UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

I.T. Telecomunicación (Sist. de Telecomunicación)

"Diseño de un interfaz hombre máquina utilizando el entorno MATLAB para la detección y eliminación de ruido impulsivo en imágenes"

TRABAJO FINAL DE CARRERA

Autor:
Daniel Seguí Simó

Director:
Vicent Vidal Gimeno

GANDIA, 2012

Agradecimientos

En primer lugar, a Vicent Vidal por toda su ayuda desinteresada, por ser el tutor de este proyecto y todo lo que he aprendido.

A Lupita Sánchez, por toda su ayuda y todas sus aportaciones al proyecto, y porque sin ella las horas en el laboratorio hubiesen sido muy aburridas.

A mis padres por la paciencia y espera, y a toda mi familia en general por los ánimos para llevar a cabo el trabajo realizado.

A todos mis amigos, porque sin ellos nada sería lo mismo.

Gracias.

ÍNDICE

ÍNDICE DE FIGURAS

RESUMEN

En la actualidad, el uso de imágenes digitales está presente en la mayoría de tecnología que usamos a diario, tanto en el ámbito doméstico como en el profesional.

La valoración de calidad de una imagen médica, militar o industrial, como puede ser una mamografía o un mapeado de gran resolución, es uno de los puntos más importantes en cualquier acabado de un equipamiento tecnológico.

Por ser una señal, cualquier imagen será susceptible de sufrir ruido, ya sea en la toma y procesado, como en la transmisión de ésta por cualquier medio.

El presente proyecto basa sus análisis de imágenes digitales en uno de los tipos de ruido más frecuente: el ruido Impulsivo, también llamado Sal y Pimienta.

Se analizará este ruido y se implementará un filtro PGFM (Peer Group y Métrica Fuzzy) para conseguir la recuperación de la imagen que ha sido degradada.

Para estudiar este filtro y determinar cuáles serán los mejores parámetros adecuados para cada tipo de imagen, hemos creado un GUI (Interfaz Gráfica de Usuario) mediante el software de Matlab, que nos facilitará un sencillo y potente entorno gráfico para manipular imágenes, pudiendo así añadir ruido y trabajar con el filtrado para poder recuperarlas.

Aparte del ruido impulsivo, en la memoria del proyecto analizaremos brevemente otros tipos de ruido comunes en las imágenes digitales, como son el ruido Gaussiano y el Uniforme, así como un rápido vistazo al ruido Poisson y al Speckle.

Para caracterizar los resultados del filtro propuesto (PGFM), se propone basar los análisis en cuatro tipos de medida de calidad: PSNR para evaluar la eliminación del ruido, MSE para medir objetivamente la coincidencia con los píxeles originales, MAE para evaluar la preservación del detalle de la imagen y NCD como error de percepción humana.

De acuerdo con los resultados obtenidos, el programa nos permitirá guardar los parámetros utilizados y clasificarlos en una tabla Excel, haciendo referencia a la imagen correspondiente.

Estos resultados facilitarán al grupo de investigación el análisis heurístico del comportamiento de los diferentes parámetros que intervienen en los filtros.

INTRODUCCIÓN

Las imágenes digitales están a la orden del día en telecomunicaciones y en la tecnología en general. Desde aplicaciones de fotos y transmisión de vídeo a nivel de usuario corriente, hasta en la ingeniería, tratamiento de imágenes médico, industrial o militar.

Al igual que cualquier señal, las imágenes pueden contaminarse durante su adquisición o cuando son transmitidas. Sus píxeles han sido modificados o alterados, perdiendo así los valores originales.

Encontramos indispensable, cada día más, el poder obtener sin errores y con alta resolución cualquier imagen con la que trabajemos.

¿Por qué un programa para el tratamiento de la imagen y el ruido impulsivo?

Con este proyecto, tratamos de facilitar el trabajo y desarrollo en el campo del tratamiento digital de imágenes. Creamos una interfaz que ayude tanto al ingeniero que se dedica a este campo de las telecomunicaciones, como al estudiante y/o cualquier usuario que desee experimentar y aprender.

Se trata de un GUI (*Graphical User Interface*) con un diseño sencillo y de fácil uso, programado en Matlab, y que reúne algoritmos de degradación y filtros de recuperación de imágenes.

En este proyecto se usará el programa para trabajar con los filtros para ruido Impulsivo implementados, aunque también se podrá trabajar con ruido Gaussiano o con los dos a la vez.

Nuestro GUI presenta una manera rápida de cargar imágenes y comprobar en breves instantes cuál es el resultado de utilizar unos filtros u otros, con los parámetros que necesitemos comprobar para cada tipo de imagen. Ante el clásico método en Matlab de estar repitiendo o modificando siempre los mismos comandos, o abriendo múltiples ventanas, etc. con las pérdidas de tiempo que esto supone y el consiguiente gasto innecesario de memoria para el ordenador.

Así pues, el GUI facilitará al grupo de investigación la tarea de trabajar con el filtrado PGFM y poder estudiar qué parámetros serán los más adecuados para cada tipo de imagen.

¿Qué es un GUI de Matlab?

GUI significa Interfaz Gráfica de Usuario (en inglés, *Graphical User Interface*).

Nace, al igual que la mayoría de software aparecido en la última década, para eliminar el uso arcaico de comandos y facilitar en pocos clics, el uso de cualquier programa.

