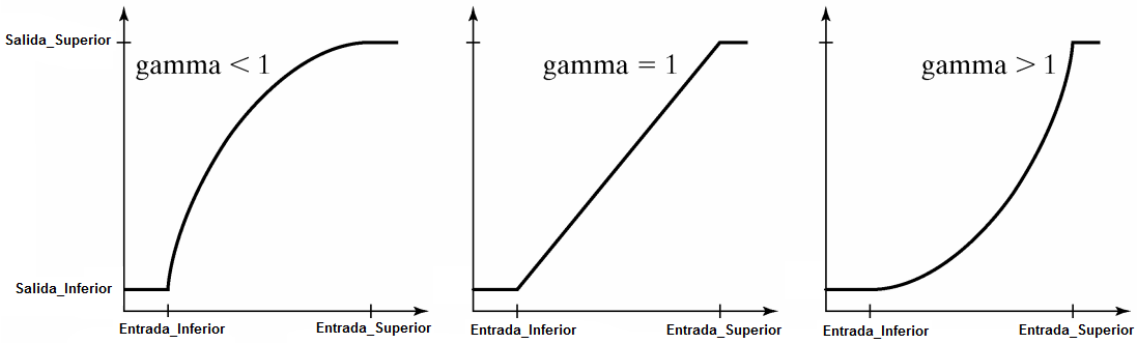


Figura 5. 16: Representación gráfica del coeficiente gamma.

Analizando las gráficas, el ecualizador fue ajustado con los valores de I=imadjust(I, [0.2 0.8], [0 1] ,1); gamma igual a 1, porque buscamos una variación lineal en los cambios de intensidad y aprovechar toda la escala de 256 grises. Si gamma es mayor o menor que 1, el exponente de la función imadjust es

exponencial y por lo tanto satura más rápido los niveles de gris, sin aprovechar todo el rango de 256 grises. Este ajuste nos ayudó a disminuir los efectos de las intermitencias ya que la imagen tiene una menor variación en la escala de grises. Ver Fig. 5.17.

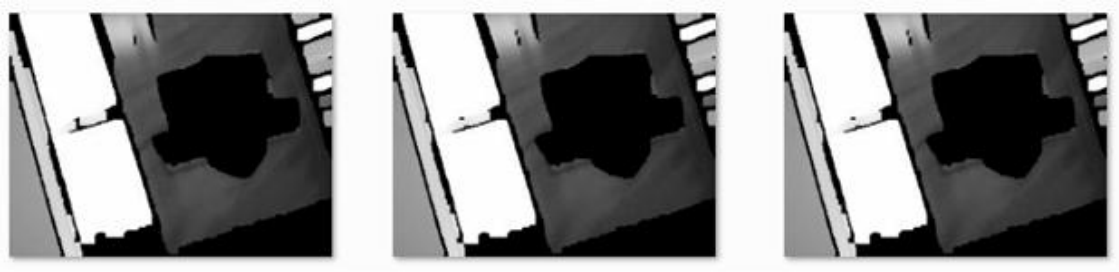


Figura 5. 17: No hay intermitencia luego del ajuste de contraste

5.4 ANALISIS DE HISTOGRAMAS DE LA SEGMENTACIÓN POR COLORES.

Realizamos un análisis de histogramas al encendido de colores, ya que este proceso es primordial para obtener una mejor segmentación de color. Un histograma es la representación gráfica de los pixeles de la Imagen. El histograma (1) corresponde al video "Vestido_colores.avi", el cual se grabó con una webcam. Ver Figura 5.18.

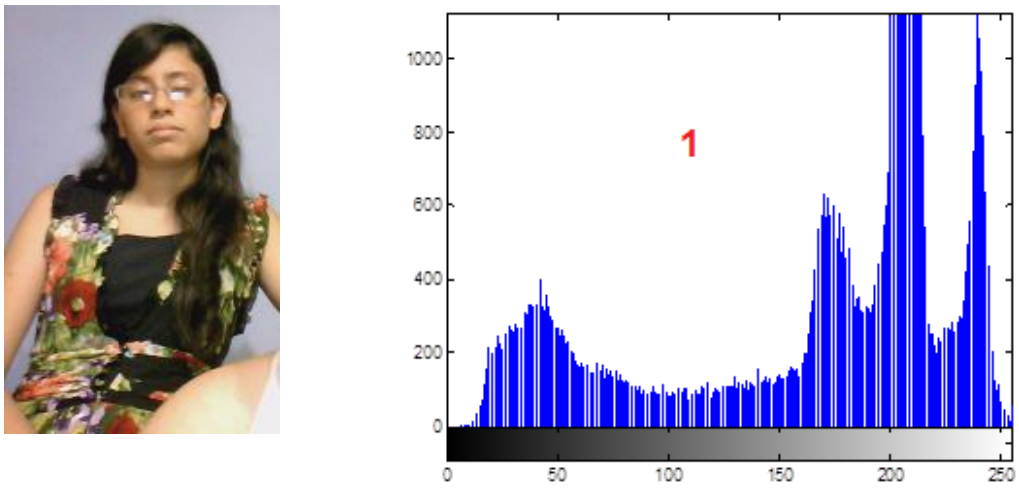


Figura 5. 18: Histograma de imagen original.

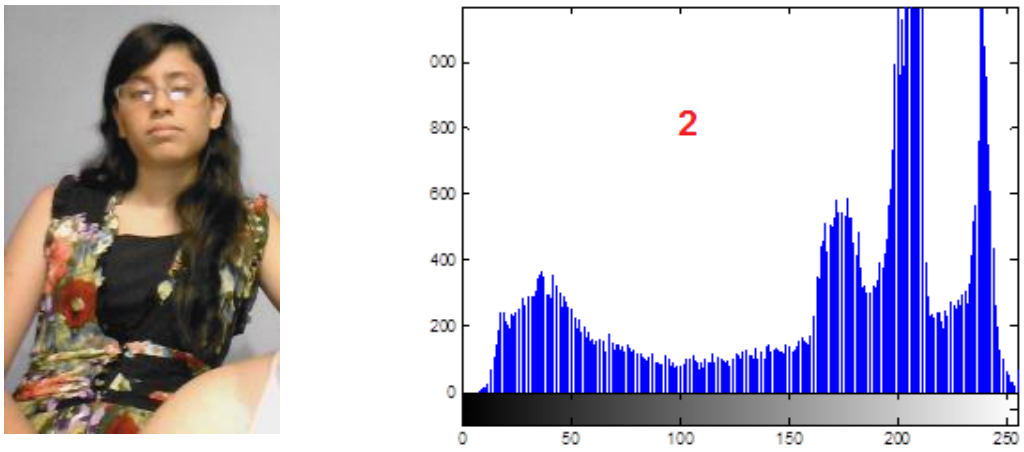


Figura 5. 19: Histograma de imagen encendido color rojo.

El histograma (2) corresponde a la imagen encendida por el color rojo (Fig. 5.19). El resultado fue el esperado, ya que los 2 histogramas son iguales, dando a conocer que el encendido del color rojo fue exitoso. Lamentablemente podemos darnos cuenta que la imagen encendida también dejó pasar otros colores, por lo que es necesario ecualizar la imagen. El histograma (3) corresponde a la misma imagen del video "Vestido_colores.avi" pero ecualizada (Fig. 5.20), luego ecualizada y encendida por el color rojo (histograma (4), Fig. 5.21).

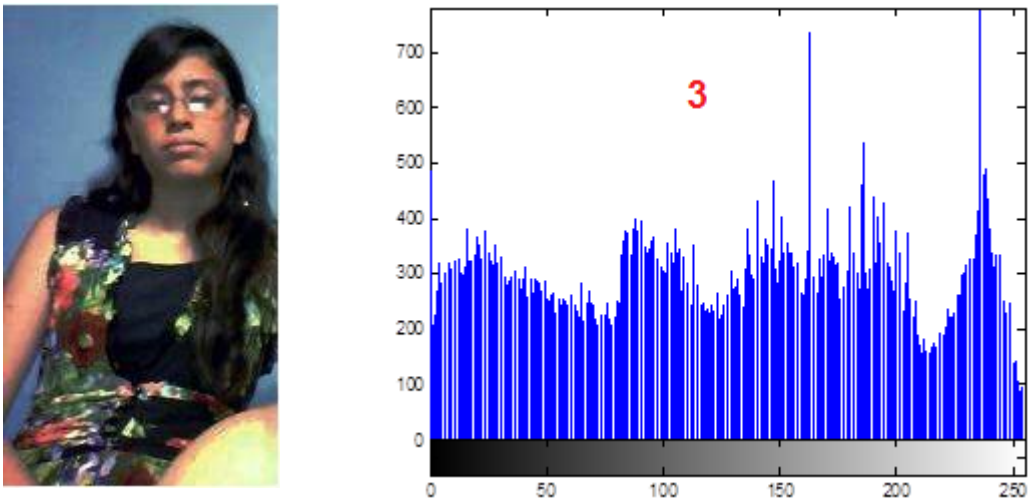


Figura 5. 20: Histograma de imagen original ecualizada.

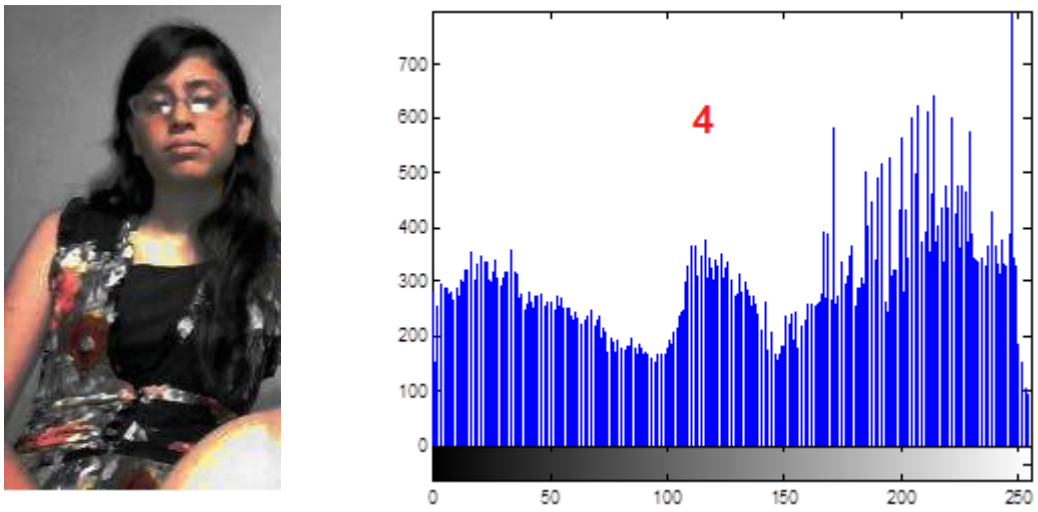


Figura 5. 21: Histograma de imagen ecualizada y encendida.

Podemos apreciar en esta ocasión que el encendido del color rojo de la imagen ecualizada dio mejores resultados, no dejando pasar los demás colores. Viendo los histogramas podemos darnos cuenta que difieren un poco uno del otro, ya que la imagen original al ser ecualizada, ciertas propiedades y tonalidades de los pixeles cambiaron (colores más claros, obscuros, opacos, etc), y por esta razón es que ciertos pixeles no han sido detectados y encendidos.

5.5 ANALISIS DE HISTOGRAMAS DE LA SEGMENTACIÓN POR BORDES

Para este análisis, vamos a trabajar con el video "Aeropuerto.avi" (grabado con una webcam), sus características las podemos ver en la Tabla IV.

CARACTERISTICAS DEL VIDEO DE PRUEBA	
Nombre del archivo	Aeropuerto.avi
Numero de cuadros	150
Cuadros por segundo	15
Dimensiones	352 x 240
Compresión de video	Ninguna
Tipo de imagen	Color Verdadero (24 bits)

Tabla IV

El proceso base para una correcta segmentación de bordes es la detección de bordes. Le aplicamos al video el algoritmo de detección por bordes de Canny y

tomamos sus histogramas para la imagen normal (1) e invertida (2). La imagen original se muestra en la figura 5.22., y detectados los bordes en la Fig. 5.23.

Figura 5. 22: Cuadro n°1 del video "Aeropuerto.avi"

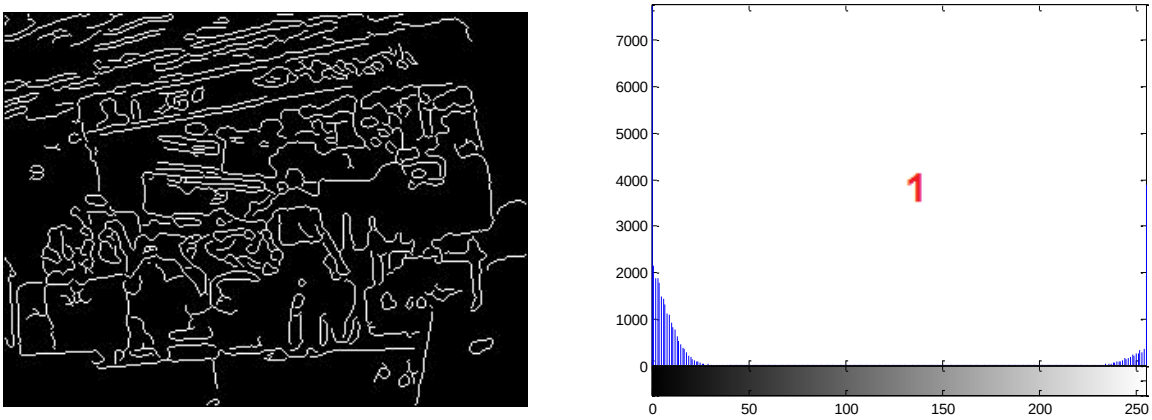


Figura 5. 23: Histograma de Aeropuerto.avi después de la detección de Canny

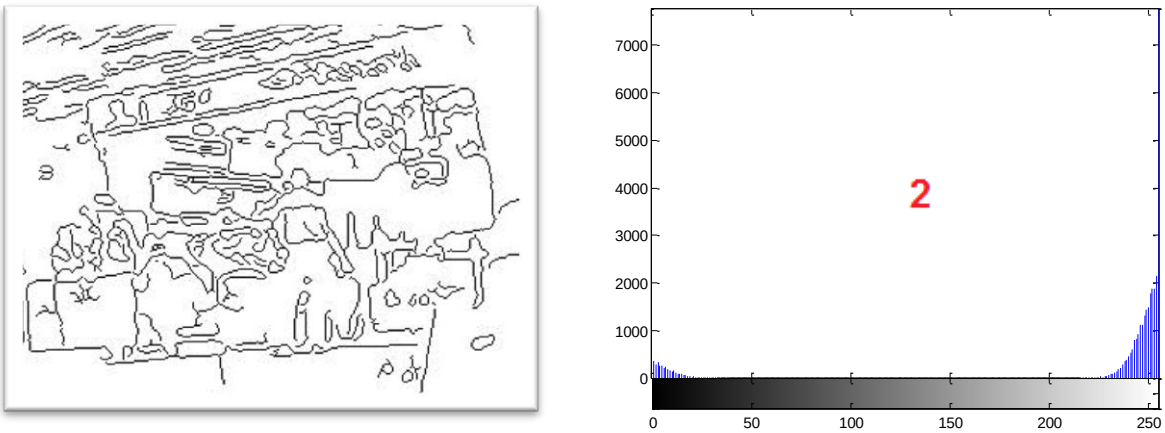


Figura 5. 24: Histograma de la imagen invertida después de la detección de Canny

En las dos imágenes podemos observar gran cantidad de bordes detectados, lo cual era de esperarse viendo la imagen original. La mayoría de figuras y formas se

pueden reconocer en las imágenes. En el histograma (1) vemos gran cantidad de pixeles negros (fondo de la imagen negro) y pocos pixeles blancos (bordes detectados), y en el histograma (2) todo lo contrario.

