UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

UNIVERSIDAD
POLITECNICA
DE VALENCIA

ESCUELA POLITECNICA
SUPERIOR DE GANDIA

“Procesado digital de imágenes de video para la detección de humo ”

TRABAJO FINAL DE CARRERA

Autor/es:

Rubén Llobregat Rubio

Director/es:
Ignacio Bosch Roig

GANDIA, 2011

Índice General

Introducción

El presente Proyecto Final de Carrera (PFC) se ha realizado dentro del Grupo de Tratamiento de Señal (GTS) del Departamento de Comunicaciones de la Universidad Politécnica de Valencia, y tiene como finalidad la detección de humo mediante video.
El GTS forma parte de uno de los grupos de investigación del Instituto de Telecomunicaciones y Aplicaciones Multimedia(iTEAM). El iTEAM es un centro de investigación integrado en la Ciudad Politécnica de la Innovación, el nuevo parque científico de la Universidad Politécnica de Valencia(UPV), donde se desarrollan actividades de Investigación, Desarrollo e innovación (I+D+i) dentro del área de las Tecnologías de la Información y las Comunicaciones. El instituto está formado por 9 grupos de investigación reconocidos por la UPV, entre los cuáles se encuentra el Grupo de Tratamiento de Señal. Éste se dedica al tratamiento de señales bi o multidimensionales y particularmente de imágenes. Entre sus áreas de trabajo en I+D+i cabe destacar:

- Tratamiento de señales ultrasónicas
- Tratamiento de señales láser-ultrasonidos para control de calidad sin contacto
- Tratamiento de señales infrarrojas
- Análisis digital para la detección, caracterización y clasificación de objetos en imágenes
- Restauración de imágenes y películas antiguas
- Sistemas de control de calidad mendiante visión artificial
- Tratamiento digital de información multimedia
- Recuperación e indexación de información multimedia
- Técnicas de compresión de vídeo
- Marcas de agua de señales (watermarks) para la autenticación e identificación de señales

Objetivos

El principal objetivo de este trabajo es conseguir detectar humo de ficheros de vídeo de incendios para ello obtendremos al final del proceso una imagen en la que se indicara las zonas de humo reconocidas .

Para conseguir el funcionamiento deseado se han planteado los siguientes objetivos:

- Detectar los distintos comportamientos del humo para aislarlo del resto de la imagen del vídeo.
- Detectar los distintos factores que pueden confundirse con humo para evitar errores.
- Medida y comparativa de distintos parámetros relacionada con las imágenes con humo.
- Compresión de las distintas herramientas de Matlab.

El procedimiento que se ha seguido para conseguir estos objetivos se resume en 4 capítulos.

Metodología

Durante la realización del PFC se han desarrollado cada uno de los puntos que se tenía como objetivo y se ha recopilado el resultado en la presente memoria. La memoria se compone de 4 capítulos que describen el trabajo realizado como sigue:

Capítulo 1: El primero de ellos contiene una explicación teórica sobre los distintos métodos y técnicas que se han tenido en cuenta para la realización de los algoritmos de detección de cambios.

Capítulo 2: En el segundo se explica todo el procedimiento que se lleva a cabo a partir de las imágenes originales hasta obtener el resultado final con las detecciones de posibles nuevas construcciones.

Capítulo 3: En el tercer capítulo se explica paso a paso el procedimiento que se ha seguido para que la aplicación finalmente funcione como se ha descrito en el capítulo 2.

Capítulo 4: En el último capítulo se explican las opciones que dispone cada una de las interfaces que han sido diseñadas para usar la aplicación.

Al final de la memoria se recopilan las conclusiones a las que finalmente se ha llegado tras todo el proceso realizado, así como posibles ampliaciones para un trabajo futuro.y las referencias bibliográficas.

Capítulo 1

Técnicas de tratamiento digital de imágenes

1.1. Introducción

Detectar regiones de cambios en distintas imágenes de una misma escena tomadas en distintos instantes de tiempo y en distintas condiciones meteorológicas resulta muy interesante para diversas aplicaciones en distintas disciplinas. Importantes aplicaciones de detecciones de cambios son la video vigilancia, tratamientos y diagnósticos médicos , ... Partimos de imágenes de una misma escena tomada en diferentes instantes de tiempo. El objetivo es identificar los píxeles significativamente diferentes entre la última imagen tomada y la anterior. Estos píxeles comprenden la máscara de cambios. La máscara de cambios es la combinación de distintos factores, que incluye la verosimilitud y la no verosimilitud entre imágenes, el movimiento de los objetos, o el cambio en la forma de los mismos. Además los objetos pueden tener cambios en cuanto al brillo y al color de la imagen. Estimar los cambios en la máscara es uno de los primeros pasos. Clasificar estos cambios requiere herramientas específicas según la aplicación a considerar. Este es el principal problema de la detección, que se tiene que adaptar según la situación en concreto.

1.2. Planteamiento del problema

Se va a precisar el problema de la detección de cambios. Se parte de una secuencia de imágenes $\{I_1, I_2, \ldots, I_M\}$ y la coordenada de un píxel $x \in M.^1$, donde $I(x) \in \mathbf{R}^k$, k subimágenes. Normalmente se trabaja en una sola dimensión, $k = 1$ (en el caso de imágenes en blanco y negro) o en tres dimensiones, $k = 3$ (imágenes a color RGB), pero hay otros posibles valores. Un algoritmo básico de detección de cambios toma como entrada una imagen y genera una imagen binaria $B : \mathbf{R}^l \rightarrow [0,1]$ llamada máscara de cambios que identifica las regiones de cambios en la última imagen según:

$$B(x) = \begin{cases} 1 & \text{si hay un cambio significante en el píxel x de la imagen } I_M, \\ 0 & \text{de lo contrario.} \end{cases} \quad (1.1$$

Figura 1.1: Distintos tipos de cambios detectados en una imagen actual a partir de una imagen antigua

La Figura 1.1 es un simple ejemplo que incluye cambios debidos a la luminosidad, así como de movimiento etiquetados como “M”, reflejos (“S”), y objetos que cambian de aspecto con el tiempo (“A”). Este ejemplo ilustra la complejidad de la detección de cambios en una imagen real. La intensidad que ha cambiado para cada píxel es debido a distintos factores. La cámara se ha movido ligeramente, la fuente de luz también ha cambiado de intensidad y posición, y algunos objetos se han movido independientemente de la cámara (ej: el reloj).

A continuación se van a describir algunas de las técnicas que se han utilizado para la realización del proyecto fin de carrera para proceder a la detección de nuevos objetos en imágenes actuales. Algunas de ellas son:

Métodos de pre-procesado: técnicas que se utilizan para ajustar las imágenes antigua y actual para que puedan ser comparadas. En este apartado se hablará sobre la georreferenciación de las imágenes, la intensidad de los píxeles, el filtrado, la iluminación y el suavizado.

Diferenciación: técnica utilizada para determinar los píxeles que contienen variaciones.

Histograma: representación de los niveles de intensidad de los píxeles utilizada para determinar los que contienen mayor grado de diferencia.

Segmentación: método utilizado para separar los píxeles que interesan del fondo de la imagen. Para ello existen 3 técnicas que se explicarán a continuación.

Procesado morfológico: técnica que se utiliza para agrupar los píxeles correspondientes a los cambios detectados para conseguir la máscara de detecciones final.

Representación del color: técnicas de representación del color en RGB y HSV utilizadas para la detección de piscinas.

1.3. Métodos de pre-procesado

El objetivo del algoritmo de la detección de cambios es detectar los cambios significativos pudiendo descartar los cambios que no son importantes o no nos interesan. Para poder hacer esta distinción se requieren métodos sofisticados que puedan modelar todos los tipos de cambios (los importantes y los menos aplicación. Los siguientes apartado describen algunos pasos del pre-procesado que se usa para eliminar o filtrar los cambios que se consideran como no importantes.

1.3.1. Ajustes geométricos

Debido a un ligero movimiento de la cámara en una de las imágenes a comparar, la intensidad de un píxel podría cambiar y por tanto se detectaría un cambio que no interesa. Por ello es importante georreferenciar las imágenes, fijarlas en un mismo sistema de coordenadas para que realmente se pueda comparar el mismo píxel en ambas imágenes. Elegir una buena transformación espacial es crítico para conseguir una buena detección.

Georreferenciación de imágenes

Georreferenciar es poner una imagen u objetos gráficos dibujados en un plano, en conformidad geométrica con la realidad, en un sistema de proyección dada. La georreferenciación es esencial para asegurar la validez de la localización de los objetos en una base de datos espaciales. Se pueden georreferenciar:

imágenes (escaneadas o procedentes de un captor)

objetos vectoriales (archivos mal georreferenciados o creados por un programa informático de diseño)

Las imágenes son conformadas por un conjunto de píxeles que tienen la misma forma. La posición absoluta de un píxel es conocida por su posición relativa en el conjunto. Solo basta con conocer la posición de un solo píxel, la forma y la dimensión de los píxeles, el número del que se dispone por columna y fila, para conocer la posición absoluta de cualquier píxel.

La georreferenciación deforma los píxeles de origen, por lo que es necesario realizar luego una operación de re-muestreo para volver a una representación bajo la forma de una imagen sin discontinuidades. El proceso de georreferenciación se puede realizar en tres etapas:

La primera etapa de la georreferenciación de una imagen consiste en definir cuál será la geometría de los píxeles en la imagen rectificada: tamaño, forma, coordenadas (plano de proyección).

La segunda etapa consiste en definir los parámetros que permiten calcular los píxeles de salida y asignarles un valor derivado del valor de los píxeles de la imagen de origen.

La tercera etapa consiste en gestionar de manera eficaz los píxeles conformando la relación georeferenciada. Para ello se usan técnicas de mosaiqueado e indexación.