Como Matlab funciona casi todo a base de funciones y comandos, ayuda también, por ejemplo, al desarrollo y trabajo en programas que requieren procesos repetidos y automatizados.

¿Por qué desarrollar el programa con MATLAB?

La primera razón, fue porque las funciones y algoritmos con los que trabajamos para recuperar las imágenes ya estaban implantados en Matlab o funcionaban bien con él y, ya que Matlab disponía de un sencillo y potente editor de GUIs, decidimos que era la mejor opción.

A su vez, el código de programación de GUIs en Matlab se presenta de manera sencilla a todos aquellos ingenieros y científicos que no poseen amplios conocimientos en materia de programación. Y existe también una comunidad de personas que colabora ampliamente en el desarrollo de aplicaciones: compartiendo sus avances y cuestiones en materia de programación de Matlab, desde un simple código hasta complejas aplicaciones, que harán que un usuario novato pueda aprender rápidamente sin mucha dificultad.

Por último, destacamos también la flexibilidad de poder utilizar cualquier sistema operativo (entre Linux, MAC y Windows) para poder trabajar con nuestro programa sin tener que cambiar el código, y la integración con otros programas y software como procesado de señales, aplicaciones de vídeo y audio, suites de ofimática, etc.

CONCEPTOS GENERALES

RUIDO EN IMÁGENES

Definiremos el ruido como toda aquella señal no deseada que proviene de la naturaleza y que perturba nuestra señal. Está presente en todas las telecomunicaciones y es inevitable.

La forma teórica de una señal más el ruido que interfiere es esta:

$$g(x,y) = f(x,y) + r(x,y) \qquad (1)$$

En nuestro caso, se trata de toda aquella información o píxeles que estarán contaminando nuestra imagen.

El origen de este ruido puede deberse a diversas causas, desde que se adquiere la imagen (por errores en los fotosensores, luz, temperatura, etc.) hasta que esta se transmite (interferencias en el canal de transmisión, desde los mismos sensores, cableado y/o el guardado final para su edición, etc.). Podrá estar generado externamente o bien, internamente por los propios componentes que capturan la información.

Este ruido se manifestará en forma de variaciones en el brillo y el color, y que se apreciarán a simple vista por puntos indeseables en la imagen o un granulado completo de la imagen haciendo que ésta pierda su calidad.

SENSORES Y CAPTURA

Los sensores que capturan la imagen pueden dividirse en dos categorías: fotoquímicos y fotoelectrónicos.

Las películas fotográficas (positivos y negativos) son sensores fotoquímicos; pueden tomar y grabar la imagen al mismo tiempo pero son difíciles de digitalizar. El ruido se manifiesta de manera que aparece un granulado aleatorio durante la exposición de la película, que podrá ser modelado como modelo de Poisson o como un modelo Gaussiano. Además del grano en la película, el ruido puede aparecer debido al polvo que va acumulándose en las lentes, sensores y los negativos durante el proceso de toma, posterior revelado y digitalización.

Los sensores fotoelectrónicos tienen como principal ventaja, el poder digitalizar la imagen directamente. Los más conocidos, los sensores CCD, consisten básicamente en celdas

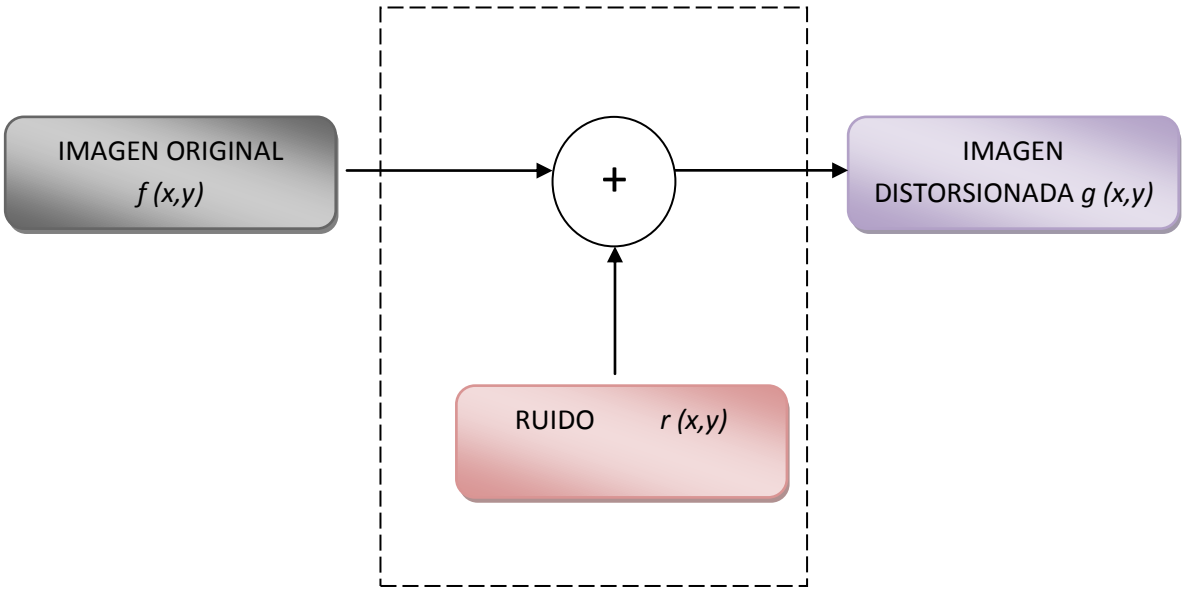
fotosensibles: cargas eléctricas que van transmitiendo información de una a otra, en escaneos de filas compuestas por éstas celdas. Aquí, el ruido puede aparecer aleatoriamente por las fluctuaciones de fotones en la superficie de los sensores.