Esperando tener una mejor apreciación de la imagen (visión completa de las figuras), se procede a ecualizar el video y tomar los histogramas correspondientes a la imagen normal (3) e invertida (4) nuevamente. Fig. 5.25 y Fig. 5.26.

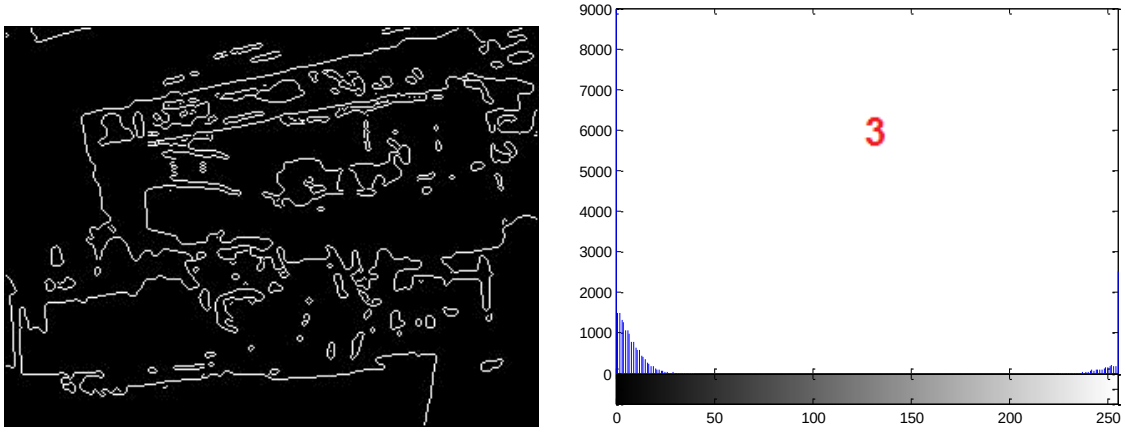


Figura 5. 25: Histograma del Cuadro 1 luego de la detección y el Binarizado

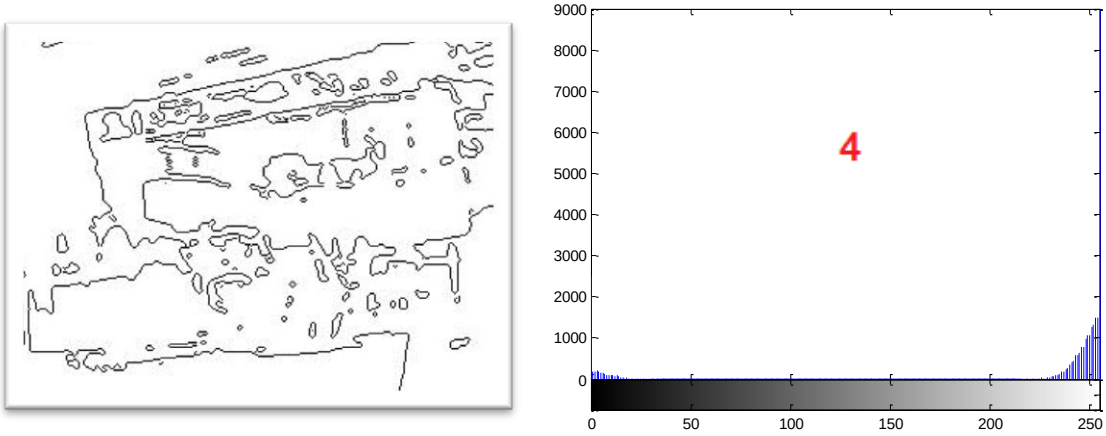


Figura 5. 26: Histograma luego de la detección y el Binarizado

En este caso en particular no se tuvieron buenos resultados, ya que la mayoría de figuras y formas en las imágenes son irreconocibles. En este ejemplo, para seguir con la segmentación no sería necesario binarizar, sino sólo detectar los

bordes con Canny. Los histogramas en este caso muestran menor concentración de pixeles negros (3) y blancos (4), como producto de que ha habido una menor detección de bordes en las imágenes.

5.6 ANALISIS DE HISTOGRAMAS DE LA SEGMENTACIÓN POR TEXTURAS

Procedemos igual, escogemos el video "Banner.avi" (grabado con una webcam), tomamos un cuadro y sacamos los histogramas antes y después de la segmentación. Ver Fig. 5.27.

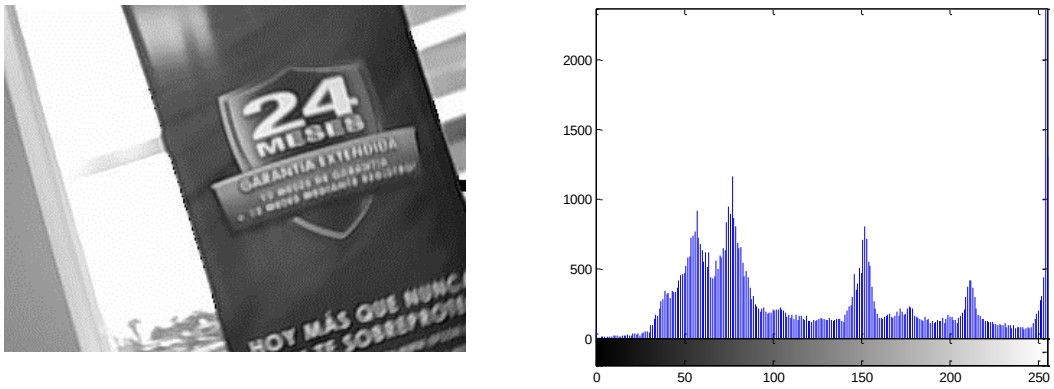


Figura 5. 27: Histograma de imagen original del video "Banner.avi"

Luego se procedió a realizar la segmentación de texturas variando tanto su área de objeto (cantidad de pixeles adyacentes de una misma textura o tonalidad) como su valor de máscara (intensidad en escala de grises). Fig. 5.28 y 5.29

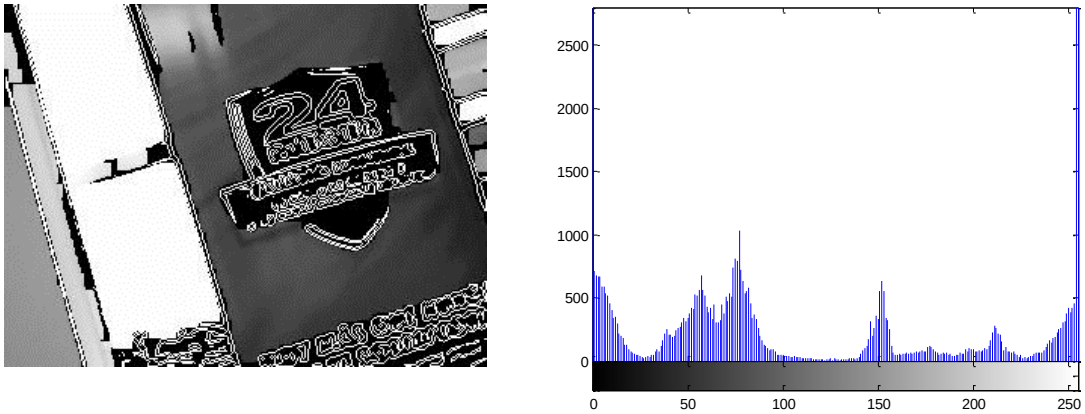


Figura 5. 28: Histograma banner.avi segmentado con Int=0.1; Area=10

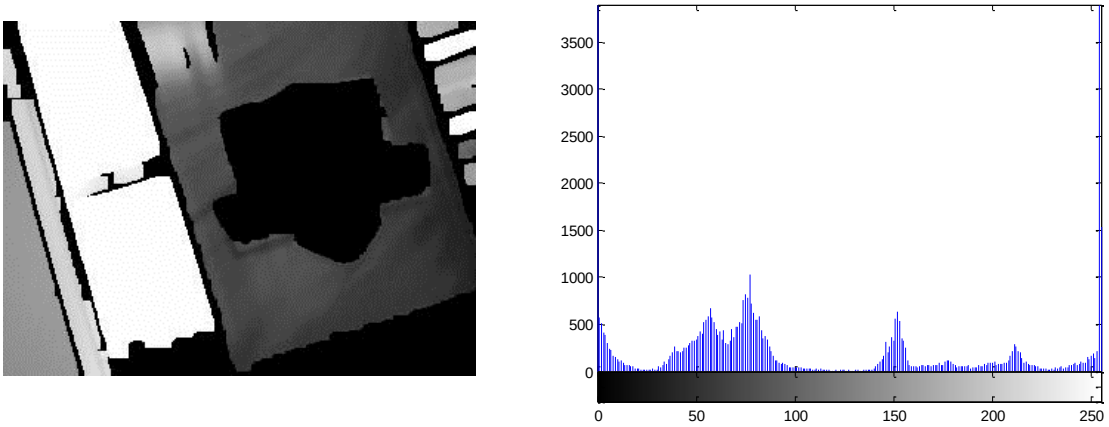


Figura 5. 29: Histograma del cuadro n°1 segmentado. I=0.1; A=3000

En la Fig. 5.28 se segmentaron áreas pequeñas, y a estas áreas se les puso una máscara negra. En la Fig. 5.29 se segmentaron áreas grandes con la misma máscara de la figura anterior. En los histogramas se observa una mayor segmentación de texturas en áreas pequeñas (mayormente por la segmentación de las letras existentes en la imagen).

Luego hicimos la ecualización por contraste del video y tomamos los histogramas correspondientes a cada imagen normal (Fig. 5.30); máscara de región dividida y ecualizada Fig. 5.31 y máscara de región completa ecualizada Fig. 5.32.

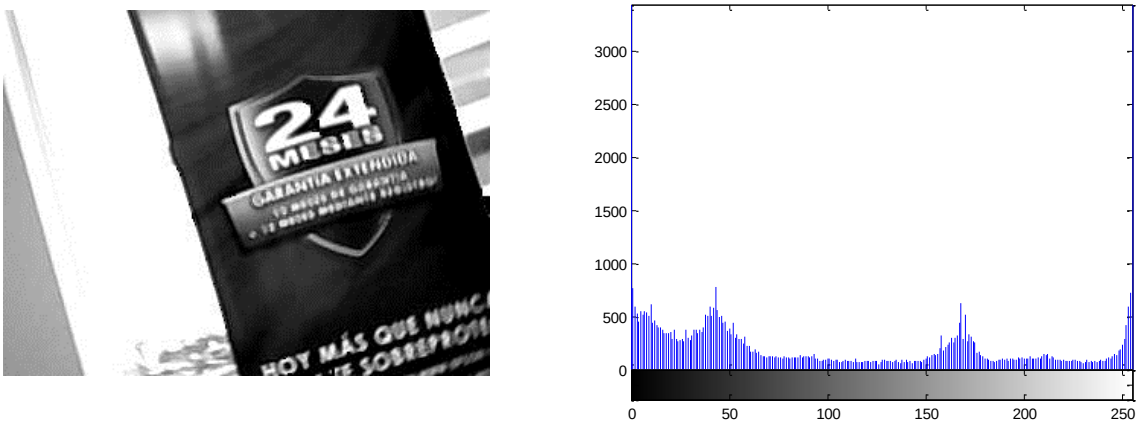


Figura 5. 30: Histograma del cuadro n°1 luego del ajuste de contraste.