Las causas de deformaciones de las imágenes son numerosas. Para las fotografías aéreas o las imágenes satelitales, se pueden resaltar las deformaciones derivadas de:

- la óptica de toma de vista
- las condiciones atmosféricas
- la posición del instrumento de toma de vista
- del relieve de la superficie terrestre

y para los documentos escaneados las deformaciones derivan de:

■la calidad del escaneo (deformaciones en los bordes)

■la deformación de la hoja

■la proyección en una superficie plana (en el caso de que sea desconocida)

Corregir geométricamente una imagen implica determinar un modelo de deformación entre las coordenadas en la imagen de origen y las coordenadas en el sistema de referencia utilizado para la relación a crear. Existen tres casos:

■se conocen los modelos de deformación con una gran precisión. En este caso solo basta con modelizar la rectificación en base a las deformaciones conocidas.

■no se conocen los modelos de deformación. Se modelizan entonces arbitrariamente estas deformaciones con una función dada, en general polinomial. Se calculan los coeficientes del modelo en base a puntos de apoyo (o puntos de control), cuyas coordenadas son conocidas de los dos lados (en la imagen y en el sistema de referencia de la relación a crear).

■se conocen los modelos de deformación, pero con una precisión insuficiente. Se ajusta entonces el modelo en base a los puntos de control. El número de puntos de control necesarios depende, entonces, del modelo.

Para finalmente representar la imagen georeferenciada sin discontinuidades sufridas tras el proceso, la imagen pasará por una etapa de re-muestreo. En esta etapa se calcula el valor para cada píxel de salida en función de los valores que se tenían para la imagen origen.

1.3.2. Ajustes de intensidad

En distintos campos de detección de cambios, las variaciones de intensidad causadas por cambios de posición de las fuentes de luz no son consideradas como importantes. Sin embargo, en el caso estudiado sí que lo es. En esta sección se describen algunas técnicas para compensar las variaciones de intensidad entre imágenes.

Normalización de la intensidad

Fué uno de los primeros pasos que se llevó a cabo para ajustar el cambio de la iluminación en la detección de cambios [3], [4]. La intensidad de un píxel en una imagen se normaliza para obtener la misma media y varianza en ambas imágenes a comparar.

$$\tilde{I}_2(x) = \frac{\sigma_1}{\sigma_2}\{I_2(x) - \mu_2\} + \mu_1 \qquad (1.2)$$

$\tilde{I}_2$ representa la imagen actual normalizada y μ_i, σ_i la media y la desviación estándar de la intensidad I_i, respectivamente. Asímismo, ambas imágenes pueden ser normalizadas para una media nula y varianza unidad. Esto permite utilizar umbrales de decisión independientes de las intensidades de ambas imágenes.

En vez de aplicar la normalización (1.2) para cada píxel utilizando estadísticas globales μi, σi las imágenes se pueden dividir en bloques y hacer la normalización para cada bloque independientemente.

Filtrado

Para algunas imágenes la intensidad del píxel x se puede modelar como el producto de dos componentes: la iluminación *Ii(x)* en la escena y el reflejo $I_r(x)$ en la superficie del objeto. Por tanto:

$$I(x) = Ii(x)I_r(x) \tag{1.3}$$

Solamente la componente del reflejo $I_r(x)$ contiene información sobre los objetos de la escena. Si la iluminación *Ii(x)* tiene menor contenido espacial en frecuencia que la componente $I_r(x)$, el filtro puede usarse para separar las dos componentes de la señal intensidad. Esto es, tomar el logaritmo neperiano en ambas partes de la ecuación (1.3) quedando:

$$lnI(x) = lnIi(x) + lnI_r(x) \tag{1.4}$$

Ahora que la componente de baja frecuencia *lnIi(x)* es aditiva, se puede usar el filtro paso alto para eliminarla y quedarse solamente con la componente que aporta la información. De esta manera la componente $I_r(x)$ se puede estimar como

$$I_r(x) = exp\{F(lnI(x))\} \tag{1.5}$$

donde *F(-)* es un filtro paso alto.

Iluminación

Modelar y compensar las variaciones locales de intensidad es necesario en diversas aplicaciones. Por ejemplo, Can y Singh modelaron la componente iluminación como una función polinomial. Negahdaripour propuso un modelo genérico lineal para la variación de la iluminación entre imágenes de una misma escena:

$$I_2(x, y) = M(x, y)I_1(x, y) + A(x, y) \tag{1.6}$$

donde M(x,y) y A(x,y) son funciones con discontinuidades cerca de los bordes de las regiones, y *I* y *I2* corresponden a la intensidad de la imagen antigua y actual respectivamente. Hager y Belhumeur usaron “principal component analysis” (PCA) para extraer una serie de imágenes $B_k(x,y)$ que representan todos los puntos de vista de una escena bajo todas las posibles combinaciones de iluminación.

$$I_2(x, y) = I_1(x, y) + \sum_{k=1}^{k} \alpha k B k(x, y) \tag{1.7}$$

Estos sofisticados modelos de compensación de iluminación no son comunmente usados en la detección de cambios. Sin embargo, Bromiley propuso el cálculo de

los histogramas de las imágenes. Estos podían ser usados para estimar y eliminar modelos no paramétricos de iluminación de una imagen.

1.3.3. Suavizado

Otra técnica que se utiliza durante el pre-procesado es el suavizado de la imagen. Se le denomina “suavizado” o filtro de ruidos, a la operación de filtrado que se usa para eliminar el ruido de una imagen. Se trata, en cuestión, de un filtro paso-bajo. Estos consisten en atenuar las componentes de la imagen con alta frecuencia, dejando pasar la baja frecuencia. Este tipo de filtro generalmente se usa para atenuar el ruido de una imagen y para provocar el efecto para que la imagen aparezca algo borrosa (difumina), por eso usualmente se le denomina suavizado. Esto es útil en el pre-procesado para eliminar pequeños detalles antes de la extracción de un objeto (grande) y el relleno de pequeños espacios entre líneas o curvas.
La operación de suavizado específica que consiste en evitar el efecto escalonado que producen los píxeles en el borde de una figura geométrica se denomina antiescalonamiento.

1.4. Diferenciación

Algunos métodos de detección de cambios se basan en la imagen diferencia $D(x) = I_1(x) — I_2(x)$. El algorítmo utilizado se basa en la máscara de la imagen de cambios B(x), que viene definida según:

$$B(x) = \begin{cases} 1 & \text{si } |\mathbf{D}(x)| > \tau \\ 0 & \text{de lo contrario,} \end{cases} \tag{1.8}$$

Este algoritmo se denomina “simple diferenciación” donde el umbral ***r*** se elige empíricamente. Han habido diversas opiniones sobre cómo elegir dicho umbral adecuadamente para una aplicación específica. Con ello se pretende disminuir las falsas alarmas y las pérdidas de detecciones. Hay otros métodos que están relacionados con el de “simple diferenciación”. Por ejemplo, el *change vector analysis* (CVA), a menudo utilizado en imágenes, se genera un vector para cada píxel de la imagen, considerando todos los canales espectrales. El módulo de la diferencia entre dos vectores de un mismo píxel en imágenes distintas proporciona la imagen diferencia (en muchos casos, la dirección de este vector puede usarse para determinar diferentes tipos de cambios). *Image ratioing* es otra técnica, pero en este caso el umbral es elegido de manera global y es poco probable que supere en prestaciones a los algorítmos más avanzados. Esta técnica es sensible al ruido y a las variaciones de iluminación, y estos casos no han sido considerados al tomar la máscara de cambios.

1.5. Histograma de una imagen

Para determinar los píxeles que contienen las variaciones en las imágenes se han utilizado los histogramas. El histograma de una imagen es el ploteo de los valores de sus píxeles. Contiene el número de píxeles que tienen el mismo nivel de gris, es decir representa la probabilidad de que un determinado nivel de gris aparezca en la imagen.

En general se representa como un gráfico de barras en el que las abscisas son los distintos colores de la imagen y las ordenadas la frecuencia relativa con la que cada color aparece en la imagen. El histograma proporciona información sobre el brillo y el contraste de la imagen, y puede ser utilizado para ajustar estos parámetros, eliminar ciertas tonalidades molestas, etc..

Una imagen en blanco tendrá todos sus valores iguales a 255, y si la mitad es negra, en la gráfica del histograma aparecerán dos líneas iguales a ambos extremos: en los valores correspondientes al 0 y al 255. Una imagen de escala de grises tendrá en su histograma “x” píxeles con el valor 0, y “y” píxeles con el valor 1. Así, el histograma es la representación de la densidad de probabilidad de cada valor de gris para esa imagen.

En las Figuras 1.2, 1.3 y 1.4, se muestran algunos ejemplos de histogramas para distintos tipos de imágenes.