Para más información sobre sensores y ruido en imágenes puede consultarse [10], [11] y [12].

DEGRADACIÓN DE IMÁGENES

Aunque una imagen no haya sido afectada por ruido en el proceso de toma, procesado y guardado; podrá ser degradada mediante algoritmos para simular lo que podría haber sido el ruido indeseable.

El proceso de degradado de la imagen podemos verlo en un simple diagrama:



RECUPERACIÓN DE IMÁGENES

El proceso de recuperación de la imagen degradada servirá para mejorar la calidad visual de la imagen; restauraremos parte de ésta o la imagen al completo. Los algoritmos determinarán qué píxeles son ruidosos y cuáles no, y recuperaremos la mayoría de píxeles, asemejándose de nuevo la imagen lo más parecido a la original.

EJEMPLO DEL PROCESO

Ilustración original con 3 pingüinos que es afectada por ruido impulsivo y posteriormente recuperada mediante filtrado:

Figura 1. Imagen original sin ruido

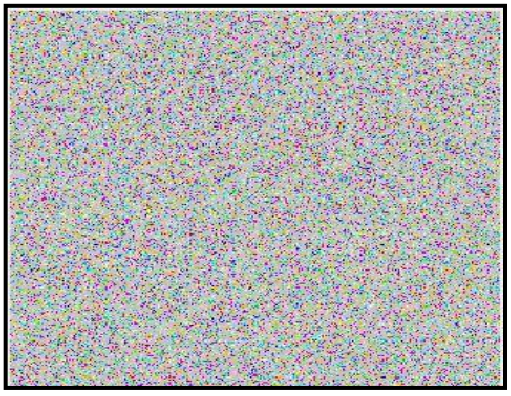


Figura 2. Ruido

Figura 3: Imagen ruidosa

Figura 4. Imagen filtrada

TIPOS DE RUIDO EN IMÁGENES

Hay diversos tipos de ruido que pueden afectar una imagen, que clasificaremos principalmente en aditivos y en multiplicativos.

Un ejemplo de ruido multiplicativo es la variable iluminación, que podemos decir que es el tipo de ruido más común en las imágenes.

Cuando hablamos de ruido aditivo nos referimos, principalmente, a que hay ruido impulsivo o ruido gaussiano. El ruido impulsivo altera aleatoriamente el valor de un grupo de píxeles: en una imagen binaria esto quiere decir que algunos píxeles blancos se volverán negros y viceversa. Por su parte, el ruido gaussiano afectará a los valores de cada píxel con una función de densidad de probabilidad Gaussiana.

La lista de funciones utilizadas en nuestro programa para aplicar ruido se encuentra en el anexo.

IMPULSIVO

El ruido “Sal y pimienta” (Salt and Pepper, en inglés) consiste en que los píxeles afectados son de diferente color o intensidad al resto de píxeles que los rodean.

Se trata de píxeles “encendidos” y “apagados”, es decir, en imágenes binarias son píxeles blancos (brillantes) sobre negros y píxeles negros (oscuros) sobre brillantes. Toman valores máximos y mínimos, no teniendo relación con el valor de la imagen ideal.

Este ruido afectará a la imagen según el valor de su función de densidad de probabilidad.

Figura 5. Imagen original sin ruido

Figura 6. Imagen afectada por ruido impulsivo

Dentro de Matlab utilizaremos la función *imnoise* para introducir ruido en las imágenes.

Para *sal y pimienta*, la función cargará la imagen original y el porcentaje de píxeles defectuosos que deseemos (densidad de ruido). El porcentaje principal de la densidad de ruido es del 5%.

Ejemplo de Matlab:

J = IMNOISE(I,'salt & pepper',D)

Añade ruido impulsivo S&P a la imagen I, donde D es la densidad de ruido. Afectará aproximadamente a "D x Nºpíxeles de I" píxeles.

GAUSSIANO

El ruido Gaussiano presenta una función de densidad de probabilidad que sigue la curva o campana de Gauss (llamada así por su descubridor, quién la halló gracias al estudio de teoría de errores en la medida de magnitudes) [9].

La forma que adquiere su función de densidad es:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \qquad (2)$$

Donde podemos ver que los parámetros μ y σ son precisamente la esperanza y la desviación típica, respectivamente.