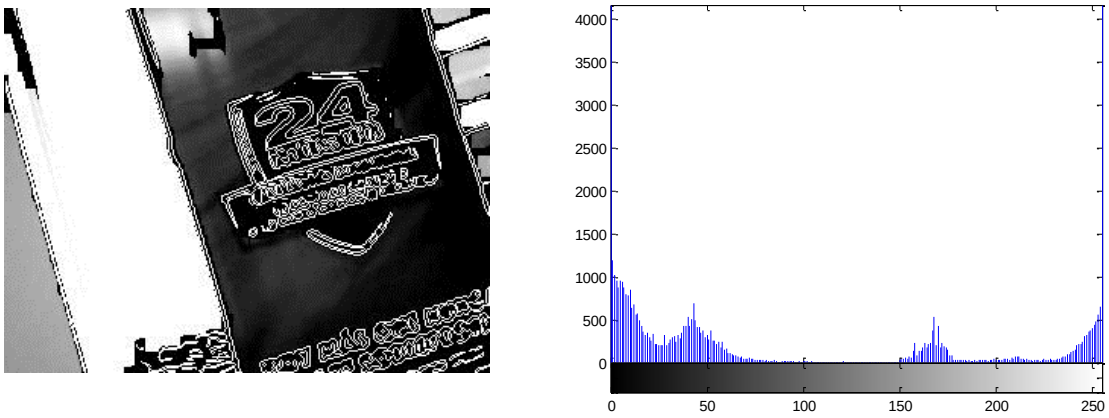


Figura 5. 31: Histograma del cuadro luego del ajuste de contraste. I=0.1; A=10

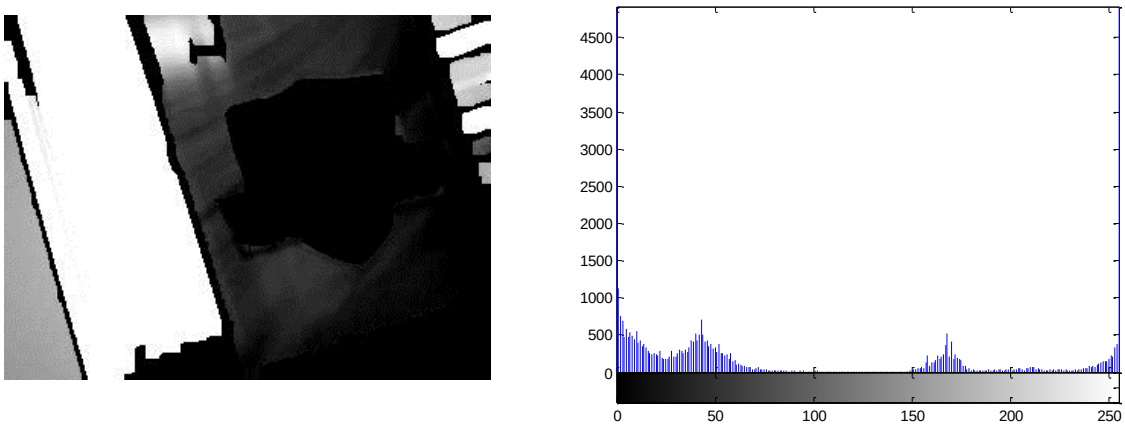


Figura 5. 32: Histograma del cuadro y ajuste de contraste. I=0.1; A=3000

La imagen contrastada de la Fig. 5.30 ahora tiene tonalidades más obscuras que la imagen original, con lo que se pudo segmentar más texturas en áreas pequeñas (Fig. 5.31) comparada con la segmentación de la imagen original (Fig. 5.28). En el histograma de la Fig. 5.32 hay una mayor concentración de pixeles negros, producto de que las áreas grandes que tienen una misma textura son mayores, debido al obscurecimiento de los pixeles por parte del ecualizador contraste.

5.7 ANALISIS DE RENDIMIENTO DE LOS METODOS DE SEGMENTACION

Para realizar este análisis hicimos la siguiente prueba, trabajamos con el video "Fiesta.avi", y luego medimos el tiempo que tomaba realizar el procesamiento de cada cuadro del video durante cada método segmentación. La medición la

tomamos para 10, 50, 100 y 200 cuadros (frames). Después realizamos las mismas mediciones pero en otros equipos con distinto procesador y memoria, con el fin de poder comparar los resultados. Los equipos utilizados en las pruebas los resumimos en la Tabla V.

MODELO / CARACTERISTICA	HP 8440p	HP 4420s	HP 530
MEMORIA RAM(GB)	4	3	2
PROCESADOR	Core i5 - 2.4 Ghz	Core i3 - 2.4 Ghz	Core2duo-1.67 Ghz
DISCO DURO(GB)	500	250	100

TABLA V

El primer equipo que usamos para las pruebas fue un computador portátil HP Elite 8440p. Los métodos de segmentación de borde y textura trabajaron rápidamente con una velocidad aproximada de 10 cuadros/segundo. Mientras que la segmentación por color alcanzo una velocidad 1 cuadro/segundo. Fig. 5.33.

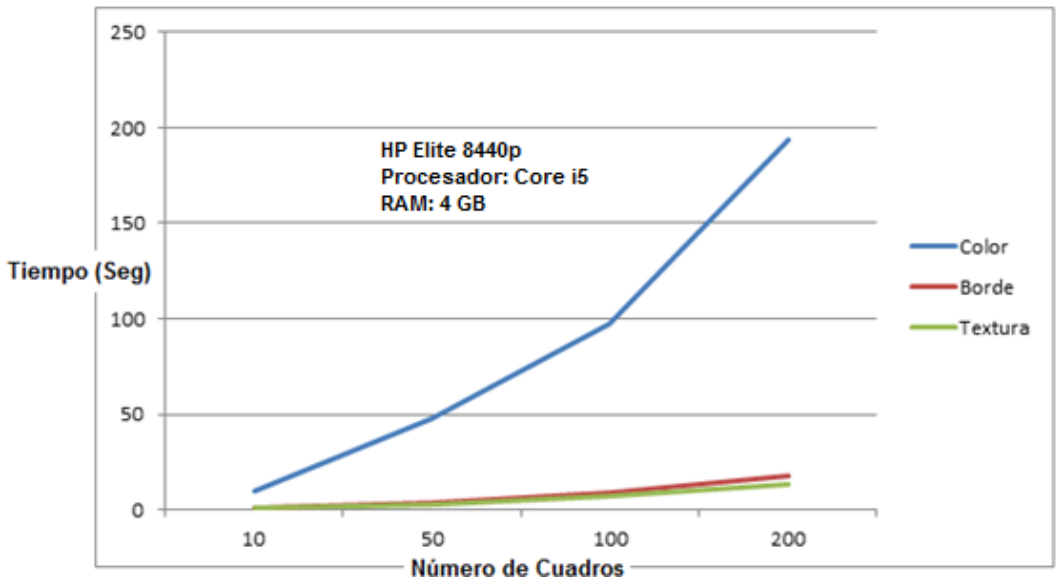


Figura 5. 33: Velocidad de procesamiento por cuadro - Portátil HP Elite 8440p

En el segundo equipo HP 4420s, la textura trabajo a 12 cuadros/segundo; Los bordes a 10 cuadros/segundo y la de colores a 0.8 cuadros/segundo. Fig. 5.34.

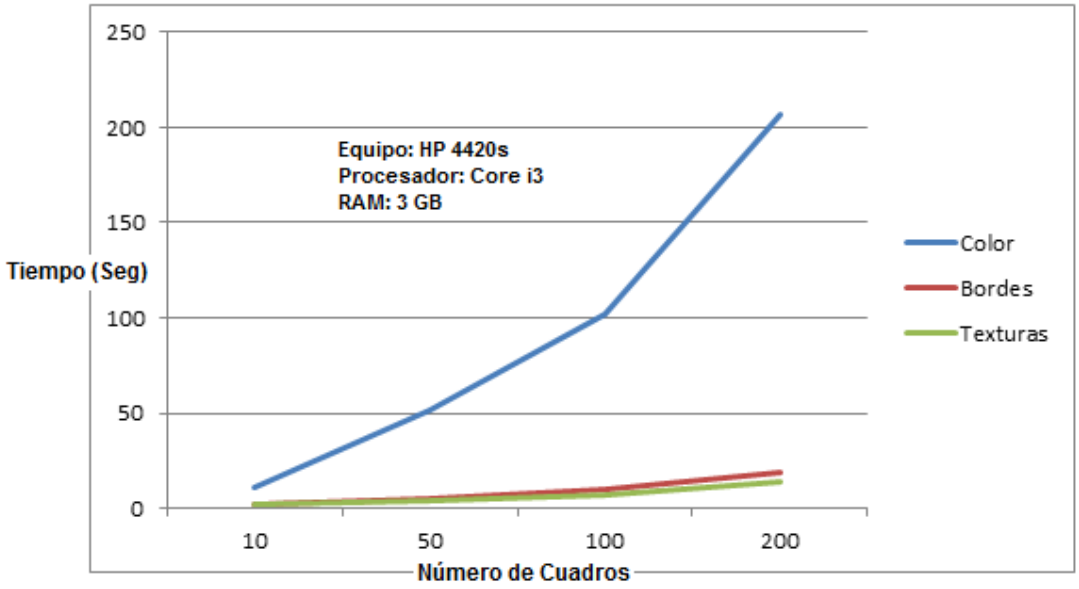


Figura 5. 34: Velocidad de procesamiento por cuadro - Portátil HP 4420s.

El último equipo que usamos para medir la velocidad de procesamiento por cuadro de los métodos de segmentación fue un HP 530; en este equipo registramos las siguientes velocidades. Texturas 5 cuadros/segundo, Bordes 4 cuadros/segundo y colores 0.3 cuadros/segundo. Ver Fig. 5.35.

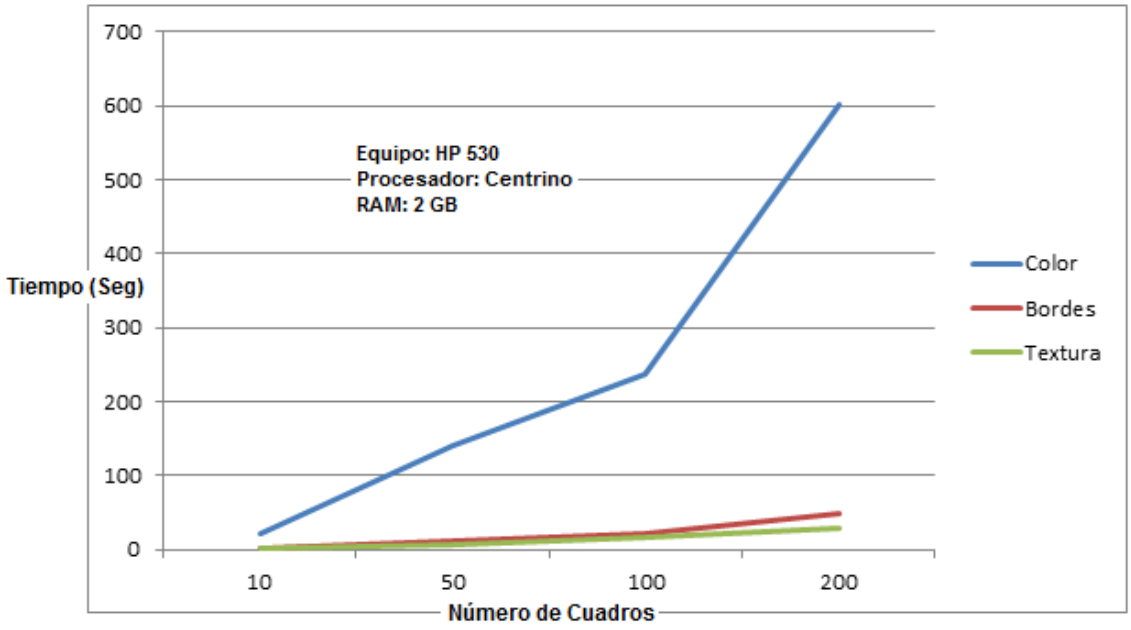


Figura 5. 35: Velocidad de procesamiento por cuadro - Portátil HP 530.

5.8 RESULTADOS ENCUESTAS DE PERCEPCION VISUAL

En esta encuesta fueron entrevistadas 50 personas con la siguiente distribución:

Categoría (15 - 20) = 9 personas - 4 hombres y 5 mujeres

Categoría (21 - 40) = 29 personas - 20 hombres y 9 mujeres

Categoría (41 - 60) = 12 personas - 6 hombre y 6 mujeres

Luego el 72% de los encuestados escogieron a la segmentación por color como el método más útil entre los tres (ya sea esta utilidad para su trabajo, entretenimiento, ayuda para comunidad científica, etc). Ver Fig. 5.36.

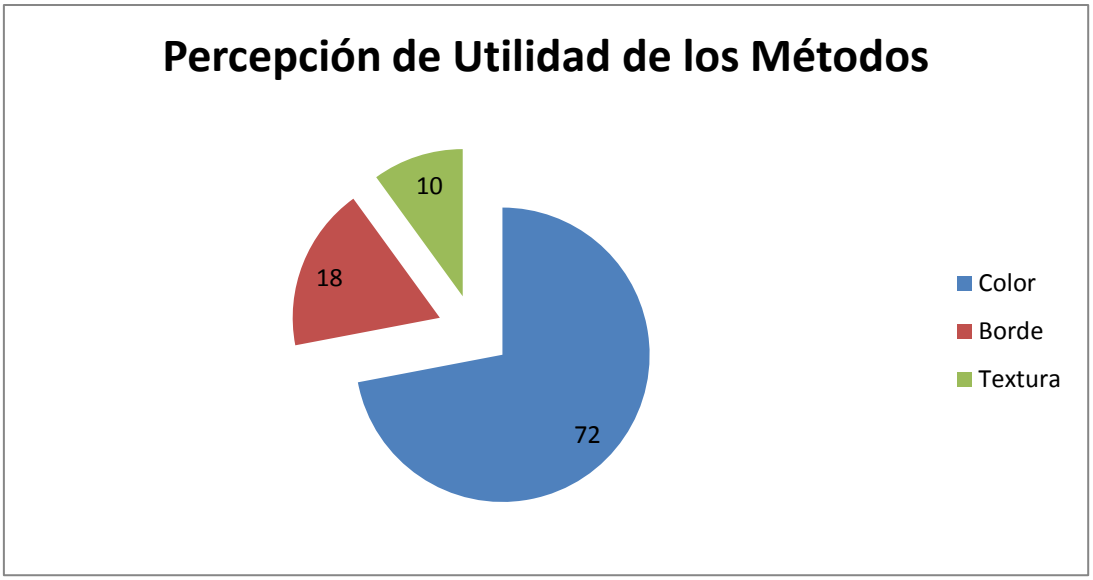


Figura 5. 36: Utilidad de los Métodos de segmentación.