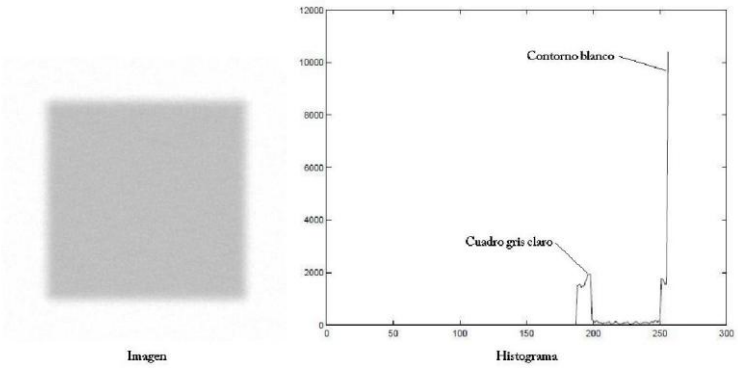


Figura 1.3: Ejemplo de histograma para una imagen gris claro con un contorno blanco

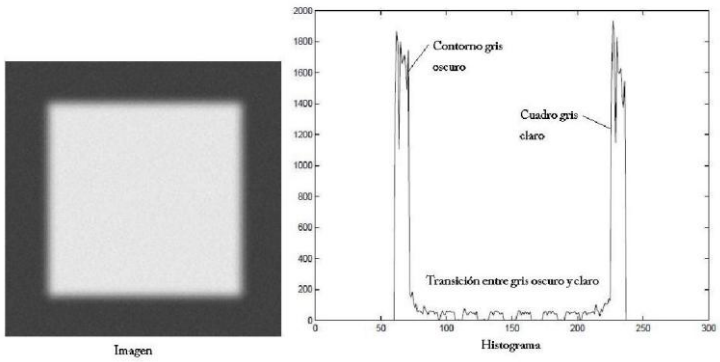


Figura 1.4: Ejemplo de histograma para una imagen con distintas tonalidades de gris

1.6. Segmentación

Uno de los pasos para analizar una imagen consiste en separar los distintos objetos, que aparecen en la misma, del fondo de la imagen. Existe una gran variedad de técnicas de segmentación, que se adaptan al tipo de imágenes y al objetivo que se persigue. Básicamente, podemos distinguir:Umbralización: los píxeles se clasifican atendiendo únicamente a su nivel de gris.

- Detección de regiones: los píxeles con características similares se agrupan en regiones, siendo cada región un objeto diferente.

- Detección de fronteras: se realiza una búsqueda de los píxeles que corresponden a fronteras que separan dos objetos diferentes. Una vez detectados, la segmentación es sencilla.

1.6.1. Segmentación mediante umbrales

Esta técnica se basa en determinar ciertos umbrales de forma que delimiten la imagen en zonas
(en lo que a valores de intensidad se refiere). Posteriormente, dependiendo del nivel de gris de
cada píxel, éste se clasificará en uno u otro objeto. Una herramienta muy útil para las técnicas
de umbralización es el estudio del histograma. De esta forma, si ***r*** representa el valor umbral
considerado y f(x,y) el nivel de gris para el punto (x,y), la imagen resultante tras la segmentación
se define como:

$$g(x,y) = \begin{cases} 1 & \text{si } f(x,y) > \tau \\ 0 & \text{delo contrario,} \end{cases} \qquad (1.)$$

En la Figura 1.5 se muestra un ejemplo de segmentación mediante umbralización. Se representa la imagen original, el correspondiente histograma con el valor umbral seleccionado, y la imagen resultante. La determinación del umbral puede hacerse de forma manual, tras la visualización de la imagen y su histograma, o de forma automática. Existen numerosas formas de calcular automáticamente los umbrales, las cuales emplean para ello el correspondiente histograma

1.6.2. Segmentación mediante detección de regiones

A diferencia de la técnica anterior, en donde cada píxel atendía únicamente a su nivel de gris, y no a su posición en la imagen o los valores de sus vecinos, en la segmentación por detección de regiones se tienen también en cuenta las propiedades espaciales de la imagen. Como región se entiende un conjunto de píxeles conectados entre sí. La conectividad puede ser de dos tipos fundamentalmente: 4 conectividad y 8 conectividad.

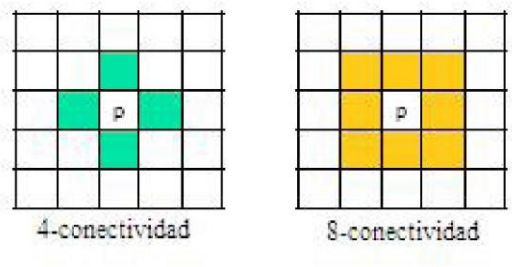


Figura 1.6: Tipos de conectividad

En la Figura 1.6 se puede observar este concepto: dado un píxel, el conjunto de píxeles que están conectados a él (coloreados en la figura) dependerá del tipo de conectividad. Comentaremos brevemente algunas técnicas:

Crecimiento de regiones: consiste en partir de un primer punto inicial (elegido manual o automáticamente), e ir agregándole todos los píxeles vecinos que cumplan una determinada condición, como puede ser que tengan un nivel de gris similar. Así, la región va creciendo a partir del píxel inicial hasta llegar a contener el máximo número de puntos conectados sin perder homogeneidad.

División y unión (Split and Merge): se parte de una imagen total y se divide en un número predeterminado de subregiones. Cada una de las subregiones puede ser ya homogénea (en función de algún criterio de homogeneidad) o no, en cuyo caso se sigue dividiendo en sub-regiones, y así sucesivamente. Al mismo tiempo, tras cada subdivisión, se comprueba si cualquiera de las regiones puede ser unida a una de sus regiones adyacentes para dar lugar a una región mayor que cumpla el criterio de homogeneidad. El proceso se aplica recur-sivamente hasta que todas las regiones finales cumplen la condición de homogeneidad. En general, la división en subregiones es siempre una división en cuadrantes, de forma que cualquier región no homogénea se descompondrá en cuatro cuadrantes iguales.

Texturas: Si nos basamos en conceptos de texturas, no siempre va a ocurrir, como hasta ahora se asumía, que píxeles que pertenecen a objetos diferentes presentan niveles de gris más o menos diferentes. Como textura de una imagen digital nos referimos a la disposición espacial de los niveles de gris de los píxeles en una determinada región. Entre los parámetros que permiten caracterizar la textura y estudiar su variación a lo largo de una imagen podemos encontrar: estadísticos como la media (información sobre claridad y oscuridad) o la varianza y desviación típica (información sobre contraste); la matriz de concurrencia, con la cual calcular parámetros como la entropía (información de ordenación de la imagen) o la energía (información de homogeneidad)

1.6.3. Segmentación mediante detección de contornos

Estas técnicas se basan en buscar discontinuidades (brusco cambio del nivel de gris) en la imagen, es decir, píxeles correspondientes a fronteras entre regiones, para, una vez localizados éstos, identificar las diferentes regiones separadas por dichas fronteras. Para ellos se emplean las derivadas primera (gradiente) y segunda

(laplaciana). Buscar puntos de contorno equivaldrá a la búsqueda de puntos de elevado gradiente o de cruces por cero de la laplaciana.

1.7. Procesado morfológico

Al hablar de morfología se hace referencia al estudio de la forma de los objetos. Más exactamente, en el análisis de las imágenes, nos referimos a las técnicas que nos permiten modificar la forma de los objetos y extraer de ésta aquellas características que nos permitan identificarlos. Con esta finalidad, existen una serie de herramientas de uso extendido en procesamiento de imagen, englobadas en lo que se denomina morfología matemática. Antes de la segmentación se pueden emplear técnicas de morfología de grises, y una vez segmentada la imagen es más sencillo aplicar morfología binaria. En este apartado se comentan brevemente algunas de las operaciones de morfología más interesantes, basándonos en imágenes binarias.
Previamente a comentar algunas de las operaciones más interesantes, se va hacer un pequeño h
formulación

1.7.1. Fu

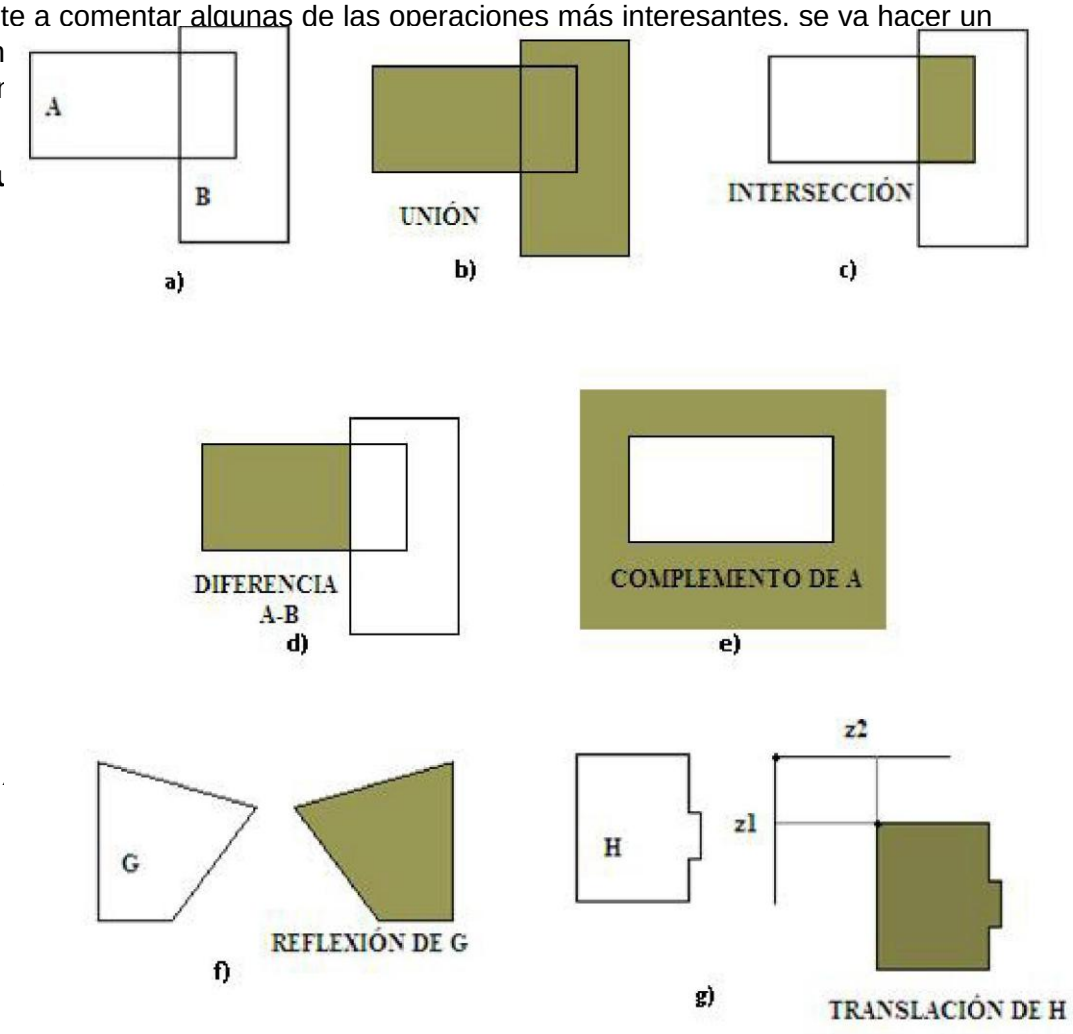


Figura 1.7: Ejemplos de operaciones básicas de conjuntos. En a) conjuntos A y B, en b) la unión de A y B, en c) la intersección de A y B, en d) la diferencia, en e) el complemento de A, en f) la reflexión, y en g) la translación