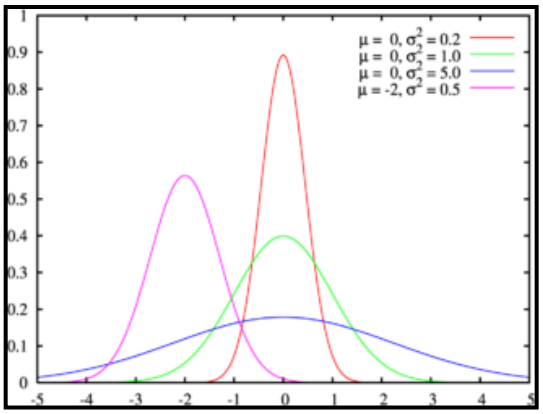


Figura 7. Ejemplo de curva gaussiana

Cuando se presenta este ruido, aparecen pequeñas variaciones en la imagen, cambiando el valor de cualquier píxel (toda la imagen se verá afectada) de manera que este puede ser diferente cada vez si tomáramos la misma imagen. En imágenes binarias, por ejemplo, aumenta o disminuye el valor del gris y es independiente de los valores que tomará la imagen.

Ejemplo de Matlab:

J = IMNOISE(I,'gaussian',M,V):

Añade ruido blanco Gaussiano de media M y varianza V a la imagen I. Cuando no se especifica, M y V toman los valores 0 y 0.01, respectivamente.

Figura 8. Imagen original sin ruido

Figura 9. Imagen afectada por ruido gaussiano

UNIFORME

El ruido sigue una distribución uniforme, por lo que la probabilidad de que un píxel tome un valor diferente es constante.

Figura 10. Imagen original sin ruido

Figura 11. Imagen afectada por ruido uniforme

OTROS TIPOS DE RUIDO

A continuación presentamos dos tipos de ruido que no aparecen en nuestro programa pero que son también comunes en las imágenes.

POISSON

La distribución de Poisson viene relacionada normalmente con fenómenos que se producen a lo largo de un intervalo de tiempo, espacio,… si estos se producen con una media conocida e independiente del último suceso.

$$P_\lambda(k) = e^{-\lambda}\frac{\lambda^k}{k!} \qquad (3)$$

Dónde $\lambda > 0$ es un parámetro de distribución y k es el número de fallos, o de veces que se obtiene un evento.

En Matlab sigue esta función:

J = IMNOISE(I,'poisson')

Figura 12. Imagen original sin ruido

Figura 13. Imagen afectada por ruido Poisson

SPECKLE

Es un tipo de ruido uniforme de carácter multiplicativo, dónde la señal es resultado de la multiplicación de dos señales.

En Matlab sigue esta función:

J = IMNOISE(I,'speckle',V)

Se añade el ruido multiplicativo a la imagen I usando la ecuación J= I+ n*I, donde n es una distribución aleatoria de ruido con media 0 y varianza V.

Figura 14. Imagen original sin ruido

Figura 15. Imagen afectada por ruido Speckle

FILTRADO DE IMÁGENES

MÉTODO PGFM

El filtro PGFM se utiliza para reducir el ruido impulsivo y realiza tres fases para eliminar el ruido de la imagen afectada. La primera fase consiste en clasificar los píxeles en corruptos, no corruptos y no diagnosticados. Los píxeles no diagnosticados son clasificados en la segunda fase como corruptos o no corruptos y, por último, en la tercera fase se filtran los píxeles etiquetados como corruptos.

Este proceso (PGFM) utiliza el concepto de *Peer Group* y la métrica Fuzzy. El *Peer Group* es el conjunto de pixeles de una ventana W similares al pixel central x_i, de acuerdo a una medida de distancia M [8], [4], y viene dado por la siguiente expresión:

$$\{ x_j \in W : M(x_I, x_J) \geq d \},$$

Donde *d* es el umbral que decide si dos píxeles son próximos (por defecto se toma 0.95) y M es la métrica Fuzzy. Una de las métricas que suele utilizar es:

$$M(xi, xj) = \prod_{l=1}^{3} \frac{\min\{xi(l), xj(l)\} + k}{\max\{xi(l), xj(l)\} + k} \quad (4)$$

El parámetro *k* se tomará por defecto como *k*=1024 por ser una configuración adecuada para mantener la calidad de la imagen.

En la tercera fase, etapa de corrección, se aplicará el filtro AMF (Filtro de Media Aritmética) a los píxeles marcados como dañados, sustituyéndolos cada uno por sus píxeles vecinos no corruptos.

Una descripción más detallada de este método y del estudio de parámetros utilizados la podremos encontrar en la tesis de J.G..Estruch.Camarena (2009) [6].

DIFUSIÓN

El método de restauración para imágenes afectadas por ruido Gaussiano, el Filtro Difusivo No Lineal, se basa en el uso de ecuaciones de difusión no lineal, que generalmente aparece asociado a un problema variacional y, se puede obtener a partir de la minimización de las funcionales apropiadas. La elección de un determinado funcional depende del objetivo específico de interés, del tipo de imagen a restaurar.