Cuando preguntamos a las personas qué métodos usarían con fines de entretenimiento, la segmentación por color fue la más votada. En la categoría comercial, la segmentación por bordes fue la escogida. En la categoría profesional también fue más votada la segmentación por color, con una votación importante para la segmentación por texturas. Y finalmente en la categoría profesional estuvieron parejos los métodos texturas y color. Ver Fig. 5.37

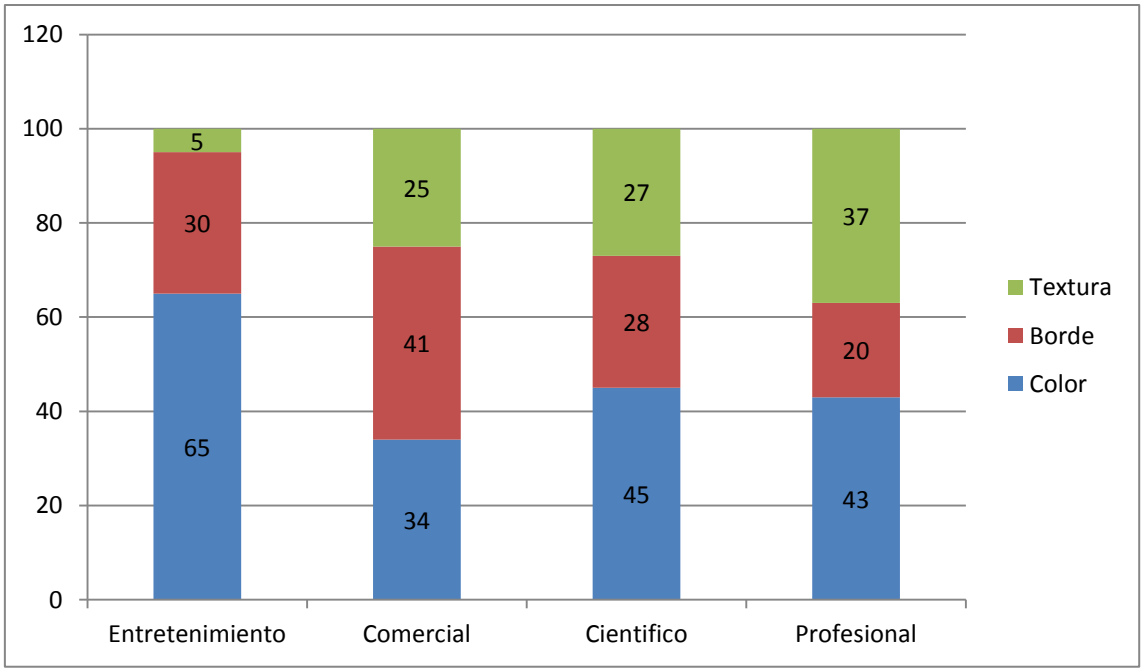


Figura 5. 37: Uso de los métodos por categorías

El 80% de los encuestados opinaron que el manejo de la herramienta no era tan sencillo. Para las encuestas se tuvo que supervisar a las personas para que puedan usar correctamente la aplicación y puedan apreciar las diferencias entre cuadros y videos originales y segmentados. Ver Fig. 5.38.

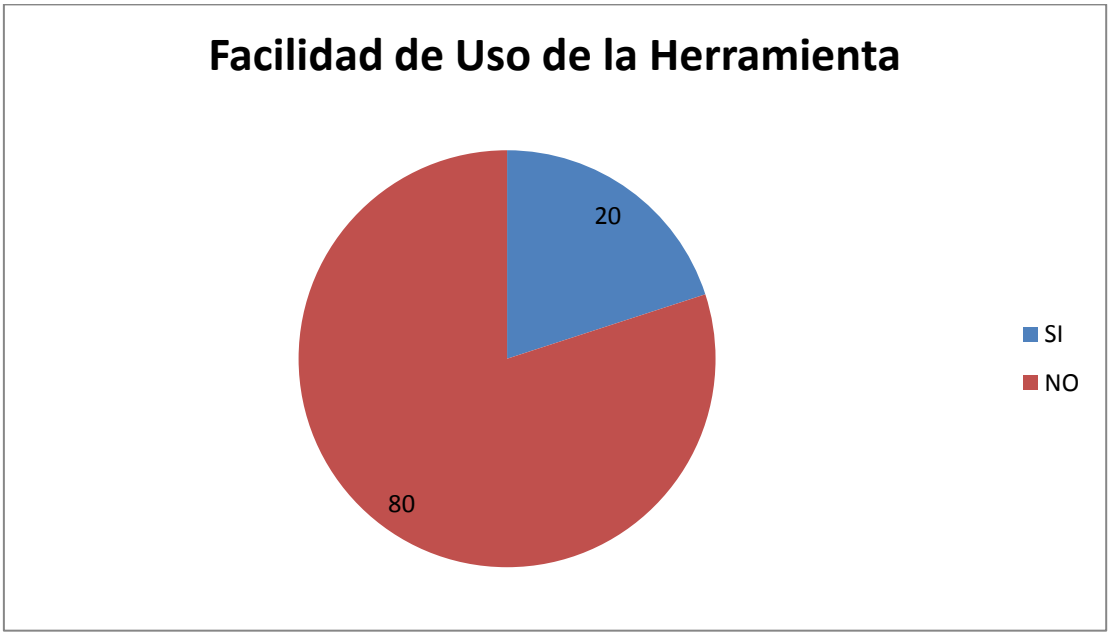


Figura 5. 38: Facilidad de uso de la Herramienta

Sobre la pregunta que mejoraría usted de la interfaz, la mayoría respondió que debíamos rediseñar la interface, haciendo más grandes las salidas visuales y añadiendo más colores a la segmentación por color. Lamentablemente esto no se pudo implementar porque añadir más colores involucra mayor procesamiento (mayores tiempos de respuesta) debido a que las segmentaciones no se las puede hacer por separado (un color a la vez). Ver Fig. 5.39.

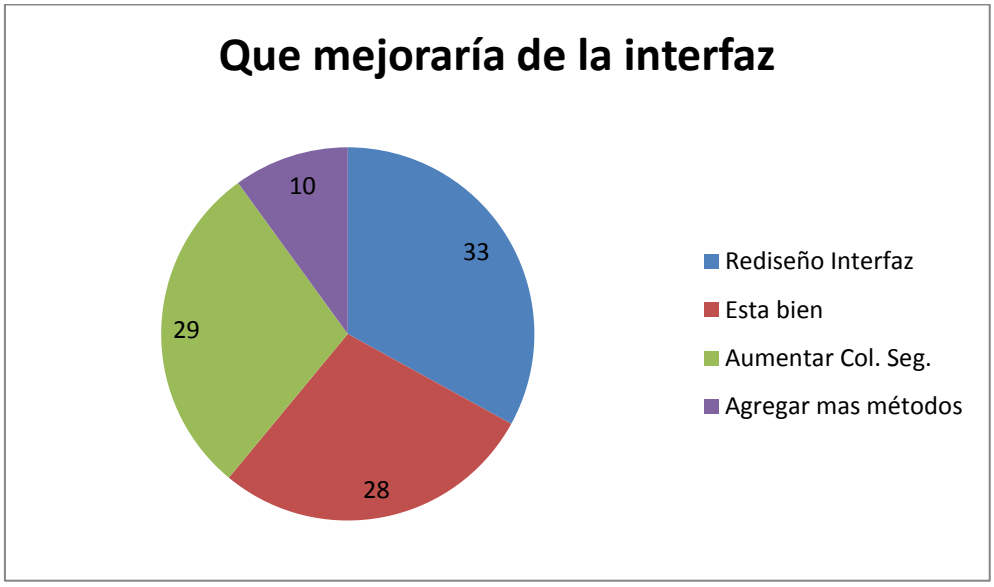


Figura 5. 39: Recomendaciones de los encuestados

El 55% de los encuestados dijo que la calidad del video segmentado (nitidez, pixeleo, intensidad, etc) es igual al de los videos de entrada. Ver Fig. 5.40.

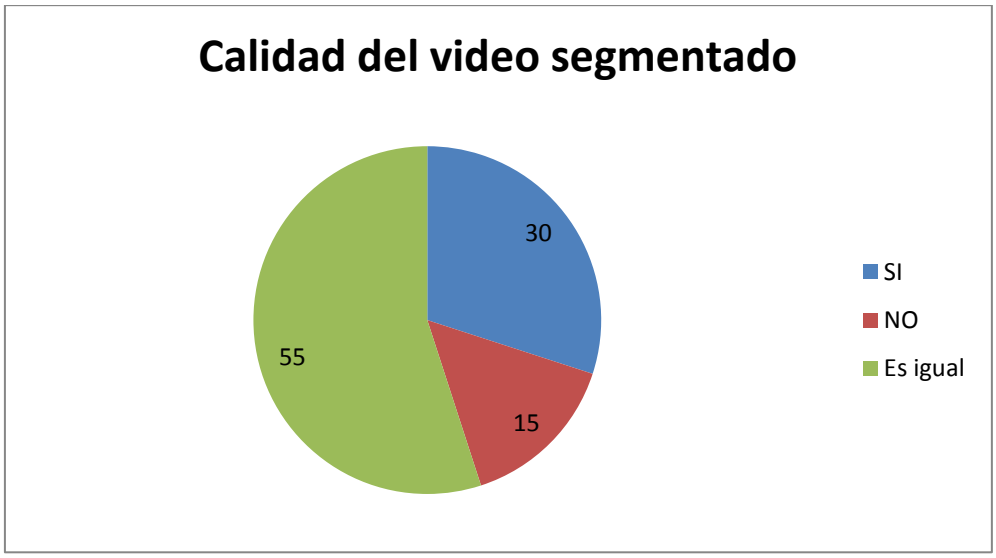


Figura 5. 40: Calidad del video de salida versus el de entrada

Sobre cuál de los ecualizadores le agregó más valor a su método, la respuesta fue contundente, el ecualizador de color con el 82% de las respuestas. Ver Fig. 5.41.

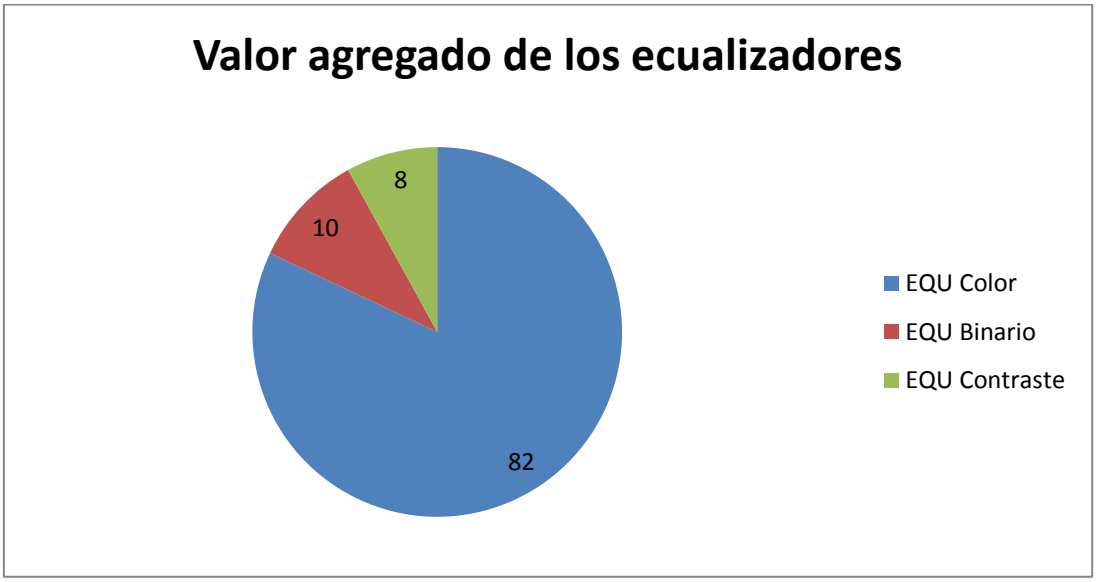


Figura 5. 41: %Percepción de utilidad de los Ecualizadores

En general, el 90% de los encuestados opino que la herramienta de segmentación no demora en segmentar los videos cuadro por cuadro. Para el tipo de video que usamos en las pruebas. Ver Fig. 5.42.

Figura 5. 42: Respuesta a la pregunta si la segmentación lenta o rápida.

También se preguntó al grupo que consideraban ellos más útil, la segmentación por imágenes independientes o la segmentación de imágenes (cuadros o frames) en secuencia para modificar todo un video. Los encuestados respondieron de la siguiente manera. 60% dijo que preferían la segmentación del video completo. Mientras que el otro 40% corresponde a la gente que piensa que mejor es segmentar imagen por imagen. Fig. 5.43.