1.7.2. Principales operaciones morfológicas

Previamente a listar las mencionadas operaciones morfológicas, es de interés realizar un inciso para definir el término “elemento estructurante”, pues aparecerá en numerosas operaciones. Cuando se lleva a cabo una transformación morfológica, el objetivo es extraer, de las imágenes o conjuntos sobre los que se opera, ciertas estructuras geométricas mediante la utilización de otra imagen o conjunto de forma

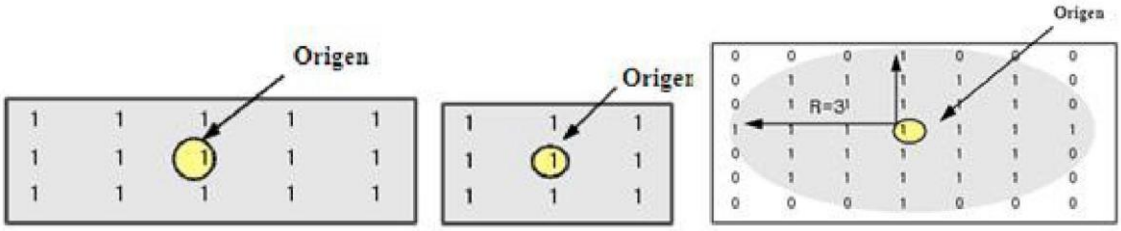


Figura 1.8: Ejemplos de elementos estructurantes

Las operaciones básicas desde el punto de vista del procesamiento morfológico de imágenes son la erosión y la dilatación. Su utilidad es importante no sólo como operadores aislados, sino como base para crear transformaciones más complejas.

■ **Erosión**

La erosión de un conjunto A por un elemento estructurante C se define como el conjunto de puntos o elementos z, pertenecientes a A, de forma que cuando el elemento estructurante C se traslada a ese punto, el elemento queda incluido en A.

$$A - C = \{z|(C)_z \subseteq A\} \tag{1.10}$$

El efecto que se consigue al erosionar un objeto es estrecharlo. Se disminuye el tamaño del objeto y también se eliminan objetos pequeños y salientes estrechos. Así mismo, este operador agranda agujeros y separa partes del objeto unidas por líneas finas. Se puede observar un ejemplo en la Figura 1.9.

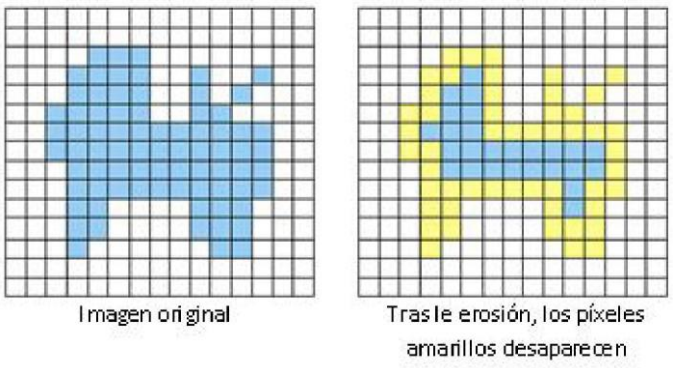


Figura 1.9: Erosión de un objeto, con un EE de tamaño 3x3

■ **Dilatación**

La dilatación de un conjunto A por un elemento estructurante C se define como el conjunto de puntos o elementos z, pertenecientes a A, de forma que cuando la

reflexión del elemento estructurante C se traslada a ese punto, el elemento y A tienen al menos un punto en común.

$$A + C = \{z|(\hat{C})_z \cap A \neq 0\} \quad (1.11)$$

El efecto conseguido al dilatar un objeto es un agrandamiento. Se aumenta su tamaño y se disminuyen o se eliminan los posibles agujeros (píxeles a '0'). También sirve para unir partes del objeto u objetos distintos que estuvieran separados por distancias pequeñas. A continuación se muestra un ejemplo de dilatación de un objeto empleando un elemento estructurante de tamaño 3x3.

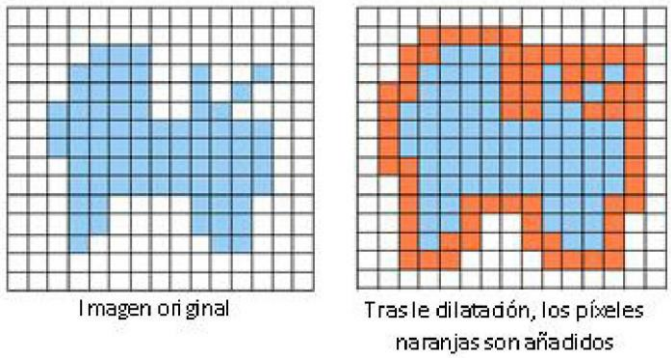


Figura 1.10: Dilatación de un objeto con un EE de tamaño 3x3

Así pues, estas técnicas se emplean mucho para eliminar pequeños objetos debidos al ruido de fondo (erosión), y para rellenar pequeños huecos aparecidos tras la umbralización (dilatación). El inconveniente que tienen es que modifican el tamaño de los objetos.

■ **Apertura y cierre**

Estas operaciones mantienen las ventajas de la erosión y la dilatación pero evitan su problema, es decir, la forma de los objetos no se altera.

Apertura = erosión + dilatación

$$A \mathbf{o} C = (A \ominus C) \oplus C \quad (1.12)$$

Cierre = dilatación + erosión

$$A \bullet C = (A \oplus C) \ominus C \quad (1.13)$$

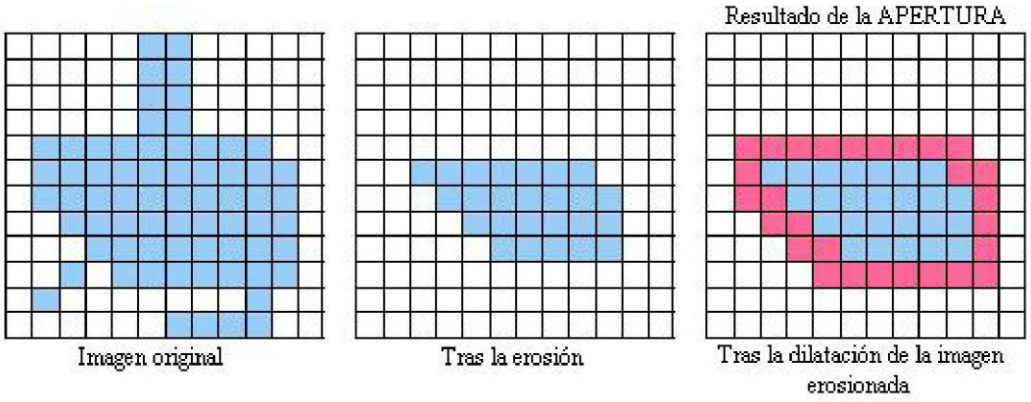


Figura 1.11: Apertura de un objeto. EE de tamaño 3x3

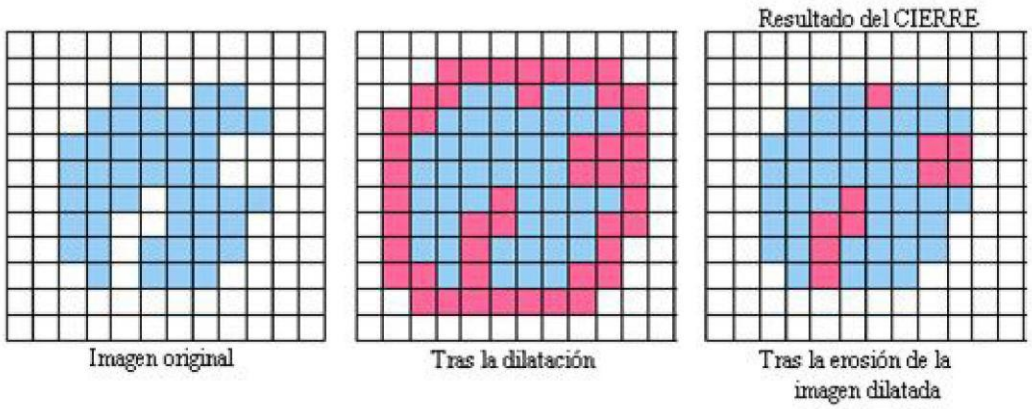


Figura 1.12: Cierre de un objeto. EE de tamaño 3x3

■ **Transformaciones "todo o nada" ("hit or miss")** Se basan en buscar un determinado patrón o máscara, elegida según la aplicación, por los píxeles de la imagen. Esta búsqueda de 'blancos y negros' es útil para detectar en la imagen formas especiales, tales como líneas, esquinas, etc., o incluso objetos de forma conocida que deseamos localizar.

$$A \mathbf{0} C = (A \mathbf{0} C\backslash) \mathbf{n} (A \quad \mathbf{0} C2) \qquad (1.14)$$

donde *C* hace referencia al patrón de 1s y 0s que queremos buscar, y *C2* a su complemento.