Consideremos la funcional (5),

$$J(u,\beta,\mu,\epsilon) = \int_{\Omega} (\sqrt{\beta^2 + \left\|\vec{\nabla}u\right\|^2} + \frac{\mu}{2}(u - I_0)^2 + \frac{\varepsilon}{2}(\vec{\nabla}u)^2)\, d\vec{x}, \qquad (5)$$

donde I_0 es la imagen observada (con ruido), μ y ε son constantes y Ω es una región convexa de $\mathbb{R}^2$ que constituyen el espacio de apoyo de la superficie *u(x, y),* que representa la imagen. El primer término en la funcional para β = 1 representa el área de la superficie que representa la imagen, el segundo término da cuenta de la distancia entre la imagen observada y la solución deseada *u* (Imagen filtrada), y el tercer término controla la regularidad de la solución. Consideraremos el problema de minimización,

$$min_u\, J(u,\beta,\mu,\epsilon) \quad \text{sujeto a} \quad \frac{\int_{\Omega}(u - I_0)^2\, d\vec{x}}{\int_{\Omega} d\vec{x}} = \sigma^2, \quad (6)$$

es decir, buscamos la imagen u que minimiza el funcional $J(u,\beta,\mu,\epsilon)$ y presenta una variación con respecto a la imagen observada I_0 igual a σ^2. En nuestro trabajo hemos estimado σ, tomando la desviación media absoluta del coeficiente wavelet empírica de la escala más fina y dividiendo por 0.6745 [4]. Para todas las imágenes estudiadas, el wavelet fue un Daubechy de orden 25.

Para la discretización del tiempo se utiliza un esquema semiimplícito, y para resolver las ecuaciones se utiliza el operador de división aditivo alternativo (AOS). La selección del tiempo de parada en la ecuación de difusión fue el propuesto por Mrázek y Navara, con base en el criterio de correlación; σ, a priori, se desconoce, pero es importante conocer su valor para minimizar la ecuación (6).

COSENO

El método de filtrado por Coseno utiliza el concepto de Peer Group y la métrica Coseno. Es un proceso muy similar al descrito anteriormente para el método PGFM. Uno de los cambios más significativos es el uso de la métrica del coseno:

Dados *x*, *y* ∈ R^n el coseno del ángulo entre dichos vectores tiene la expresión

$$x^t y \Big/ (\|x\|_2 \cdot \|y\|_2) \quad (7) \ .$$

Considerando los vectores *x* e *y* como píxeles de la imagen obtenemos una métrica acotada entre 0 y 1 que sirve para evaluar la proximidad entre ambos.

MEDIDAS DE CALIDAD

La imagen ruidosa pasará por un proceso de filtrado y podremos evaluarla a fin se ser comparada con la original (libre de ruido), con el objetivo de ver la similitud de la recuperada con la original.

Evaluaremos la calidad del filtro, basándonos en estos 4 tipos de medidas y centrándonos en la eliminación del ruido y el la conservación de los detalles [4], [8].

PSNR – Peak Signal to Noise Ratio

Esta medida de calidad calcula la relación Señal/Ruido entre las imágenes original y la recuperada.

$$PSNR = 10log_{10}\left(\frac{255^2}{MSE}\right) \quad (8)$$

Es la función que sirve para expresar la capacidad de eliminación de ruido [7].

MSE – Mean Squared Error

Error cuadrático Medio

$$MSE = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}\left\|O_{i,j} - O_{i,j}\right\|}{MN} \quad (9)$$

M y *N* indican las dimensiones de la imagen. $O_{i,j}$ y $O_{i,j}$, muestra los píxeles originales y los píxeles restaurados localizados en (*i*, *j*), respectivamente [7], [8].

RGB no es un espacio perceptualmente uniforme en el sentido de que las diferencias entre los colores en este espacio no se corresponden con diferencias de color percibidas por los seres humanos. Por ello hallaremos también la Diferencia de Color Normalizada (NCD), comentada más adelante.

MAE – Mean Absolute Error

La función más utilizada para evaluar la conservación de detalles.

$$MAE = \frac{\sum_{i=1}^{NM}\sum_{q=1}^{Q}\left|F_i^q - \hat{F}_i^q\right|}{NMQ} \quad (10)$$

M, *N* son las dimensiones de la imagen, *Q* es el número de canales de la imagen (*Q* = 3 para imágenes en color), y F_i^q y $\hat{F}_i^q$ representa la q-ésima componente de la imagen original y filtrada al píxel *i* respectivamente [8].

NCD – Normalized Color Difference

Diferencia de Color Nomalizada

Muy utilizada ya que mide la percepción humana.

$$NCD = \frac{N\Delta E}{\sum\sqrt{(L_{0i}^*)^2 + (u_{0i}^*)^2 + (v_{0i}^*)^2}} \quad (11)$$

donde L * representa los valores de luminosidad y u *, v * valores de crominancia correspondientes al original *0i* y restaurado *xi* muestras expresadas en la CIE LUV espacio de color [8].

DESARROLLO DE LA INTERFAZ

A continuación, revisaremos brevemente algunos puntos básicos en la programación de GUIs en Matlab:

La mayoría de puntos de la interfaz se compone de los *botones* o *controles*, que siguen el comando *uicontrol* para su creación. *Uicontrol* permite la edición de todas las propiedades de cada control, ya sea desde la interfaz de código o desde la interfaz de edición gráfica.