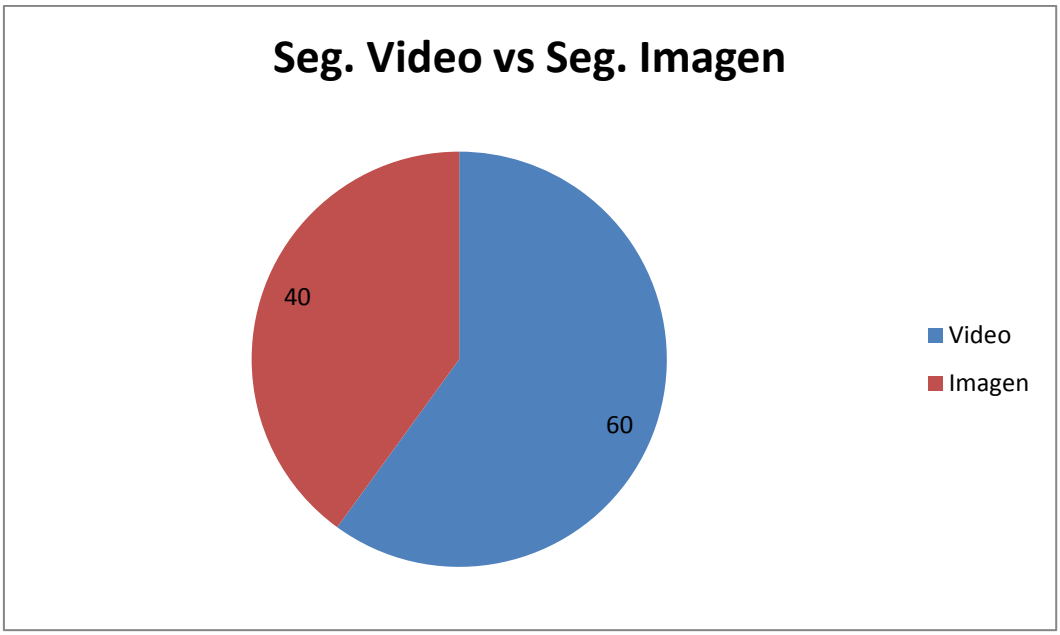


Figura 5. 43: Preferencias segmentación de imágenes versus videos

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

1. Luego de terminado el proyecto de segmentación de videos, hemos concluido que de los 3 métodos implementados, el que mejor aplica para trabajar con video digital es el de segmentación por color. Ya que el resultado que este método entrega, es de mucho valor e impacto visual al permitir el apagado y encendido de colores tradicionales como rojo, verde y azul en áreas grandes y pequeñas como quedó demostrado en el video "Vestido_colores.avi". Además, este método presenta muchas posibilidades de usarlo en aplicaciones comerciales y de entretenimiento como lo manifestaron el 72% de los encuestados.
2. En cambio de los otros 2 métodos, el que llamó más la atención a los encuestados fue el de segmentación por bordes pero con la imagen invertida. Aproximadamente un 30% de las personas le encontraron la utilidad de aplicar el resultado de esta segmentación en videos de caricaturas, que son muy usados para propagandas comerciales y videos con fines de entretenimiento. El video con detección de bordes por Canny en su forma original no les llamó la atención.
3. En el análisis del rendimiento de la herramienta de segmentación, concluimos que esta fue bastante estable en su funcionamiento, sobre todo cuando los videos no superaban los 600 o 700 cuadros. Esto se debe a que en videos largos o muy comprimidos (de miles de cuadros) la segmentación de las imágenes en Matlab involucra mucho procesamiento, agotando la memoria RAM de la computadora. Si se quisiere segmentar videos que tienen miles de cuadros, hay que tomar en cuenta esta limitante (tener una mayor cantidad de memoria disponible en el sistema) y contar con un buen procesador. Para equipos con menos de 2GB, solo se pudieron leer hasta

500 cuadros. Para leer videos con más de 1000 cuadros se necesitan al menos de 3 a 4 GB de RAM.

4. El uso de los ecualizadores nos ayudó bastante, especialmente en el método de segmentación por colores ya que como hemos visto en el reporte, algunas veces tuvimos problemas con la segmentación de colores y tonalidades parecidas, también en áreas pequeñas, o con imágenes multicolor, pero una vez aplicado el ecualizador al video completo la detección tuvo mejores resultados. En el caso de la segmentación por Bordes, el ecualizador binario ayuda quitando el ruido y exceso de bordes luego de la segmentación, pero en cambio se pierde mucho la fidelidad de la imagen original. En el método de las texturas el ecualizador de ajuste de contraste nos ayudó en cambio a atenuar los efectos de intermitencia en las imágenes con mucha variación de movimiento o intensidad.
5. En los histogramas de las imágenes, no se observó mucha diferencia entre el histograma de la imagen original y la segmentada, en ninguno de los tres métodos. Todos estos resultados estaban dentro de lo esperado incluso en la segmentación por color, ya que para obtener el histograma de una imagen RGB, primero la debemos descomponer en sus planos Rojo, Verde y Azul, que era muy parecido a lo que hacía la segmentación de apagar dos colores y encender el otro. Se pudo observar los cambios que ocurren en la imagen cuando es ecualizada, cambiando las propiedades y características de los pixeles de la imagen, pudiendo con esto segmentar zonas que no pudieron ser detectadas en la imagen original.
6. La velocidad de segmentación de un video depende de algunos factores, como lo son el método de segmentación utilizado, la memoria RAM y el procesador con que cuenta el computador, la cantidad de cuadros que componen el video, la compresión del video, y cantidad de programas que se estén ejecutando en el computador (disminuyendo la capacidad disponible de memoria). El método que mayores tiempos de respuesta tuvo

fue el de color, seguido por el de bordes, y el de textura fue el mas veloz; esto se debe a que hay mayor cantidad de procesamiento en el encendido y apagado de todos los pixeles de la imagen (detectar color predominante en cada pixel y cálculos de brillo de luminancia) y luego segmentar las áreas de interés (color deseado).

RECOMENDACIONES

1. Recomendamos que para desarrollar interfaces gráficas en Matlab que usen imágenes de video, lo hagan con la versión R2010a del producto o superior, ya que esta versión maneja mucho mejor las características del video, tanto en la reproducción como en la toma de cuadros y también en las funciones de Matlab para construcciones de datos de video como *mmreader.*
2. Recomendamos tener precaución con los videos que se descargan por internet, ya que no sabemos como han sido editados o cortados, y la duración de tiempo sobre todo de los cuadros finales puede ser diferente a la de los demás. Este problema lo notamos cuando al leer el video daba como resultado un numero X de cuadros, pero al tratar de extraer sus cuadros, el número no coincidía y esto provocaba que el Matlab no lo pueda leer. Este problema lo tuvimos con el video "Ambulancia.avi", el cual al ser leído dice que tiene 120 cuadros, pero al extraer sus cuadros dice que tiene 121 y se produce un error. Para resolver este problema con videos que presenten una diferencia de uno o dos cuadros respecto de los cuadros leídos, implementamos un casillero, denominado "Ajuste de Frames", en el cual ingresamos el valor de la diferencia entre los cuadros extraídos y leídos.
3. Respecto a la velocidad de procesamiento de cuadros por segundo en cada método de segmentación, las pruebas realizadas en la sección 5.3; indicaron que mientras mejor procesador tenga el equipo más rápido es el

procesamiento de los cuadros. Si se quiere trabajar con videos cortos (hasta 600 cuadros) es suficiente contar con una memoria RAM de 2GB en el computador, caso contrario se recomienda aumentar la capacidad a 3 o 4GB para realizar una lectura y extracción completa de los cuadros del video.

4. La recomendación para la siguiente versión de esta herramienta, es que si se quisiere incrementar el número de colores a segmentar de un video, cada segmentación (color) debería manejarse por separado y no de un solo paso como se implementó actualmente, reduciendo drásticamente los tiempos de procesamiento.
5. El método de segmentación por texturas nos pareció que es un método para aplicarlo más a nivel de imagen individual que para sucesión de cuadros de un video. La recomendación es que no apliquen la segmentación por texturas a todo un video, sino a una foto, imagen o seleccionar un cuadro especifico de un video para solo a éste segmentarlo por texturas, pudiendo de esta manera tener una mayor apreciación de las diferentes texturas que se encuentran en la imagen. En un video, al ser una sucesión de imágenes, no se podrán distinguir muy bien las texturas ya que en cada cuadro habrán diferentes regiones segmentadas.

ANEXO A

MANUAL DEL USUARIO

1.1 Prerrequisitos del Sistema

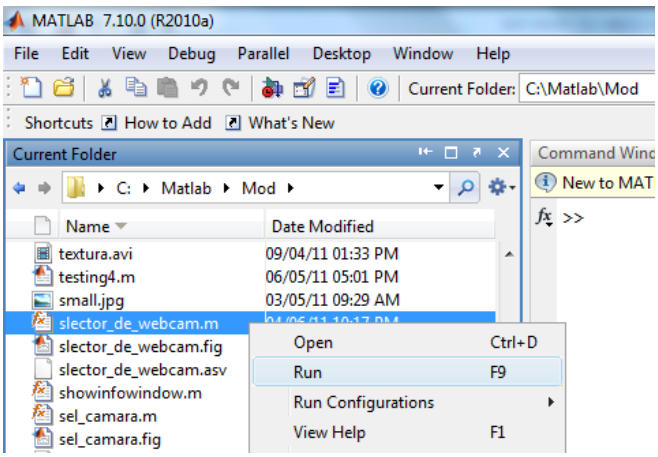
Para utilizar la herramienta de Segmentación de video en una computadora, son necesarios los siguientes prerrequisitos del sistema:

- Windows XP professional SP2 o Windows 7 professional SP1
- 2 GB de memoria RAM mínimo; recomendado 4 GB
- 40 GB de espacio en disco
- Tener instalado Matlab R2010a
- Crear un directorio de aplicación; ejemplo C:\MatLab\MOD
- Copiar los siguientes archivos en el directorio de trabajo. Los archivos se encuentran en el CD de trabajo.
    - Slector_de_Webcam.m;
    - Slector_de_Webcam.fig;
    - sel_camara.m;
    - sel_camara.fig;
    - Showinfowindow.m;

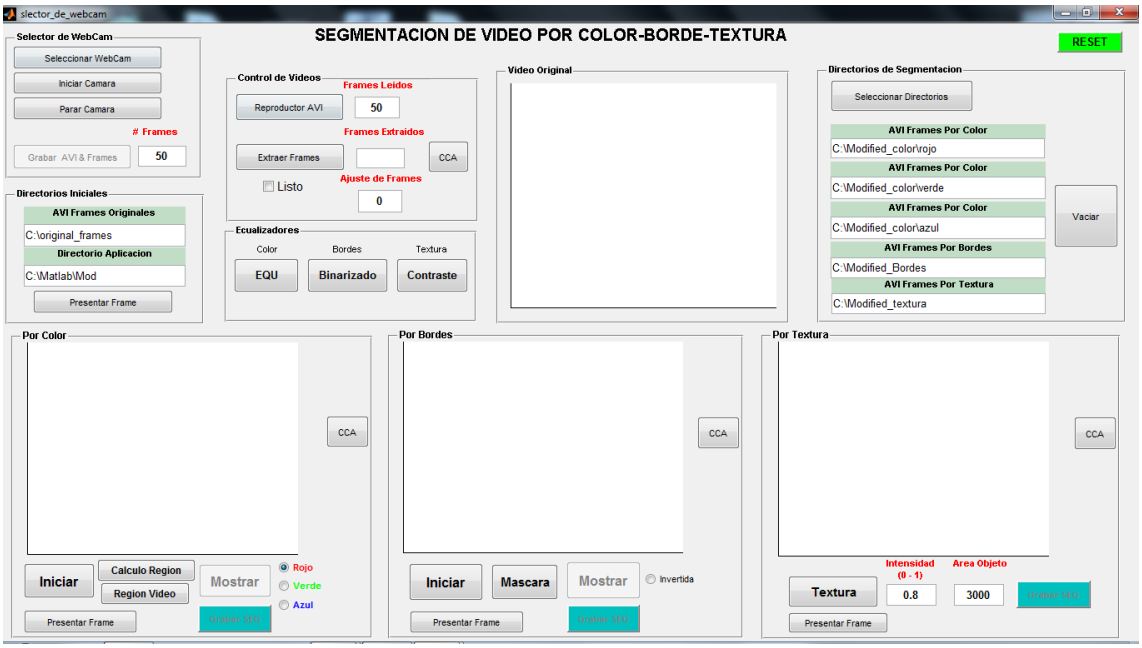
1.1.1 Lanzamiento de la Aplicación

Para arrancar la herramienta de segmentación seguimos los siguientes pasos:

- Cargar el programa Matlab R2010a
- Dentro de Matlab ingresamos al directorio de la aplicación y ejecutamos el archivo: Slector_de_Webcam.m



- Esperamos unos segundos y nos aparece el tablero de control



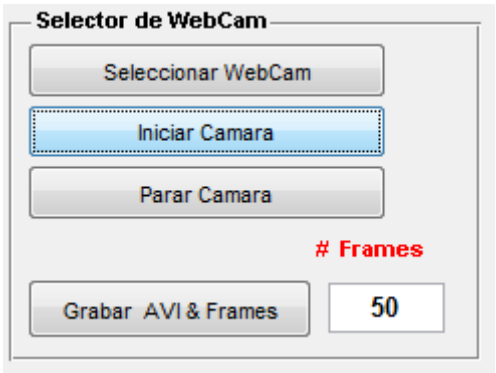
1.1.2 Cierre de la Aplicación

- Para el cierre de la aplicación, podemos hacerlo con las teclas Alt+F4 ó cerrando la ventana.

1.1.3 Como reinicializar la Aplicación

- Hemos colocado en la parte superior derecha del tablero de control el botón RESET; Que sirve para cerrar y recargar la aplicación en caso de ser necesario.

1.2 Selector de Cámaras



1.2.1 Botón - Seleccionar Webcam

Nos permite seleccionar la cámara desde la cual vamos a ingresar la señal de video.

1.2.2 Botón - Iniciar cámara

Inicia la toma de video desde la cámara seleccionada.

1.2.3 Botón - Parar cámara

Para la toma de video desde la cámara seleccionada.

1.2.4 Botón - Grabar AVI & Frames

Este botón realiza varias acciones; permite establecer un nombre para el archivo de video que se generará a través de la cámara web, luego toma instantáneas del video de la cámara web y finalmente convierte las instantáneas en cuadros de video para completar el archivo AVI. El número de instantáneas por defecto que toma este botón es de 50.

1.3 Directorios

1.3.1 Directorios iniciales

Los directorios iniciales de la aplicación son los siguientes:

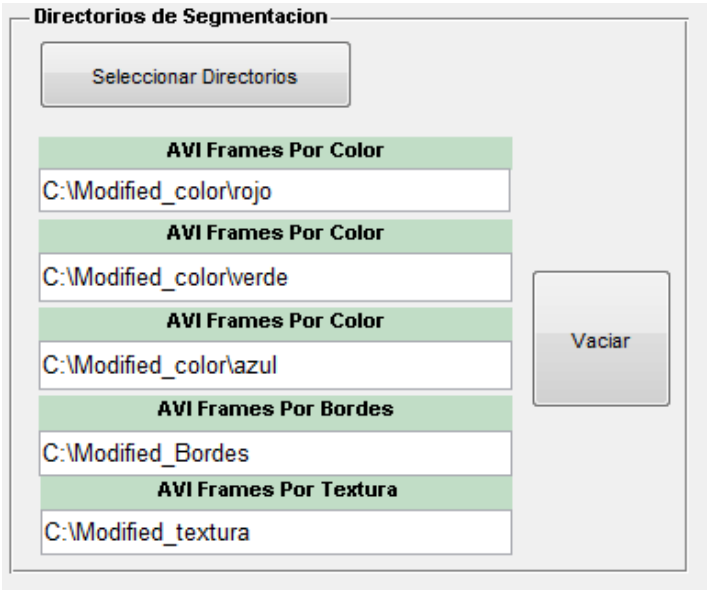
C:\Original_Frames; aquí se almacenan las imágenes originales del video.