En el ejemplo de la Figura 1.13 tenemos un caso en el que se quieren detectar las esquinas del objeto. Para ello, los elementos estructurantes empleados son los mostrados en a), y en b) tenemos el resultado de la aplicación.

Relleno

Se basa en recorrer la imagen empleando una máscara de 3x3, y tomando como origen el píxel central de dicha máscara. El píxel es convertido a '1' si sus cuatro vecinos (4-*conectividad*) tienen valor '1'. La Figura 1.14 ilustra un ejemplo de esta operación.

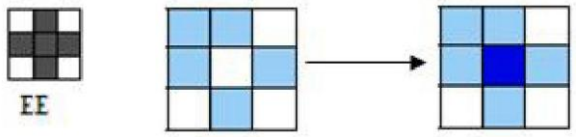


Figura 1.14: Ejemplo de relleno y del EE empleado

- **Limpieza**

Se basa en recorrer la imagen empleando una máscara de 3x3, y tomando como origen el píxel central de dicha máscara. Éste es convertido a '0' si se trata de un píxel aislado, es decir, sus ocho vecinos tienen valor '0'. En el ejemplo de la Figura 22 se aprecia un ejemplo de esta operación.

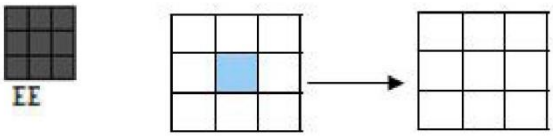


Figura 1.15: Ejemplo de limpieza y del EE empleado

■ **Adelgazamiento ("thinning")**

Esta técnica tiene el objetivo de reducir el tamaño del objeto hasta convertirlo en un trazo mínimamente conectado. El proceso consiste básicamente en lo siguiente: para cada uno de los píxeles del objeto, se comprueba si en ese punto la imagen coincide con un determinado elemento estructurante previamente determinado. En ese caso, el píxel en cuestión se elimina del objeto y así se 'adelgaza'.

$$A \circledR B = A - (A * B) \qquad (1.15)$$

En la Figura 1.16 se muestra un ejemplo de esta operación de adelgazamiento, en la cuál se reduce el grosor del objeto

.

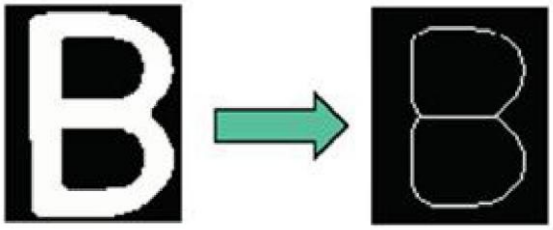


Figura 1.16: Ejemplo de adelgazamiento

■ **Engrosamiento ("thickening")**

Es la operación dual al adelgazamiento, que se basa en engordar los objetos añadiéndoles píxeles externos, pero sin unir objetos previamente separados. El proceso se basa en una transformación 'todo o nada' aplicada a los píxeles del fondo: los que den como resultado un acierto o coincidencia serán etiquetados como objeto, haciendo así mayor el tamaño del mismo.

$$AQB = Au(A*B) \tag{1.16}$$

A continuación se puede observar un ejemplo de la operación de engrosamiento (ver Figura 1.17)

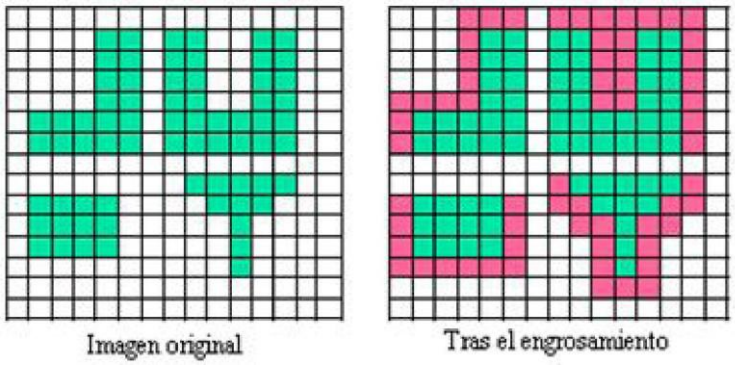


Figura 1.17: Ejemplo de engrosamiento

Mapa de distancias

Para cada uno de los píxeles pertenecientes al objeto, consiste en calcular la distancia de dicho píxel al fondo. La imagen final o mapa de distancias contendrá, para cada uno de los píxeles, dicha distancia. También es posible realizar el mapa de distancias del fondo, en el cual cada píxel perteneciente al fondo recibiría el valor de la distancia que lo separa del contorno del objeto más cercano. La Figura 1.18 muestra un ejemplo de cálculo del mapa de distancias para una imagen.

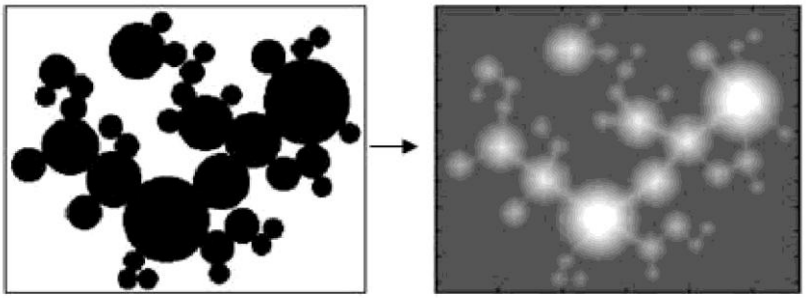


Figura 1.18: Ejemplo de mapa de distancias de una imagen

Esqueletización ("skel") Consiste en obtener la 'estructura interna' del objeto, cuyo cálculo se realiza a partir del mapa de distancias. En la Figura 1.19 tenemos un ejemplo de esqueleto de un objeto rectangular.

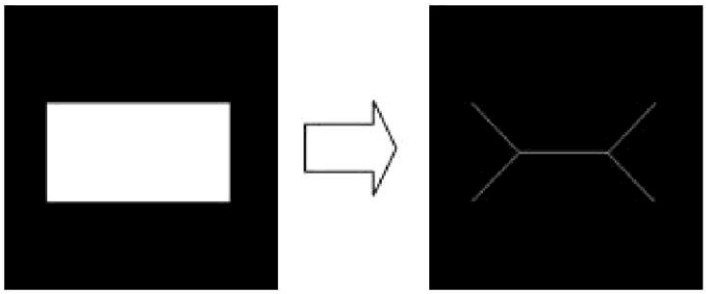


Figura 1.19: Cálculo del esqueleto de un objeto

Reconstrucción Esta transformación morfológica surge a partir de una imagen base, una 'máscara', y otra imagen llamada 'marcador' o 'semilla'. Se trata de obtener otra imagen comenzando la reconstrucción en los puntos que indique la imagen marcador. Los puntos de la imagen marcador señalan los puntos iniciales de la imagen máscara en los cuales hay que comenzar a realizar la operación. La conectividad puede ser 4 u 8, y empezando por dichos puntos se buscan los píxeles conectados, hasta completar la reconstrucción. Un ejemplo de este proceso lo encontramos en la Figura 1.20. Se puede observar la imagen máscara, compuesta por cuatro objetos, y el marcador, compuesta por dos puntos (la línea punteada simplemente se muestra como referencia de los contornos de los objetos). Asimismo también se puede

cuatro
los por

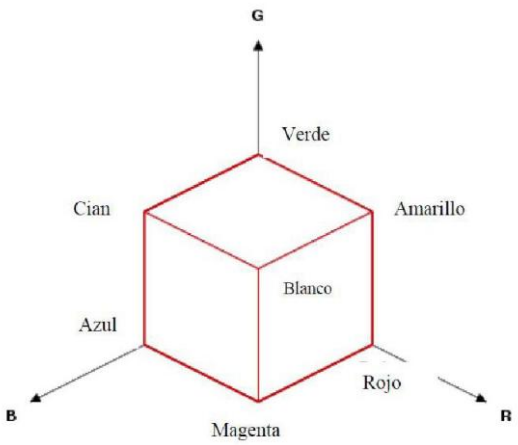
Figura 1.20: Ejemplo de reconstrucción morfológica

1.8. Representación del color

Para la detección de piscinas se han utilizado 4 métodos distintos. Para cada uno de estos métodos se han utilizado las componentes de color que representan las piscinas, tanto en RGB como en HSV. Conviene destacar que la representación del color no es un problema trivial y que ésta influye notablemente en el modo en que se emplean y su eficacia.

1.8.1. Espacio RGB

El espacio RGB es el espacio de color más extendido y el que utilizan la gran mayoría de cámaras de video y fotográficas para construir una imagen de color. Y de ahí, su importancia en visión artificial ya que trabajar con el mismo espacio de color con el que trabaja la cámara con la que se capturan las imágenes permite evitar la alteración de las propiedades del color durante el proceso de segmentación, propia de los errores de conversión y transformación, y por otro lado conseguir una mayor velocidad de segmentación por ahorro de esas operaciones de conversión y redondeo.
El espacio RGB se representa como un cubo dónde un color viene definido por la mezcla de valores de intensidad de tres colores primarios, rojo, verde y azul. Un color viene descrito por una tupla de 3 coordenadas en el cubo. El color negro se representa por (r=0, g=0, b=0) y el color blanco por (r=255, g=255, b=255). La gama acromática de escala de grises está representada por la diagonal del cubo.