- *Pushbuttons:* invoca una acción al pulsarlo.
- *Checkbox:* estado de una opción
- *Radio buttons:* selecciona una opción
- *Sliders:* desplazarse por un rango de valores
- *Pop-up Menu:* muestra lista de opciones
- *List Box:* muestra lista de opciones deslizable
- *Editable textboxes:* permite la introducción y edición de texto
- *Static textboxes:* muestra texto (string)
- *Frames:* separan elementos de una figura

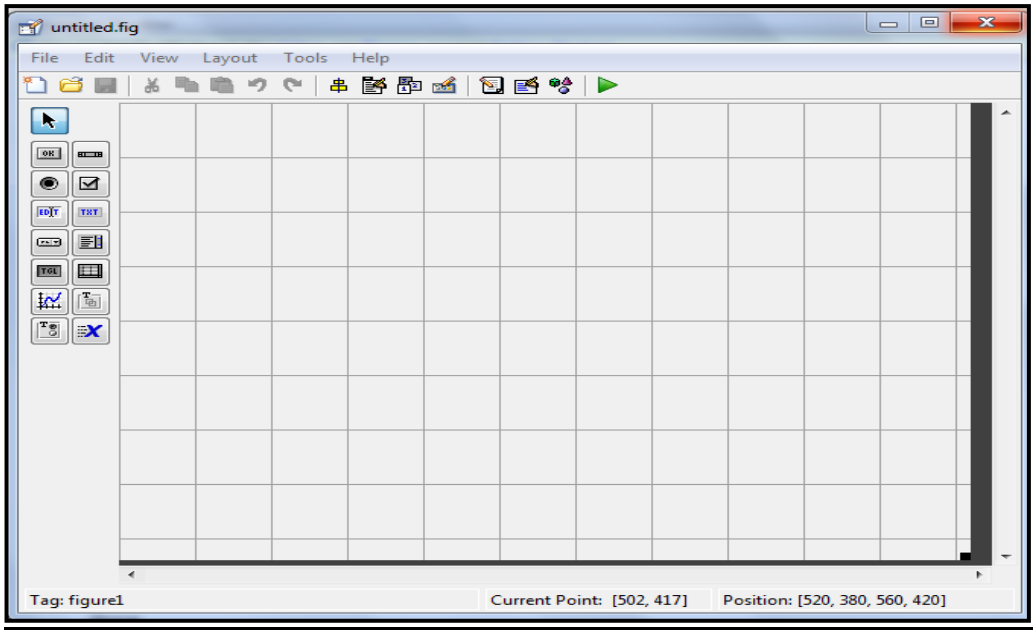


Figura 16. Editor de GUI Matlab

HANDLES, GUIDATA

Los valores que corresponden a cada propiedad de cada elemento, y los valores de las variables que se utilizarán a lo largo del programa, se almacenan en una estructura y sólo pueden ser accedidos mediante una identificación única. Este *identificador* se asigna en:

```
handles.output=hObject;
```

A su vez, necesitará de *guidata* para guardar los datos de la aplicación.

```
guidata(hObject, handles);
```

Salvará las variables y propiedades de cada elemento en la estructura de datos de la aplicación, por lo que el final de cada subrutina debería incluir *guidata (hObject, handles)*;.

GET, SET

Mediante el identificador handles. y el nombre del control, podemos utilizar las sentencias get y set para tomar o introducir valores.

`valores= get(handles.slider5,'Value');` - Obtenemos los datos del slider 5 mediante GET y utilizando el identificador handles.

`set(handles.text1,'String',valores);` - asignamos el valor obtenido en "valores" en el text1.

UIGETFILE

Una función muy importante para nuestra interfaz, ya que nos permite abrir un archivo, así como obtener su nombre y dirección. Con ello se nos abrirá una ventana que permitirá buscar en el directorio de carpetas de nuestro ordenador.

```
[FileName Path]=uigetfile({'*.jpg;*.bmp;*.gif;'},'Abrir Imagen');
```

SWITCH

Este comando es similar a las sentencias *"if"*. Permite realizar una subrutina o tarea según la opción que se ha escogido, por ejemplo, en un *popupmenu*.

FOPEN, FPRINTF

Fopen abrirá un archivo nuevo o ya creado para poder trabajar con él mediante comando fprintf.

Fprintf es el comando que nos permite exportar los datos a un archivo de texto (y a su vez, poder exportar a Microsoft Office Excel). Muestra y escribe en el documento asignado cualquier valor o string para el que se programe.

Este ejemplo crea y abre el archivo MEDIDAS_CALIDAD.txt y la opción *'a'* indica que debe escribirse desde la última línea del documento que se haya guardado.

```
fid = fopen('MEDIDAS_CALIDAD.txt','a');
```

A continuación, *fprintf* escribirá en MEDIDAS_CALIDAD.txt, el string *x* que recoge el tipo de color de la imagen. *'%10.5s'es la separación y el tamaño del campo.*

```
fprintf(fid,'%10.5s',x);%tipo de color: color,gris,o byn
```

Para la elaboración del proyecto y desarrollo de GUI en Matlab se ha consultado los manuales [1], [2] y [3]. Encontraremos más información acerca del uso de botones, controles, animaciones y desarrollo de simulaciones, además de edición de gráficos.

DESARROLLO DEL PROYECTO

INICIO

Figura 17. Intro programa

Nuestro proyecto se compone principalmente de dos archivos m-files de Matlab y sus correspondientes archivos .fig.: GUIRIMPULSIVO e INTERFAZ 1

GUIRIMPULSIVO corresponde a la portada del programa, dónde nos aparecen 2 opciones: abrir el documento *GUIA* en formato pdf y/o continuar y ejecutar el programa principal, dónde ya podremos trabajar con nuestras imágenes.