C:\Matlab\Mod; Directorio de la aplicación.

1.3.2 Botón - Presentar Frames

Este botón nos permite navegar por el directorio *C: \original_frames,* y seleccionar un cuadro del video para mostrarlo y analizarlo en pantalla.

1.3.3 Directorios de segmentación



1.3.4 Botón - Seleccionar directorios

Los directorios de segmentación se escogen a través del botón Seleccionar Directorios y permanecen así hasta que sean cambiados nuevamente.

1.3.5 Botón - Vaciar

Sirve para borrar todos los archivos de formato JPG que estén dentro de los directorios iniciales y de segmentación.

1.4 Control de Video

1.4.1 Botón - Reproductor AVI

Con este botón podemos reproducir archivos de video en formato AVI y mostrar de cuantos cuadros está compuesto el video en el campo Frames Leídos.

1.4.2 Botón - Extraer Frames

Extrae los cuadros del video de un archivo AVI existente, y los guarda en la carpeta *C:\original_frames*; La cantidad de cuadros extraídos se muestra en el campo Frames Extraídos. Al finalizar la extracción se marca temporalmente con un visto la casilla de verificación LISTO. En el campo ajuste de frames colocamos la diferencia entre los cuadros leídos y los cuadros extraídos cuando esta diferencia es diferente de cero.

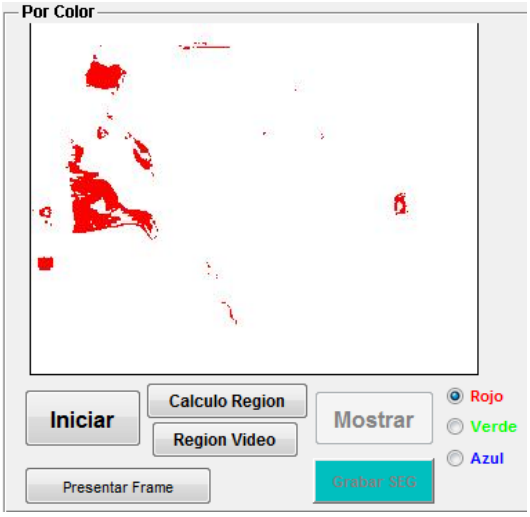
1.4.3 Botón - CCA (Limpiador de pantalla)

Como su nombre lo indica, este botón sirve para limpiar las salidas por pantallas

1.4.4 Salida - Video Original

Aquí se muestra la reproducción del video original antes de ser segmentado.

1.5 Segmentación por Color



1.5.1 Botón - Iniciar

Inicia el proceso de segmentación por color y cuando finaliza habilita al botón Mostrar.

1.5.2 Botón - Mostrar

Con este botón presentamos las segmentaciones realizadas en el video por cualquiera de los tres colores segmentados, rojo, verde o azul.

1.5.3 Botón - Presentar Frame

Nos permite navegar por los directorios que tienen los resultados de la segmentación por color y seleccionar un cuadro para mostrar el resultado por pantalla.

1.5.4 Botón - Grabar SEG

Permite grabar las salidas por pantalla en 3 archivos uno por cada color segmentado.

1.5.5 Salida - Por color

Aquí se muestran los videos segmentados uno por cada color.

1.5.6 Botón - CCA (Limpiador de pantalla)

Limpia y deja en blanco la salida para mostrar otros videos o imágenes.

1.5.7 Botón - Cálculo de Región

Calcula la región de interés de la imagen a partir del color especificado.

1.5.8 Botón Región Video

Muestra el resultado del cálculo de la región de interés.

1.6 Segmentación por Borde

1.6.1 Botón - Iniciar

Con este botón iniciamos la segmentación por bordes.

1.6.2 Botón - Mostrar

Nos muestra la segmentación por bordes de Canny en su forma normal e invertida.

1.6.3 Botón - Presentar Frame

Nos permite navegar por el directorio *C:\Modified_bordes* para seleccionar y mostrar los cuadros extraídos

1.6.4 Botón - Grabar SEG

Graba dos archivos AVI uno con Canny normal y otro con Canny invertido.

1.6.5 Salida - Por borde

Aquí se reproducen las segmentaciones por borde.

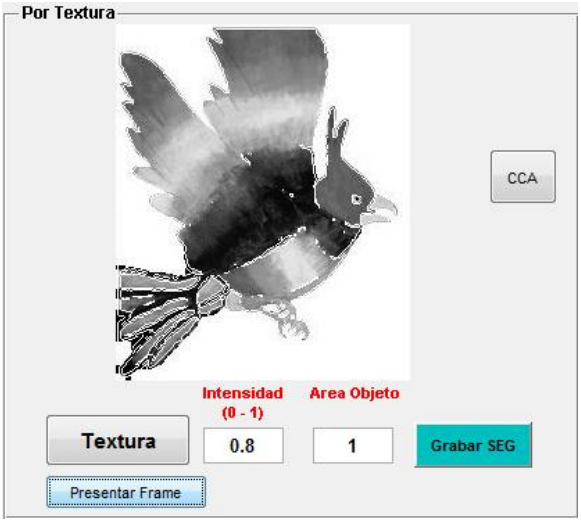
1.6.6 Botón - CCA (Limpiador de pantalla)

Permite limpiar y vaciar el último contenido cargado en la pantalla.

1.6.7 Botón Máscara

Este botón nos permite ingresar una máscara binaria de forma libre sobre el primer cuadro del video en análisis, de manera que en el resto de las imágenes se vaya procesando también esta misma máscara.

1.7 Segmentación por Textura



1.7.1 Botón - Textura

Inicia el proceso de segmentación por texturas, y muestra por pantalla directamente el resultado de la segmentación.

1.7.2 Botón - Presentar Frame

Nos permite navegar por el directorio C*:\Modified_Textura*; y seleccionar un cuadro para mostrar el resultado de la segmentación.

1.7.3 Botón - Grabar SEG

Graba el resultado de la segmentación en un archivo AVI.

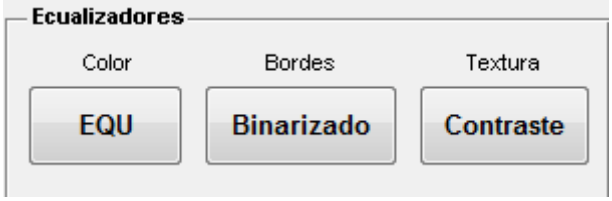
1.7.4 Salida - Por textura

Aquí se reproduce el video segmentado por texturas.

1.7.5 Botón - CCA (Limpiador de pantalla)

Permite limpiar y vaciar el último contenido cargado en la pantalla.

1.8 Ecualizadores



1.8.1 Botón - EQU

Equaliza las imágenes originales ubicadas en *C:\Original_Frames*. Resaltando la intensidad de los colores.

1.8.2 Botón - Binarizado

Binariza el video, pone las imágenes originales en blanco y negro, para luego hacer la detección por bordes de Canny.

1.8.3 Botón - Contraste

Se aplica un ajuste de contrate al video segmentado por texturas.

ANEXO B

CODIGO MATLAB - (Extractos)

% Ajustar Condiciones Iniciales del programa

```
movegui(hObject,'center')
[SUCCESS,MESSAGE]= mkdir('C:\Matlab\Mod');
[SUCCESS,MESSAGE]= mkdir('C:\original_frames');
[SUCCESS,MESSAGE]= mkdir('C:\Modified_Bordes');
[SUCCESS,MESSAGE]= mkdir('C:\Modified_Color\rojo');
[SUCCESS,MESSAGE]= mkdir('C:\Modified_Color\verde');
[SUCCESS,MESSAGE]= mkdir('C:\Modified_Color\azul');
[SUCCESS,MESSAGE]= mkdir('C:\Modified_textura');
set(handles.grabadora,'Enable','off')
set(handles.show_canny,'Enable','off')
set(handles.show_color,'Enable','off')
set(handles.Grabar_Canny,'Enable','off')
set(handles.Grabar_Color,'Enable','off')
set(handles.Grabar_Textura,'Enable','off')
set(handles.edit4,'String','C:\original_frames')
set(handles.edit6,'String','C:\Modified_Bordes')
set(handles.edit8,'String','C:\Modified_color\rojo')
set(handles.edit10,'String','C:\Modified_color\verde')
set(handles.edit11,'String','C:\Modified_color\azul')
set(handles.edit9,'String','C:\Modified_textura')
set(handles.edit7,'String','C:\Matlab\Mod')
set(handles.axes1,'XTick',[],'YTick',[])
set(handles.axes2,'XTick',[],'YTick',[])
set(handles.axes3,'XTick',[],'YTick',[])
set(handles.axes4,'XTick',[],'YTick',[])
set(handles.checkbox1,'Value',0)
set(handles.rcolor,'Value',1)
addpath('C:\original_frames','C:\Modified_Bordes','C:\Matlab\MOD','C:\
Modified_textura','C:\Modified_color\verde','C:\Modified_color\rojo','
C:\Modified_color\azul')
```

% Iniciar_video.

```
function Iniciar_video_Callback(hObject, ~, handles)
global id
handles.vidobj = videoinput('winvideo',id,'RGB24_320x240');
start(handles.vidobj);
guidata(hObject, handles);
vidRes = get(handles.vidobj, 'VideoResolution');
nBands = get(handles.vidobj, 'NumberOfBands');
hImage = image(zeros(vidRes(2), vidRes(1), nBands), 'Parent',...
handles.axes1);
preview(handles.vidobj,hImage);
set(handles.grabadora,'Enable','on')
```

% Leer un archivo AVI.

```
function Abrir_avi_Callback(hObject, ~, handles)
 [nombre,ruta]=uigetfile('*.avi','Video');
   if ruta==0, return, end
   obj = mmreader(fullfile(ruta,nombre));
   NumberOfFrames=obj.NumberOfFrames;
   video = read(obj);
   for cnt = 1:.5:NumberOfFrames
        the_image=read(obj,cnt);
        imagesc(the_image,'parent',handles.axes1);
```

```
        set(handles.axes1,'XTick',[ ],'YTick',[ ])
        axis image off
        drawnow;
   end
   set(handles.edit3,'String',NumberOfFrames);
```

% Selección de Directorios.

```
function Select_Dir_Callback(hObject, ~, handles)
persistent  Directorio2 Directorio3 Directorio4
[Directorio2]=uigetdir;
if Directorio2==0, return, end
set(handles.edit8,'String',Directorio2);
[Directorio3]=uigetdir;
if Directorio3==0, return, end
set(handles.edit6,'String',Directorio3);
[Directorio4]=uigetdir;
if Directorio4==0, return, end
set(handles.edit9,'String',Directorio4);
set(handles.Presentar_Frame,'Enable','on')
```

% Grabación de entrada webcam en archivo AVI.

```
function grabadora_Callback(hObject, ~, handles)
 [nombre,ruta]=uiputfile('*.avi','Video');
   if ruta==0, return, end
   NF = str2double(get(handles.edit1,'String'));
   preview(handles.vidobj);
cd(get(handles.edit4,'String'));
delete('*.jpg')
showinfowindow('REVISE SI EL NUMERO DE FRAMES INGRESADO ES
CORRECTO','Aviso');
pause(2)
close Aviso
pause(4)
wb = waitbar(0,'Por favor espere...');
for frame=1:NF+1
data = getsnapshot(handles.vidobj);
outputBaseFileName = sprintf('Frame %4.4d.jpg', frame);
outputFullFileName = fullfile(outputBaseFileName);
if frame==NF+1
text(80,20,'Extracción Completa', 'Color',[0 1 0],'FontSize', 15,
'Parent',handles.axes1);
else
data = uint8(data);
recalledMovie(frame) = im2frame(data);
imwrite(recalledMovie(frame).cdata, outputFullFileName, 'jpg');
waitbar(frame/(NF),wb);
end
end
delete(wb);
movie2avi(recalledMovie, fullfile(ruta,nombre), 'compression',
'None');
cd(get(handles.edit7,'String'));
stoppreview(handles.vidobj);
```