1.8.2. Espacio HSV

El espacio HSV [7] representa uno de los espacios de coordenadas más clásicos e intiutivos existentes en la literatura. Su interpretación geométrica viene determinada por un cono de base quasi-hexagonal. Con esta representación del espacio de color, cada color trabaja con 3 componentes básicas: matiz(tono), saturación y brillo. El matiz, h_{HSV} , hace referencia al valor de cromaticidad o clase

de color. La saturación, s_{HSV} , se refiere a las longitudes de onda que se suman a la frecuencia del color, y determina la cantidad de blanco que contiene un color. Contra menos saturado este un color más cantidad de blanco, y contra más saturado este un color menor cantidad de blanco. En definitiva, la saturación representa la pureza e intensidad de un color. Así, la falta de saturación viene dada por la genratriz en la representación del cono HSV. Esa falta de saturación representa la gama de grises desde el blanco hasta el negro. La luminancia, v_{HSV} , se corresponde con la apreciación subjetiva de claridad y oscuridad.

Cuando se quiere representar una imagen en color con un espacio de color HSV, es importante determinar como influyen las componentes de color RGB sobre el espacio HSV. Así, el sistema HSV viene definido por:

Existen muchas variantes de espacios de color intuitivos, en función de cómo se modifique su representatividad, cabe destacar otros espacios como el HSI, HLS, y las variantes de éste último según su forma de construirlo [8][9]. El espacio de color HSV representa mejor que HLS la saturación, aunque tiene peor representación de la luminancia. El HLS viene a representarse como un doble cono donde los vértices determinan la máxima y mínima luminancia.

Capítulo 2

Implementación de algoritmos para el procesado

2.1. Introducción

En el presente capítulo se describe la algorítmica implementada y utilizada en el tratamiento de las imágenes para conseguir la detección automática de humo. El procedimiento en cuestión se basa en las siguientes etapas mostradas en la Figura 2.1.

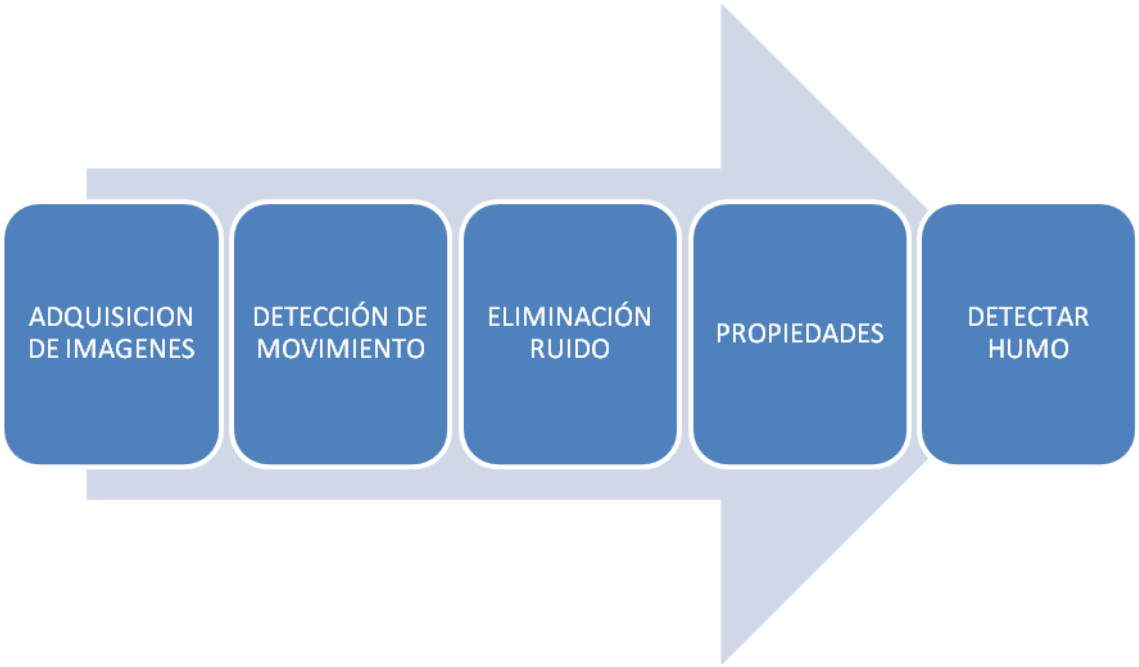


Figura 2.1: Diagrama de bloques de la algorítmica implementada

Como se puede observar en el diagrama de la Figura 2.1, estas 5 etapas son:

Adquisición de las imágenes: Etapa en la que se adquieren y cargan las imágenes sobre las que se desea realizar la detección.

Detección de movimiento: Etapa en la que detectamos el movimiento que se produce ya sea humo o otra cosa.

Eliminación de ruido: Etapa en el que eliminamos pequeñas manchas en la imagen que nos pueden llevar a datos erróneos.

Propiedades: Etapa en la que obtenemos las distintas propiedades de la imagen para su futura utilización.

Detectar humo: Etapa en la que descartamos todo movimiento que no es humo.

2.2. Adquisición de imágenes

En esta etapa primero cargaremos el video que vamos a analizar guardando toda la información que necesitaremos en un futuro en una variable, de esa información obtendremos cuantos frames tiene el video y a qué velocidad van.
De estos frame cogeremos el primero que se vea limpio es decir que este el paisaje de la imagen en calma sin interferencias de objetos no estáticos y esta será la imagen base que utilizaremos para el siguiente proceso.

2.3. Detección de movimiento.

En esta fase detectaremos cualquier movimiento que se vea en el video, para ello cogeremos la imagen original y la restaremos a otra desfasada cierto tiempo este desfase será un bucle que recorra desde la imagen origen hasta el numero de frames que tiene el video, este dato lo obtuvimos en el anterior proceso.

Figura 2.3.1 dector humo

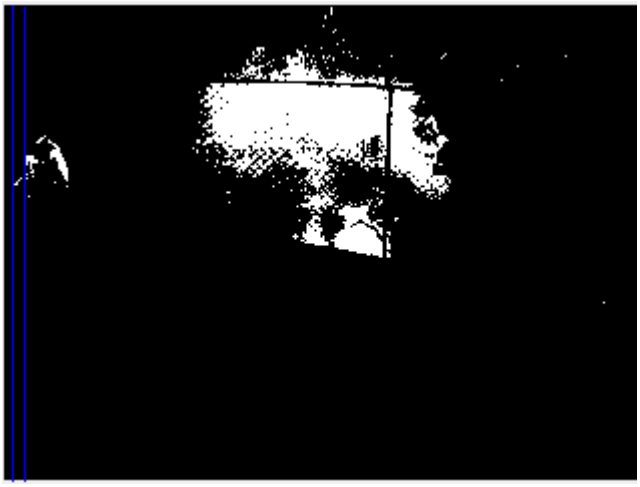


Figura 2.3.2 detección movimiento

Como se ve en la figura 2.3.2 además del humo se aprecia al chico que pasa y ruido que no nos interesa para eliminar ese pequeño ruido pasaremos al siguiente proceso.

2.4. Eliminación de ruido.

Para la eliminación del ruido usaremos dos técnicas de tratamiento de la imagen como son el erode y el dilate, primero erosionaremos la imagen indicando el tipo de erosión y el tamaño que mejo resultado nos dé, en mi caso elijo de disco y tamaño 4 tras pasar por esos dos procesos vemos como queda la imagen.

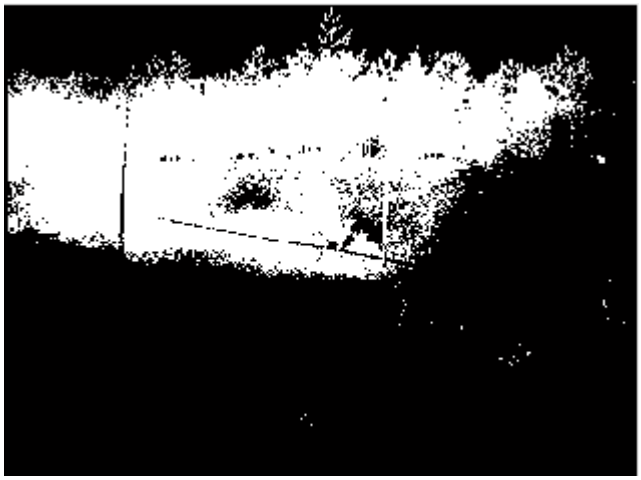


Figura 2.4.1 movimiento con ruido

Figura 2.4.2 movimiento sin ruido

2.4. Propiedades.

Primero conseguiremos todas las propiedades de la matriz ya que en futuros procesos necesitaremos algunos de ellos además haremos un histograma de la imagen resultante de la detección de movimiento.

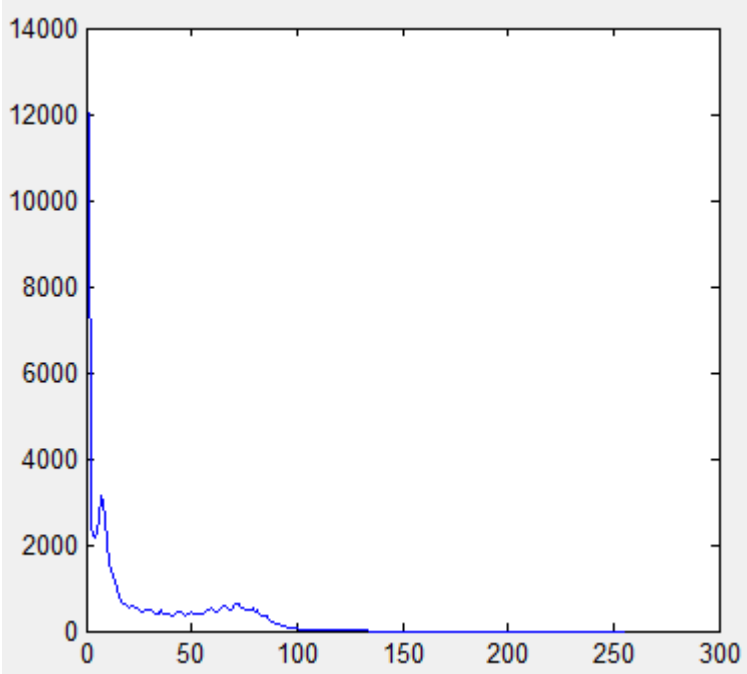


Figura 2.4.1 histograma movimiento

Este histograma nos sirve para saber si hay movimiento en la imagen, ya que si presenta valores distintos a 0 significa que hay movimiento, esto nos sirve para crear la condición para que se inicialice el siguiente proceso.