Figura 18. Portada guía de usuario

INTERFAZ1 corresponde a la parte principal del programa, que explicaremos a continuación:

INTERFAZ GUI MATLAB

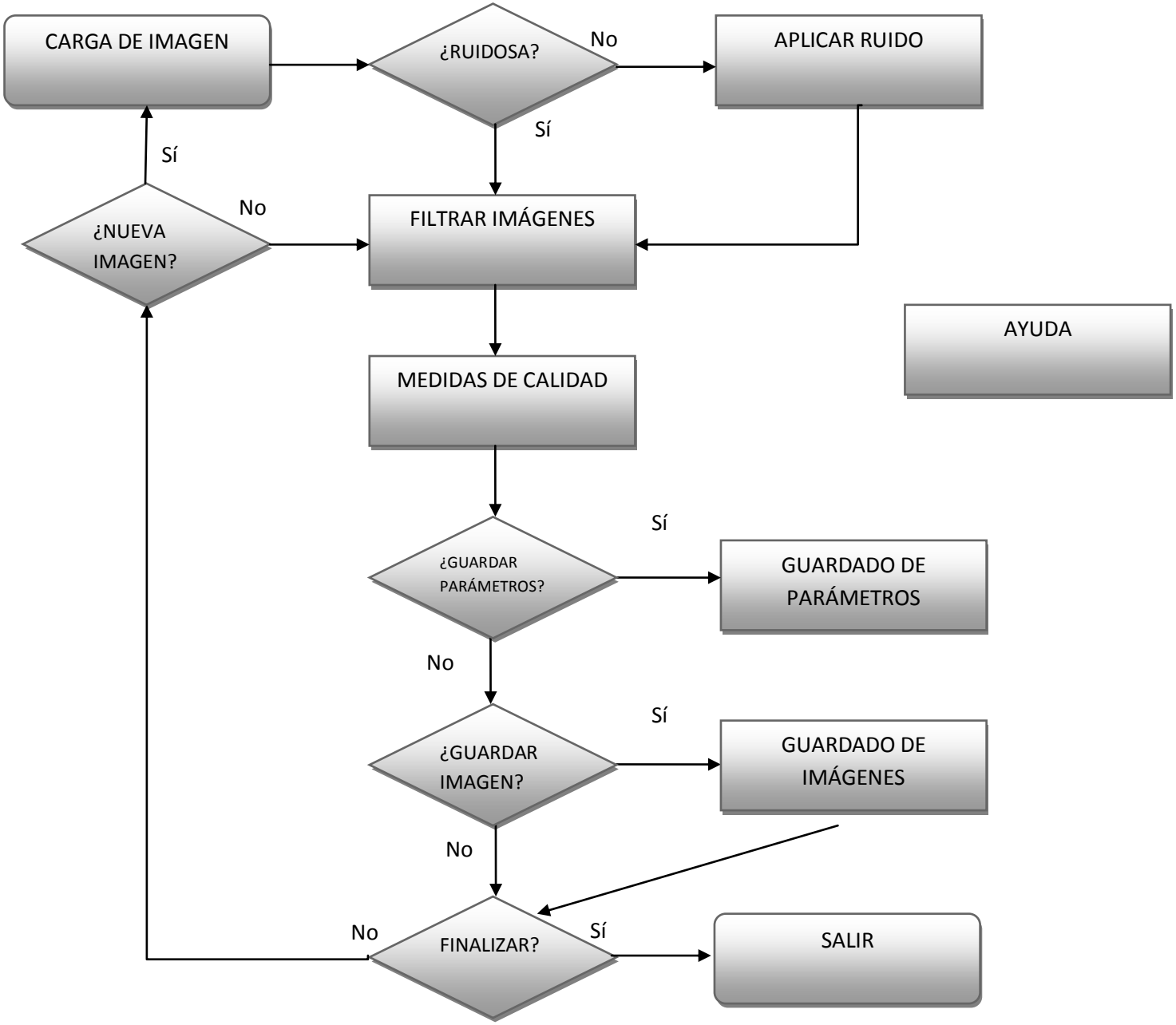
Figura 19. Programa principal

Nuestro programa está compuesto por una interfaz principal que se compone de 7 módulos de trabajo y visualización de imágenes.

Los módulos son:

1. CARGA DE IMAGEN
2. VISUALIZACIÓN DE IMÁGENES
3. APLICACIÓN DE RUIDO
4. FILTRADO DE IMÁGENES
5. MEDIDAS DE CALIDAD
6. GUARDADO DE IMÁGENES
7. PARÁMETROS UTILIZADOS
8. GUARDADO DE PARÁMETROS
9. AYUDA
10. SALIR

A continuación, en el siguiente diagrama de flujo se muestran los pasos que se siguen en nuestro programa para trabajar con las imágenes digitales, desde su carga hasta el filtrado, y posterior guardado de parámetros:


CARGA DE IMAGEN
¿RUIDOSA?
No
APLICAR RUIDO
Sí
Sí
¿NUEVA IMAGEN?
No
FILTRAR IMÁGENES
AYUDA
MEDIDAS DE CALIDAD
Sí
¿GUARDAR PARÁMETROS?
GUARDADO DE PARÁMETROS
No
Sí
¿GUARDAR IMAGEN?
GUARDADO DE IMÁGENES
No
No
FINALIZAR?
Sí
SALIR

1. CARGA DE IMAGEN:

Este módulo es el que utilizaremos para comenzar nuestros análisis. Como su nombre indica, cuando pulsemos sobre él se nos abrirá una ventana donde podremos buscar la imagen con la que deseemos trabajar.