% Presentación de cuadros individuales.

```
function Presentar_Frame_Callback(hObject, ~, handles)
cd(get(handles.edit4,'String'));
[nombre2,ruta2]=uigetfile('*.jpg','Frame');
   if ruta2==0, return, end
   PF=imread(nombre2);
   imagesc(PF,'parent',handles.axes1);
   set(handles.axes1,'XTick',[],'YTick',[])
   drawnow;
```

```
   cd(get(handles.edit7,'String'));
```
% Extractor de cuadros de video.
```
function Extract_Callback(hObject, ~, handles)
cd(get(handles.edit4,'String'));
AJ=str2double(get(handles.Ajuste,'String'));
[nombre,ruta]=uigetfile('*.avi;*.mpg','Video');
if ruta==0, return, end
obj = mmreader(fullfile(ruta,nombre));
NumberOfFrames=obj.NumberOfFrames-AJ;
video = read(obj);
for k = 1 : NumberOfFrames
mov1(k).cdata = video(:,:,:,k);
mov1(k).colormap = [];
outputBaseFileName = sprintf('Frame %4.4d.jpg', k);
outputFullFileName = fullfile(outputBaseFileName);
imwrite(mov1(k).cdata, outputFullFileName, 'jpg');
end
set(handles.edit12,'String',NumberOfFrames);
set(handles.edit3,'String',NumberOfFrames);
set(handles.checkbox1,'Value',1)
pause(3)
set(handles.checkbox1,'Value',0)
```
% Inicio de Algoritmo de segmentación por Bordes.
```
function Ini_Canny_Callback(hObject, ~, handles)
cd(get(handles.edit4,'String'));
NF2=str2double(get(handles.edit3,'String'));
wb = waitbar(0,'Por favor espere...');
for frame = 1 : NF2
        outputBaseFileName = sprintf('Frame %4.4d.jpg', frame);
        outputFullFileName = fullfile(outputBaseFileName);
        outputBaseFileName1 = sprintf('Frame%3.3d.jpg', frame);
        outputFullFileName1 = fullfile(outputBaseFileName1);
        thisFrame = imread(outputFullFileName);
        BW=rgb2gray(thisFrame);
        BW1=edge(BW,'canny',0.2,0.8);
        BW2=~BW1;
        cd(get(handles.edit6,'String'));
        imwrite(BW1, outputFullFileName, 'jpg');
        imwrite(BW2, outputFullFileName1, 'jpg');
        waitbar(frame/(NF2),wb);
        cd(get(handles.edit4,'String'));
end
delete(wb);
uiwait(msgbox('Segmentacion Terminada','Informacion','modal'));
set(handles.show_canny,'Enable','on')
```
% Presentacion de segmentacion por Bordes.
```
function show_canny_Callback(hObject, ~, handles)
cd(get(handles.edit6,'String'));
NF2=str2double(get(handles.edit3,'String'));
for frame = 1 : NF2
    if (get(handles.CannyI,'Value'))== get(handles.CannyI,'Max')
        outputBaseFileName1 = sprintf('Frame%3.3d.jpg', frame);
        outputFullFileName1 = fullfile(outputBaseFileName1);
        NewFrame1 = imread(outputFullFileName1);
        NewFrameRotFinal1 = imrotate(NewFrame1,180);
        BI=fliplr(NewFrameRotFinal1);
        grapmap = gray(256);
        recalledMovie(frame) = im2frame(BI, grapmap);
    else
        outputBaseFileName = sprintf('Frame %4.4d.jpg', frame);
```

```
        outputFullFileName = fullfile(outputBaseFileName);
        NewFrame = imread(outputFullFileName);
        NewFrameRotFinal = imrotate(NewFrame,180);
        B=fliplr(NewFrameRotFinal);
        grapmap = gray(256);
        recalledMovie(frame) = im2frame(B, grapmap);
    end
end
hold on
movie(handles.axes3, recalledMovie);
set(handles.Grabar_Canny,'Enable','on')
```

% Vaciado de imagenes.

```
function Vaciado_Callback(hObject, ~, handles)
cd(get(handles.edit4,'String'));
delete('*.jpg')
cd(get(handles.edit8,'String'));
delete('*.jpg')
cd(get(handles.edit9,'String'));
delete('*.jpg')
cd(get(handles.edit6,'String'));
delete('*.jpg')
cd(get(handles.edit10,'String'));
delete('*.jpg')
cd(get(handles.edit11,'String'));
delete('*.jpg')
```

% Inicio de la segmentación por Color.

```
function Ini_Color_Callback(hObject, ~, handles)
cd(get(handles.edit4,'String'));
NF2=str2double(get(handles.edit3,'String'));
wb = waitbar(0,'Por favor espere...');
for frame = 1 : NF2
R=1;
outputBaseFileName = sprintf('Frame %4.4d.jpg', frame);
outputFullFileName = fullfile(outputBaseFileName);
colorFrame = imread(outputFullFileName);
[m n plane] = size(colorFrame);
colorFrame = double(colorFrame);
 C = zeros(m,n,plane);
 factor = max(colorFrame(:)) * 0.2;
 for i = 1:m
 for j = 1:n
 if (colorFrame(i,j,R) > factor && colorFrame(i,j,R) ==
max([colorFrame(i,j,1) colorFrame(i,j,2) colorFrame(i,j,3)]))
 C(i,j,1:3) = colorFrame(i,j,1:3);
 else
 C(i,j,1:3) = (colorFrame(i,j,1) * 0.3) + (colorFrame(i,j,2) * 0.59) +
(colorFrame(i,j,3) * 0.11);
 end
 end
 end
 C = uint8(C);
 cd C:\Modified_color\rojo;
 imwrite(C, outputFullFileName, 'jpg');
 cd(get(handles.edit4,'String'));
R=2;
outputBaseFileName = sprintf('Frame %4.4d.jpg', frame);
outputFullFileName = fullfile(outputBaseFileName);
colorFrame = imread(outputFullFileName);
[m n plane] = size(colorFrame);
colorFrame = double(colorFrame);
 C = zeros(m,n,plane);
```

```
 factor = max(colorFrame(:)) * 0.2;
 for i = 1:m
 for j = 1:n
 if (colorFrame(i,j,R) > factor && colorFrame(i,j,R) ==
max([colorFrame(i,j,1) colorFrame(i,j,2) colorFrame(i,j,3)]))
 C(i,j,1:3) = colorFrame(i,j,1:3);
 else
 C(i,j,1:3) = (colorFrame(i,j,1) * 0.3) + (colorFrame(i,j,2) * 0.59) +
(colorFrame(i,j,3) * 0.11);
 end
 end
 end
 C = uint8(C);
 cd C:\Modified_color\verde;
 imwrite(C, outputFullFileName, 'jpg');
 cd(get(handles.edit4,'String'));
R=3;
outputBaseFileName = sprintf('Frame %4.4d.jpg', frame);
outputFullFileName = fullfile(outputBaseFileName);
colorFrame = imread(outputFullFileName);
[m n plane] = size(colorFrame);
colorFrame = double(colorFrame);
 C = zeros(m,n,plane);
 factor = max(colorFrame(:)) * 0.2;
 for i = 1:m
 for j = 1:n
 if (colorFrame(i,j,R) > factor && colorFrame(i,j,R) ==
max([colorFrame(i,j,1) colorFrame(i,j,2) colorFrame(i,j,3)]))
 C(i,j,1:3) = colorFrame(i,j,1:3);
 else
 C(i,j,1:3) = (colorFrame(i,j,1) * 0.3) + (colorFrame(i,j,2) * 0.59) +
(colorFrame(i,j,3) * 0.11);
 end
 end
 end
 C = uint8(C);
 cd C:\Modified_color\azul;
 imwrite(C, outputFullFileName, 'jpg');
 waitbar(frame/(NF2),wb);
 cd(get(handles.edit4,'String'));
end
delete(wb);
uiwait(msgbox('Segmentacion Terminada','Informacion','modal'));
set(handles.show_color,'Enable','on')
```

% Mostrar segmentación por color.

```
function show_color_Callback(hObject, eventdata, handles)
NF3=str2double(get(handles.edit3,'String'));
for frame = 1 : NF3
     if (get(handles.rcolor,'Value'))== get(handles.rcolor,'Max')
        cd(get(handles.edit8,'String'));
        outputBaseFileName = sprintf('Frame %4.4d.jpg', frame);
        outputFullFileName = fullfile(outputBaseFileName);
        thisFrame = imread(outputFullFileName);
        recalledMovie(frame) = im2frame(thisFrame);
     else
        if (get(handles.vcolor,'Value'))== get(handles.vcolor,'Max')
        cd(get(handles.edit10,'String'));
        outputBaseFileName = sprintf('Frame %4.4d.jpg', frame);
        outputFullFileName = fullfile(outputBaseFileName);
        thisFrame = imread(outputFullFileName);
        recalledMovie(frame) = im2frame(thisFrame);
```

```
        else
        cd(get(handles.edit11,'String'));
        outputBaseFileName = sprintf('Frame %4.4d.jpg', frame);
        outputFullFileName = fullfile(outputBaseFileName);
        thisFrame = imread(outputFullFileName);
        recalledMovie(frame) = im2frame(thisFrame);
        end
        end
        end
 movie(handles.axes2,recalledMovie);
 set(handles.Grabar_Color,'Enable','on')
```

% Equalizador de ajuste de contraste texturas.

```
function adjust_Callback(hObject, eventdata, handles)
Umbral=str2double(get(handles.edit14,'String'));
if (Umbral>=1) || (Umbral<=0)
 uiwait(msgbox('Valor fuera de rango; se usará el valor por defecto
0.8','Informacion','modal'));
 set(handles.edit14,'String',0.8);
 showinfowindow('De click en TEXTURA para segmentar el video
nuevamente ','Aviso');
pause(2)
close Aviso
pause(4)
else
cd(get(handles.edit4,'String'));
NF8=str2double(get(handles.edit3,'String'));
wb = waitbar(0,'Por favor espere...');
for frame = 1 : NF8
outputBaseFileName = sprintf('Frame %4.4d.jpg', frame);
outputFullFileName = fullfile(outputBaseFileName);
II= imread(outputFullFileName);
I=rgb2gray(II);
I=imadjust(I,[0.2 0.8],[0 1],1);
E = stdfilt(I);
Eim = mat2gray(E);
BW1 = im2bw(Eim,Umbral);
BWao = bwareaopen(BW1,1);
nhood = true(9);
closeBWao = imclose(BWao,nhood);
mask2 = imfill(closeBWao,'holes');
I2 = I;
I(mask2) = 0;
E2 = stdfilt(I2);
E2im = mat2gray(E2);
BW2 = im2bw(E2im,graythresh(E2im));
mask2 = bwareaopen(BW2,str2double(get(handles.edit15,'String')));
boundary = bwperim(mask2);
SR = I;
SR(boundary) = 255;
NewFrameRot = imrotate(SR,180);
BIN=fliplr(NewFrameRot);
grapmap=gray(256);
cd(get(handles.edit9,'String'));
imwrite(SR, outputFullFileName, 'jpg');
 waitbar(frame/(NF8),wb);
recalledMovie(frame) = im2frame(BIN,grapmap);
cd(get(handles.edit4,'String'));
end
end
delete(wb);
uiwait(msgbox('Segmentacion Terminada','Informacion','modal'));
```

```
movie(handles.axes4,recalledMovie)
cla(handles.axes4,'reset');
set(handles.axes4,'XTick',[],'YTick',[])
set(handles.Grabar_Textura,'Enable','on')
```

% Equalizador segmentación por color.

```
function equalizer_Callback(hObject, eventdata, handles)
cd(get(handles.edit4,'String'));
NF6=str2double(get(handles.edit3,'String'));
for frame = 1 : NF6
        outputBaseFileName = sprintf('Frame %4.4d.jpg', frame);
        outputFullFileName = fullfile(outputBaseFileName);
        x= imread(outputFullFileName);
Red=x(:,:,1);%separacion de la matriz RGB en RED;
Green=x(:,:,2);%separacion de la matriz RGB en GREEN;
Blue=x(:,:,3);%separacion de la matriz RGB en BLUE;
RedQ=histeq(Red,255);
GreenQ=histeq(Green,255);
BlueQ=histeq(Blue,255);
y=cat(3,RedQ,GreenQ,BlueQ);
imwrite(y, outputFullFileName, 'jpg');
recalledMovie(frame) = im2frame(y);
end
movie(handles.axes1,recalledMovie);
```

% Equalizador Binario por bordes.

```
function Binarizado_Callback(hObject, eventdata, handles)
cd(get(handles.edit4,'String'));
NF2=str2double(get(handles.edit3,'String'));
wb = waitbar(0,'Por favor espere...');
for frame = 1 : NF2
        outputBaseFileName = sprintf('Frame %4.4d.jpg', frame);
        outputFullFileName = fullfile(outputBaseFileName);
        outputBaseFileName1 = sprintf('Frame%3.3d.jpg', frame);
        outputFullFileName1 = fullfile(outputBaseFileName1);
        outputBaseFileName2 = sprintf('Frame%4.3d.jpg', frame);
        outputFullFileName2 = fullfile(outputBaseFileName2);
        thisFrame = imread(outputFullFileName);
        BW0=rgb2gray(thisFrame);
        BW3=im2bw(BW0,colormap(gray(256)),0.5);
        BW4 = uint8(BW3);
        grapmap=gray(2);
        BW5 = imrotate(BW4,180);
        BW6=fliplr(BW5);
        BW1=edge(BW4,'canny',0.2,0.8);
        BW2=~BW1;
        cd(get(handles.edit6,'String'));
        imwrite(BW4, grapmap, outputFullFileName2, 'jpg');
        imwrite(BW1, outputFullFileName, 'jpg');
        imwrite(BW2, outputFullFileName1, 'jpg');
        recalledMovie(frame) = im2frame(BW6,grapmap);
        waitbar(frame/(NF2),wb);
        cd(get(handles.edit4,'String'));
end
delete(wb);
uiwait(msgbox('Segmentacion Terminada','Informacion','modal'));
movie(handles.axes1,recalledMovie)
set(handles.show_canny,'Enable','on')
```

% Inicio de la segmentación por Texturas.

```
function Back_removal_Callback(hObject, eventdata, handles)
Umbral=str2double(get(handles.edit14,'String'));
if (Umbral>=1) || (Umbral<=0)
```

```
uiwait(msgbox('Valor fuera de rango; se usará el valor por defecto
0.8','Informacion','modal'));
set(handles.edit14,'String',0.8);
showinfowindow('De click en TEXTURA para segmentar el video nuevamente
','Aviso');
pause(2)
close Aviso
pause(4)
else
cd(get(handles.edit4,'String'));
NF8=str2double(get(handles.edit3,'String'));
wb = waitbar(0,'Por favor espere...');
for frame = 1 : NF8
outputBaseFileName = sprintf('Frame %4.4d.jpg', frame);
outputFullFileName = fullfile(outputBaseFileName);
II= imread(outputFullFileName);
I=rgb2gray(II);
E = stdfilt(I);
Eim = mat2gray(E);
BW1 = im2bw(Eim,Umbral);
BWao = bwareaopen(BW1,1);
nhood = true(9);
closeBWao = imclose(BWao,nhood);
mask2 = imfill(closeBWao,'holes');
I2 = I;
I(mask2) = 0;
E2 = stdfilt(I2);
E2im = mat2gray(E2);
BW2 = im2bw(E2im,graythresh(E2im));
mask2 = bwareaopen(BW2,str2double(get(handles.edit15,'String')));
boundary = bwperim(mask2);
SR = I;
SR(boundary) = 255;
NewFrameRot = imrotate(SR,180);
BIN=fliplr(NewFrameRot);
grapmap=gray(256);
cd(get(handles.edit9,'String'));
imwrite(SR, outputFullFileName, 'jpg');
 waitbar(frame/(NF8),wb);
recalledMovie(frame) = im2frame(BIN,grapmap);
cd(get(handles.edit4,'String'));
end
end
delete(wb);
uiwait(msgbox('Segmentacion Terminada','Informacion','modal'));
movie(handles.axes4,recalledMovie)
cla(handles.axes4,'reset');
set(handles.axes4,'XTick',[],'YTick',[])
set(handles.Grabar_Textura,'Enable','on')
```