2.4. Detección de humo.

A continuación describiré el proceso más largo y complejo del programa.
En lo que consiste este proceso es en comprobar como el area de nuestra imagen va evolucionando, teniendo en cuenta que en principio tratándose de humo esta tendrá que ir aumentando con el tiempo
Primero una vez iniciado la condición que detecta el movimiento crearemos un bucle donde almacenaremos en una variable el área de las matrices de la detección de movimiento esta matriz la convertiremos en un numero, para que sea más fácil trabajar con ella.
A continuación restaremos esta área consigo misma pero transcurrido un periodo de tiempo, el valor resultante lo tendremos que comparar con un valor predefinido por nosotros, para asignar este valor hay que tener en cuenta factores como la lejanía del humo que queremos detectar o el transcurso de tiempo que queremos asignarle, ya que dependiendo si el humo está más lejos o más cerca la evolución del área será más rápida o más lenta.
Si este valor es mayor que el que nosotros designamos el programa lo detectara como humo si no simplemente detectara el movimiento.

Capítulo 3

Simulaciones y resultados

En este apartado veremos distintas imágenes procedentes de la simulación final del proyecto y explicando algunas variaciones y errores.

En esta primera figura vemos la detección del humo con el contorno en rojo lo que significa que este movimiento lo detecta como humo y el programa a funcionado.

En la siguiente figura pese a que el humo es bastante abundante no lo detecta porque esta disminuyendo debido al aire y solo detecta humo cuando va en aumento aunque sí que señala con verde el movimiento detectado.

Capítulo 4

Generación de un entorno gráfico de usuario

4.1 Acerca de GUIDE

Las interfaces gráficas de usuario (GUI - Graphical User Interface en inglés), es la forma en que el usuario interactúa con el programa o el sistema operativo de una computadora. Una GUI contiene diferentes elementos gráficos tales como : botones, campos de texto, menús, gráficos, etc.

Existen diferentes leguajes de programación que permiten crear una GUI tales como: C, Visual Basic, TK, etc. solo por mencionar algunos. Todos ellos permiten usar diferentes controles y maneras de programarlos. MatLab nos permite realizar GUIs de una manera muy sencilla usando GUIDE (Graphical User Interface Development Enviroment).

4.1.1 Creando una GUI en MatLab

Una de las tantas herramientas con la que cuenta MatLab, es la creación de GUIs. La forma de implementar las GUI con MatLab es crear los objetos y definir las acciones que cada uno va a realizar. Al usar GUIDE obtendremos dos archivos:

- Un archivo FIG – Contiene la descripción de los componentes que contiene la interfase.
- Un archivo M – Contiene las funciones y los controles del GUI así como el callback

Un callback se define como la acción que llevará a cabo un objeto de la GUI cuando el usuario lo active. Para ejemplificarlo, suponga que en una ventana existe un botón el cual al presionarlo ejecutará una serie de acciones, a eso se le conoce como la función del callback.

4.2 Iniciando GUIDE

Para crear una GUI en MatLab usamos GUIDE, ya sea que tecleemos guide en la ventana de comandos de MatLab o lo ejecutemos del menú principal File –> New -> GUI (Fig 4.2.1).

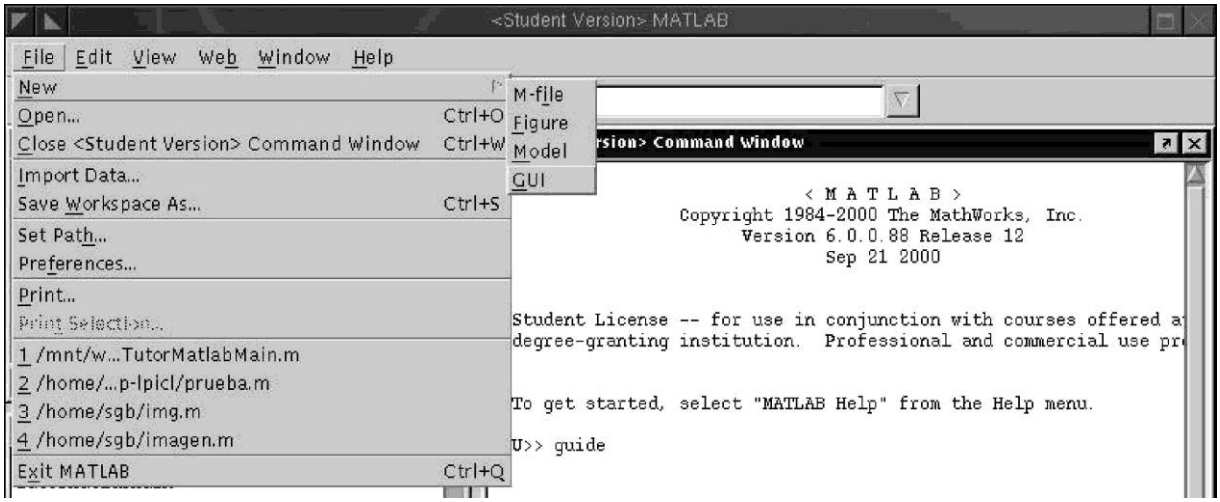


Fig. 4.2.1 – Ejecución de GUIDE usando el menú principal o la ventana de comandos.

Una vez hecho lo anterior MatLab nos mostrará una área de diseño similar a la de la figura 4.2.2. En la parte superior se encuentran los menús y opciones de GUIDE, en la parte izquierda se aprecian los diferentes controles y en la parte central el área de diseño donde pondremos los controles a usar.

4.2.1 Partes de GUIDE

De todo lo anterior mencionaremos las partes más importantes de GUIDE:

Inspector de propiedades - Cada control cuenta con diferentes propiedades y es aquí donde podremos cambiar: el color, el nombre, el tag, el valor, el callback entre otros.

Activar Figura - Una vez que hayamos terminado de diseñar presionamos este botón para activar la figura y poder probar nuestra GUI.

Push Button - Crea un botón

Radio Button - Crea un botón

circular

Edit Text - Crea una campo

de texto

Axes - Crea una área para gráficas.

Frame - Crea un marco que puede contener otros

controles

Static Text – Crea un letrero

4.2.2 Propiedades de los controles

Para entender las propiedades de un control primero crearemos un botón y luego activamos el inspector de propiedades (Fig. 4.2.2.1)

Como se ha mencionado las propiedades varían dependiendo del control a usar, a continuación se explican las más comunes:

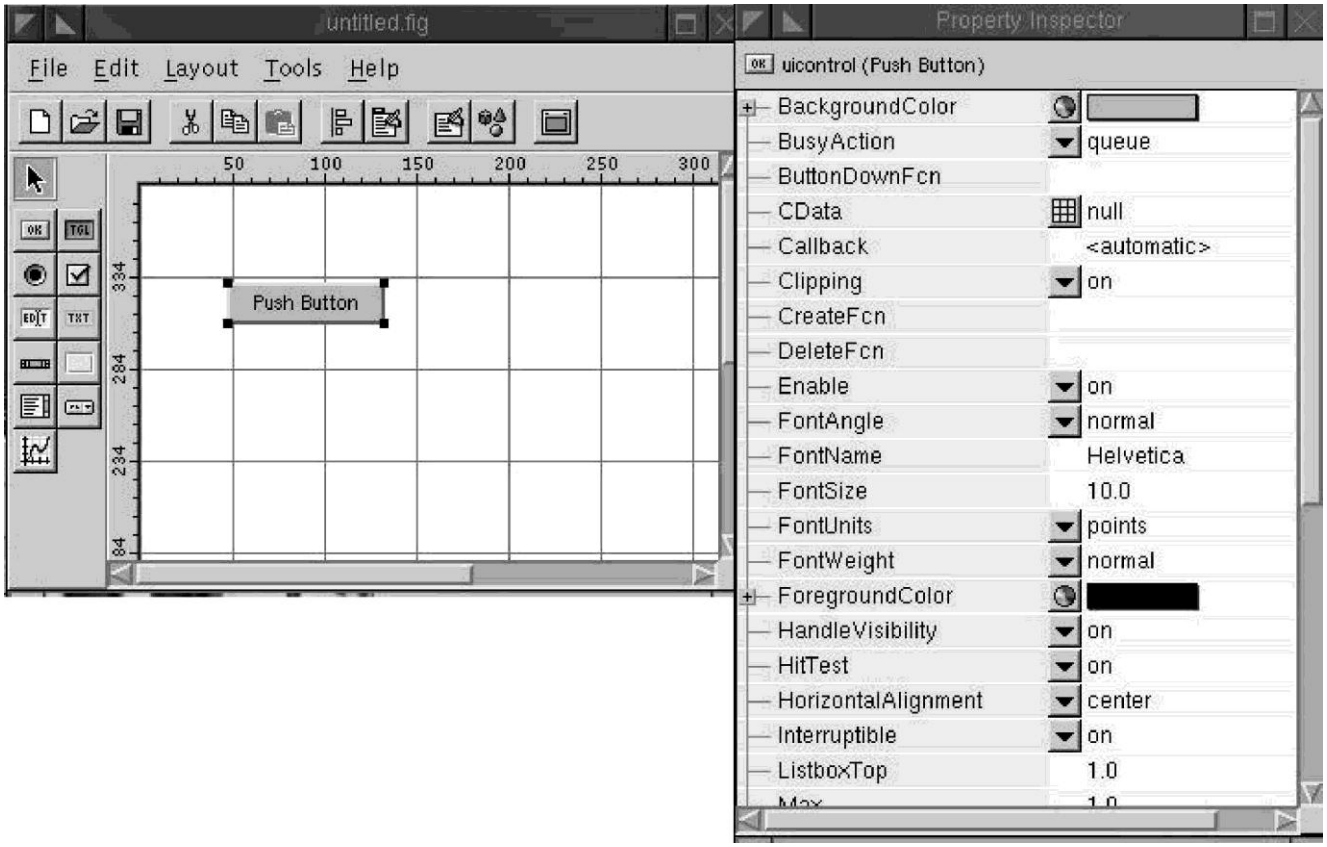


Fig. 4.2.2.1 – Propiedades del control botón

Background Color – Cambia el color del fondo del control

CallBack – La propiedad más importante del control, ya que le dice al control que hacer cuando

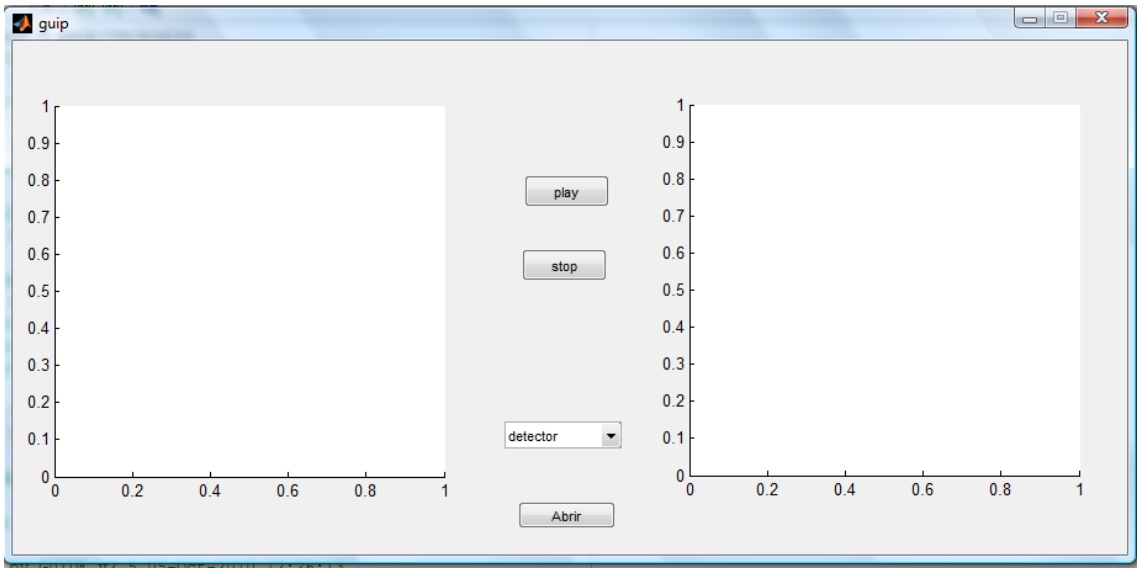
este se active

Enable – Activa o desactiva un control

String – En el caso de botones, cajas de texto, texto estático; es el texto que muestra el control

Tag – Otra de las propiedades más importantes ya que con este es posible regresar datos o identificar al control.

4.3 Entorno gráfico del detector de humos



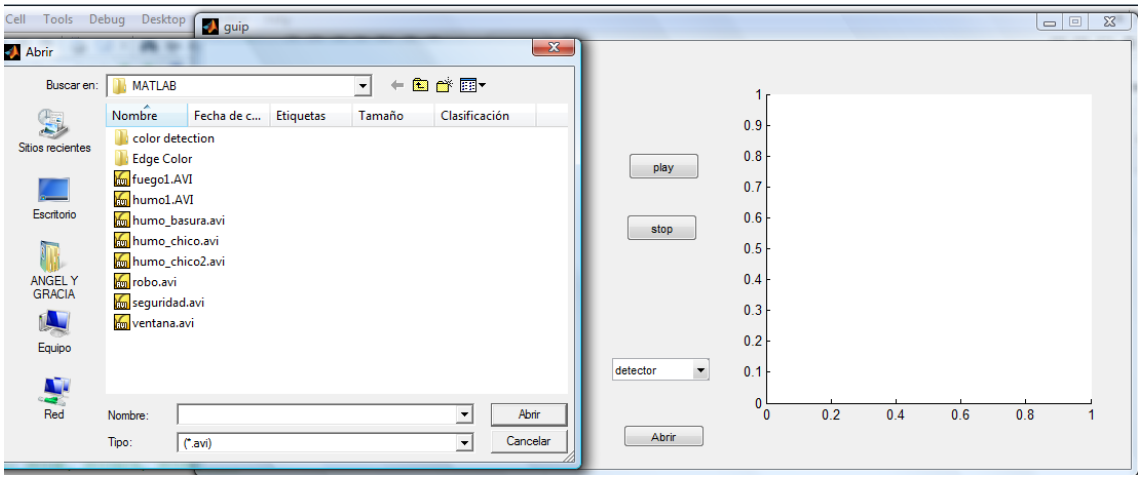
Este es el entorno gráfico del detector de humos, donde vemos en la parte izquierda se reproducirá el video seleccionado, se rodeará con color verde el movimiento y con color rojo la detección de humo.
En la parte central tenemos el botón Abrir para cargar el video que deseamos este sería el código:

```
% --- Executes on button press in abrir.
function abrir_Callback(hObject, eventdata, handles)
% hObject    handle to abrir (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[nombre dire]=uigetfile('*.avi','Abrir');
if nombre==0
    return
end
imagen_video=mmreader(fullfile(dire,nombre));
imagencolor=read(imagen_video,20);
axes(handles.axes1)

imshow(imagencolor)
axis off
handles.img=imagen_video;
guidata(hObject,handles);
```

Y así podremos cargar el video que queramos.



Luego tenemos el selector para poder ver el proceso que seleccionemos así como el histograma para ello tenemos que hacer llamadas a cada uno de los procesos.

```
 case 'proceso1'
     axes(handles.axes2)
     bw_fin=handles.bw;
     guidata(hObject, handles);
     imshow(bw_fin)
      seleccion=2;
 case 'proceso2'
     axes(handles.axes2)
     bw3=handles.bw3;
     guidata(hObject, handles);
     imshow(bw3)
     seleccion=3;
case 'proceso3'
     axes(handles.axes2)
     bw4=handles.bw4;
     guidata(hObject, handles);
     imshow(bw4)
     seleccion=4;
case 'proceso4'
     axes(handles.axes2)
     bw5=handles.bw5;
     guidata(hObject, handles);
     imshow(bw5)

   seleccion=5;
 case 'histograma'
     axes(handles.axes2)
     hist=handles.hist;
     guidata(hObject, handles);
     plot(hist)
```

El botón stop cancelaria el programa también media un llamada que cambiaria una variable la cual pararía el proceso.

```
function stop_Callback(hObject, eventdata, handles)
% hObject    handle to stop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global parar
parar = 1;
```

El botón play seria el que iniciaría el proceso una vez tengamos seleccionado el video y lo que queremos visualizar en la parte derecha.


guip
HAY HUMO
play
stop
proceso1
Abrir

Capítulo 5

Conclusiones y líneas de trabajo futuro

5.1 Conclusiones

En este proyecto final de carrea los objetivos era crear un detector de humo por imágenes a partir de conseguir toda la información característica visuales del humo y descartar todo lo que no es humo mediante comparaciones.
Las principales caracterisricas visuales que se pueden encontrar en el humo son:

- El color, aunque el color del humo puede variar dependiendo del material que se este quemando o incluso reflejos del sol y del entorno.
- Otro seria el origen del incendio ya que todo humo sale de un mis mo foco y se va expandiendo, tambien podemos tener problemas si hay aire y parte el humo en algun problema.
- El mejor factor es la expansion del humo midiendo la areas ya que el humo tienede a aumentar su area conforme pasa el tiempo, aquí hay que tener en cuenta para que distancia queremos abarcar ya que si queremos que se active cuando aumenta x respecto la matriz de la imagen ese aumento sera más moderado en largas distancias.

5.2 Líneas de trabajo futuro

Como líneas de trabajo futuro se puede siempre mejorar el programa para que tenga un mejor rendimiento en distintos tipos de video así como mejor optimización de la programación para que funcione más fluido a la hora de realizar todos los bucles y procesos. También se podría aplicar mediante video en directo con una webcam que sería más lo que se busca para su utilización de la vida real.

Bibliografia

1. González, R.C., Wintz, P. (1996). Procesamiento digital de imágenes, Addison-Wesley.
2. Acharya, T., Ray, A. K. (2005). Image processing: principles and applications, John Wiley & Sons.
3. Alegre, E., Sánchez, L., Fernández, R. Á., Mostaza, J. C. (2003). Procesamiento Digital de Imagen: fundamentos y prácticas con Matlab, Universidad de León.
4. Rafael C. González, Richard E. Woods, Steven L. Eddins (2009). Digital image processing using Matlab, Gonzalez, Woods, & Eddins.
5. The MathWorks Image Processing Toolbox, for use with Matlab.2006
6. www.youtube.com
7. www.mathworks.es