% Botón de reseteo de la aplicación.

```
function pushbutton28_Callback(hObject, eventdata, handles)
cd(get(handles.edit7,'String'));
close('slector_de_webcam')
run .\slector_de_webcam.m
```

% Botón de reseteo de pantallas.

```
function pushbutton29_Callback(hObject, eventdata, handles)
cla(handles.axes1,'reset');
set(handles.axes1,'XTick',[],'YTick',[])
```

% Grabar la segmentación por bordes en AVI.

```
function Grabar_Canny_Callback(hObject, eventdata, handles)
```

```
cd(get(handles.edit6,'String'));
[nombre4,ruta4]=uiputfile('*.avi','Video');
   if ruta4==0, return, end
[nombre5,ruta5]=uiputfile('*.avi','Video');
   if ruta5==0, return, end
NF9=str2double(get(handles.edit3,'String'));
wb = waitbar(0,'Por favor espere...');
for frame = 1 : NF9
        outputBaseFileName = sprintf('Frame %4.4d.jpg', frame);
        outputFullFileName = fullfile(outputBaseFileName);
        outputBaseFileName1 = sprintf('Frame%3.3d.jpg', frame);
        outputFullFileName1 = fullfile(outputBaseFileName1);
        thisFrame = imread(outputFullFileName);
        thisFrame1 = imread(outputFullFileName1);
        grapmap = gray(256);
        recalledMovie(frame) = im2frame(thisFrame, grapmap);
        recalledMovie1(frame) = im2frame(thisFrame1, grapmap);
        waitbar(frame/(NF9),wb);
end
movie2avi(recalledMovie, fullfile(ruta4,nombre4), 'compression',
'None');
movie2avi(recalledMovie1, fullfile(ruta5,nombre5), 'compression',
'None');
delete(wb);
uiwait(msgbox('Grabacion Completa','Informacion','modal'));
```

% Grabar la segmentación por color en AVI.

```
function Grabar_Color_Callback(hObject, eventdata, handles)
cd(get(handles.edit8,'String'));
[nombre6,ruta6]=uiputfile('*.avi','Video');
   if ruta6==0, return, end
cd(get(handles.edit10,'String'));
[nombre7,ruta7]=uiputfile('*.avi','Video');
   if ruta7==0, return, end
cd(get(handles.edit11,'String'));
[nombre8,ruta8]=uiputfile('*.avi','Video');
   if ruta8==0, return, end
NF10=str2double(get(handles.edit3,'String'));
wb = waitbar(0,'Por favor espere...');
for frame = 1 : NF10
        cd(get(handles.edit8,'String'));
        outputBaseFileName3 = sprintf('Frame %4.4d.jpg', frame);
        outputFullFileName3 = fullfile(outputBaseFileName3);
        thisFrame3 = imread(outputFullFileName3);
        grapmap = gray(256);
        recalledMovie3(frame) = im2frame(thisFrame3, grapmap);
        cd(get(handles.edit10,'String'));
        outputBaseFileName1 = sprintf('Frame %4.4d.jpg', frame);
        outputFullFileName1 = fullfile(outputBaseFileName1);
        thisFrame1 = imread(outputFullFileName1);
        grapmap = gray(256);
        recalledMovie1(frame) = im2frame(thisFrame1, grapmap);
        cd(get(handles.edit11,'String'));
        outputBaseFileName2 = sprintf('Frame %4.4d.jpg', frame);
        outputFullFileName2 = fullfile(outputBaseFileName2);
        thisFrame2 = imread(outputFullFileName2);
        grapmap = gray(256);
        recalledMovie2(frame) = im2frame(thisFrame2, grapmap);
        waitbar(frame/(NF10),wb);
end
movie2avi(recalledMovie3, fullfile(ruta6,nombre6), 'compression',
'None');
```

```
movie2avi(recalledMovie1, fullfile(ruta7,nombre7), 'compression',
'None');
movie2avi(recalledMovie2, fullfile(ruta8,nombre8), 'compression',
'None');
delete(wb);
```

% Grabar la segmentación por textura en AVI.

```
function Grabar_Textura_Callback(hObject, eventdata, handles)
cd(get(handles.edit9,'String'));
[nombre9,ruta9]=uiputfile('*.avi','Video');
   if ruta9==0, return, end
NF11=str2double(get(handles.edit3,'String'));
wb = waitbar(0,'Por favor espere...');
for frame = 1 : NF11
        outputBaseFileName = sprintf('Frame %4.4d.jpg', frame);
        outputFullFileName = fullfile(outputBaseFileName);
        % Read the image in from disk.
        thisFrame = imread(outputFullFileName);
        grapmap = gray(256);
        recalledMovie(frame) = im2frame(thisFrame, grapmap);
        waitbar(frame/(NF11),wb);
end
movie2avi(recalledMovie, fullfile(ruta9,nombre9), 'compression',
'None');
delete(wb);
uiwait(msgbox('Grabacion Completa','Informacion','modal'));
```

%CALCULO DE LA REGION ROI Y MASCARAS BINARIAS

```
% --- Executes on button press in RegionColor.
function RegionColor_Callback(hObject, eventdata, handles)
%ROJO
cd(get(handles.edit8,'String'));
NF2=str2double(get(handles.edit3,'String'));
%wb = waitbar(0,'Por favor espere...');
for frame = 1 : NF2
outputBaseFileName = sprintf('Frame %4.4d.jpg', frame);
outputFullFileName = fullfile(outputBaseFileName);
outputBaseFileName1 = sprintf('Frame %3.3d.jpg', frame);
outputFullFileName1 = fullfile(outputBaseFileName1);
RGB = imread(outputFullFileName);
RGBR = roicolor(RGB(:,:,1),140,255) & roicolor(RGB(:,:,2),0,100) &
roicolor(RGB(:,:,3),0,100);
imwrite(RGBR,[1 1 1;1 0 0], outputFullFileName1, 'jpg');
end
%VERDE
cd(get(handles.edit10,'String'));
NF2=str2double(get(handles.edit3,'String'));
for frame = 1 : NF2
outputBaseFileName = sprintf('Frame %4.4d.jpg', frame);
outputFullFileName = fullfile(outputBaseFileName);
outputBaseFileName1 = sprintf('Frame %3.3d.jpg', frame);
outputFullFileName1 = fullfile(outputBaseFileName1);
RGB = imread(outputFullFileName);
RGBG = roicolor(RGB(:,:,1),0,160) & roicolor(RGB(:,:,2),100,255) &
roicolor(RGB(:,:,3),0,95);
imwrite(RGBG,[1 1 1;0 1 0], outputFullFileName1, 'jpg');
end
%AZUL
cd(get(handles.edit11,'String'));
NF2=str2double(get(handles.edit3,'String'));
for frame = 1 : NF2
```

```
% Construct an output image file name.
outputBaseFileName = sprintf('Frame %4.4d.jpg', frame);
outputFullFileName = fullfile(outputBaseFileName);
outputBaseFileName1 = sprintf('Frame %3.3d.jpg', frame);
outputFullFileName1 = fullfile(outputBaseFileName1);
% Read the image in from disk.
RGB = imread(outputFullFileName);
RGBB = roicolor(RGB(:,:,1),0,120) & roicolor(RGB(:,:,2),0,120) &
roicolor(RGB(:,:,3),120,255);
imwrite(RGBB,[1 1 1;0 0 1], outputFullFileName1, 'jpg');
end

% --- Executes on button press in RegionVideo.
function RegionVideo_Callback(hObject, eventdata, handles)
% hObject    handle to RegionVideo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
NF3=str2double(get(handles.edit3,'String'));
for frame = 1 : NF3
     if (get(handles.rcolor,'Value'))== get(handles.rcolor,'Max')
        cd(get(handles.edit8,'String'));
        % Construct an output image file name.
        outputBaseFileName = sprintf('Frame %3.3d.jpg', frame);
        outputFullFileName = fullfile(outputBaseFileName);
        xFrame = imread(outputFullFileName);
        %thisFrame = imrotate(thisFrame,180);
        %BI = fliplr(imrotate(xFrame,180));
        grapmap = gray(256);
        recalledMovie(frame) = im2frame(xFrame, grapmap);
     else
        if (get(handles.vcolor,'Value'))== get(handles.vcolor,'Max')
        cd(get(handles.edit10,'String'));
        % Construct an output image file name.
        outputBaseFileName = sprintf('Frame %3.3d.jpg', frame);
        outputFullFileName = fullfile(outputBaseFileName);
        thisFrame = imread(outputFullFileName);
        %thisFrame = imrotate(thisFrame,180);
        %BI=fliplr(thisFrame);
        grapmap = gray(256);
        recalledMovie(frame) = im2frame(thisFrame, grapmap);
        else
        cd(get(handles.edit11,'String'));
        % Construct an output image file name.
        outputBaseFileName = sprintf('Frame %3.3d.jpg', frame);
        outputFullFileName = fullfile(outputBaseFileName);
        thisFrame = imread(outputFullFileName);
        %thisFrame = imrotate(thisFrame,180);
        %BI=fliplr(thisFrame);
        grapmap = gray(256);
        recalledMovie(frame) = im2frame(thisFrame, grapmap);
        end
        end
        end
 movie(handles.axes2,recalledMovie);
 set(handles.Grabar_Color,'Enable','on')
 guidata(hObject, handles);


% --- Executes on button press in pushbutton35.
function pushbutton35_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton35 (see GCBO)
```

```
cd(get(handles.edit4,'String'));
NF2=str2double(get(handles.edit3,'String'));
outputBaseFileName = sprintf('Frame %4.4d.jpg', 1);
outputFullFileName = fullfile(outputBaseFileName);
firstFrame = imread(outputFullFileName);
imshow(firstFrame);
hFH = imfreehand();
binaryImage = hFH.createMask();
imshow(binaryImage);
wb = waitbar(0,'Por favor espere...');
for frame = 1 : NF2
        outputBaseFileName = sprintf('Frame %4.4d.jpg', frame);
        outputFullFileName = fullfile(outputBaseFileName);
        outputBaseFileName1 = sprintf('Frame%3.3d.jpg', frame);
        outputFullFileName1 = fullfile(outputBaseFileName1);
        thisFrame = imread(outputFullFileName);
        BW=rgb2gray(thisFrame);
        BW1=edge(BW,'canny');
        BW2=and(BW1,binaryImage);
        cd(get(handles.edit6,'String'));
        imwrite(BW2, outputFullFileName1, 'jpg');
        waitbar(frame/(NF2),wb);
        cd(get(handles.edit4,'String'));
end
delete(wb);
uiwait(msgbox('Segmentacion Terminada','Informacion','modal'));
cd(get(handles.edit6,'String'));
NF2=str2double(get(handles.edit3,'String'));
for frame = 1 : NF2
        outputBaseFileName1 = sprintf('Frame%3.3d.jpg', frame);
        outputFullFileName1 = fullfile(outputBaseFileName1);
        NewFrame1 = imread(outputFullFileName1);
        NewFrameRotFinal1 = imrotate(NewFrame1,180);
        BI=fliplr(NewFrameRotFinal1);
        grapmap = gray(256);
        recalledMovie(frame) = im2frame(BI, grapmap);
    end
hold on
movie(handles.axes3, recalledMovie);
set(handles.Grabar_Canny,'Enable','on')
guidata(hObject, handles);
```

ANEXO C

CUESTIONARIO DE PREGUNTAS PARA LA ENCUESTA DE PERCEPCIÓN VISUAL.

1. ¿Qué método de segmentación le pareció de mayor utilidad?
   a. Color
   a. Borde
   b. Textura

2. Según el método que haya escogido en el literal anterior, mencione la utilidad más importante que usted le daría al método.
   a. Entretenimiento
   b. Comercial
   c. Científico
   d. Profesional
   e. Otro especifique

3. ¿Le parece sencillo el uso de la herramienta?
   a. Si
   b. No

4. ¿Qué cambios le haría usted para mejorarla?
   a. Cambiaría el diseño de la interfaz
   b. Está bien así como está
   c. Otro especifique

5. ¿Cree usted que la calidad del video resultante se degrada (pixelea) con respecto a la del video original?

6. ¿Cuál de los ecualizadores según usted le agrega mayor valor a la segmentación?
   a. Ecualizador de color
   b. Ecualizador Binario
   c. Ecualizador de Textura

7. ¿Considera que el procesamiento de extracción de los cuadros del video toma mucho tiempo?
8. ¿Considera útil la segmentación en un video o prefiere hacerla mejor sobre una imagen?

BIBLIOGRAFÍA

[1] Báez, J.J. , Segmentación de imágenes de color, http://www.ejournal.unam.mx/rmf/no506/RMF50605.pdf , fecha de consulta enero de 2011.

[2] Banterla, Flavio , Análisis de Texturas en Matlab, http://www.ehu.es/ccwintco/uploads/d/d7/Texturas.pdf , fecha de consulta enero de 2011.

[3] Valverde, Jorge , Detección de Bordes mediante el algoritmo de Canny, http://www.seccperu.org/files/Deteccionde Bordes-Canny.pdf, fecha de consulta enero de 2011

[4] Arfan, Jaffar, color video segmentation, http://www.wseas.us/e-library/conferences/2009/istanbul/SIP-WAV/SIP-WAV-03.pdf, fecha de consulta febrero 2011

[5] Zaldivar, Daniel, Procesamiento Digital de imágenes, http://es.scribd.com/doc/61579860/23371-Procesamiento-de-Imagenes-Con-Matlab, fecha de consulta Marzo de 2011.