El directorio principal abierto es, normalmente, el que tengamos por defecto en Matlab o como carpeta actual en éste. Por defecto, los archivos a encontrar serán de tipo JPG, BMP y GIF, ya que son los tipos de imagen más utilizados y más estables para trabajar con nuestros algoritmos.

Una vez seleccionada la imagen, se nos preguntará si la nueva ésta viene ya con ruido o no.

Indicaremos al programa el tipo de imagen utilizado: imagen en color (COLOR), en grises (GRISES) o en blanco y negro (BYN). Por defecto, se toma el tipo de imagen como COLOR. La función de éste módulo secundario es la clasificación de parámetros para posteriores consultas, que podemos realizar al terminar el análisis de nuestra imagen y que está explicada más adelante.

1. Pulsaremos en el botón IMAGEN para cargar una nueva imagen.

Figura 20. Menú Cargar Imagen

2. A continuación se abrirá una ventana y buscaremos el archivo deseado.

Figura 21. Ventana Abrir Imagen

3. El programa nos preguntará si la imagen original ya viene con ruido o no. (Por defecto se ha seleccionado NO)

Figura 22. Indicamos si la imagen es ruidosa o no

4. Una vez hayamos cargado nuestra imagen, Indicaremos al programa el tipo de color de ésta.

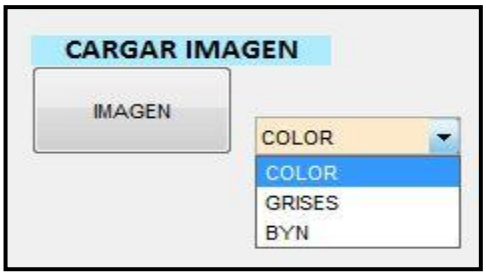


Figura 23. Tipo color imagen

2. VISUALIZACIÓN DE IMÁGENES

Para visualizar las imágenes y percibir los cambios, tenemos un módulo con dos ventanas: IMG_ORIGINAL e IMG_ RECUPERADA.

La ventana IMG_ORIGINAL nos permitirá ver la imagen que cargamos para trabajar. Cuando ésta sufra alguna modificación al aplicarle un degradado, el nombre pasará a ser IMG_ORIG_RUIDOSA, indicando así que hemos comenzado a trabajar con ella y que lo que visualizamos ya no es la imagen original.

La segunda ventana, IMG_RECUPERADA, se activará y mostrará nuestra imagen recuperada cuando se utilice cualquiera de los filtros de recuperación del módulo correspondiente.

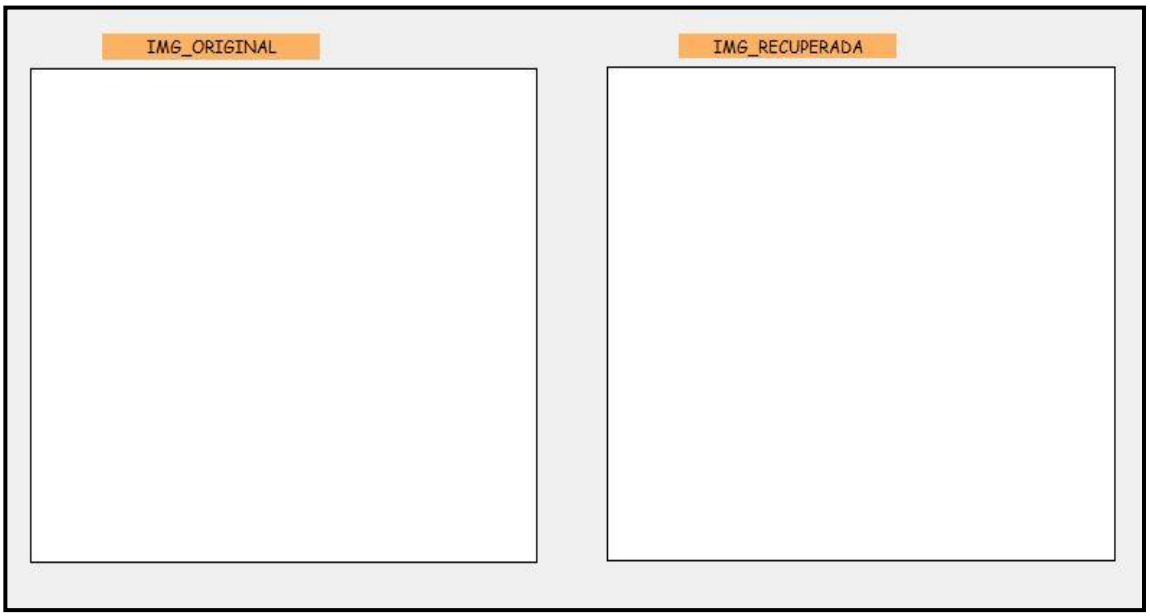


Figura 24. Panel de imágenes

El título de cada ventana indicará de qué imagen se trata: ORIGINAL, ORIGINAL RUIDOSA O RECUPERADA.

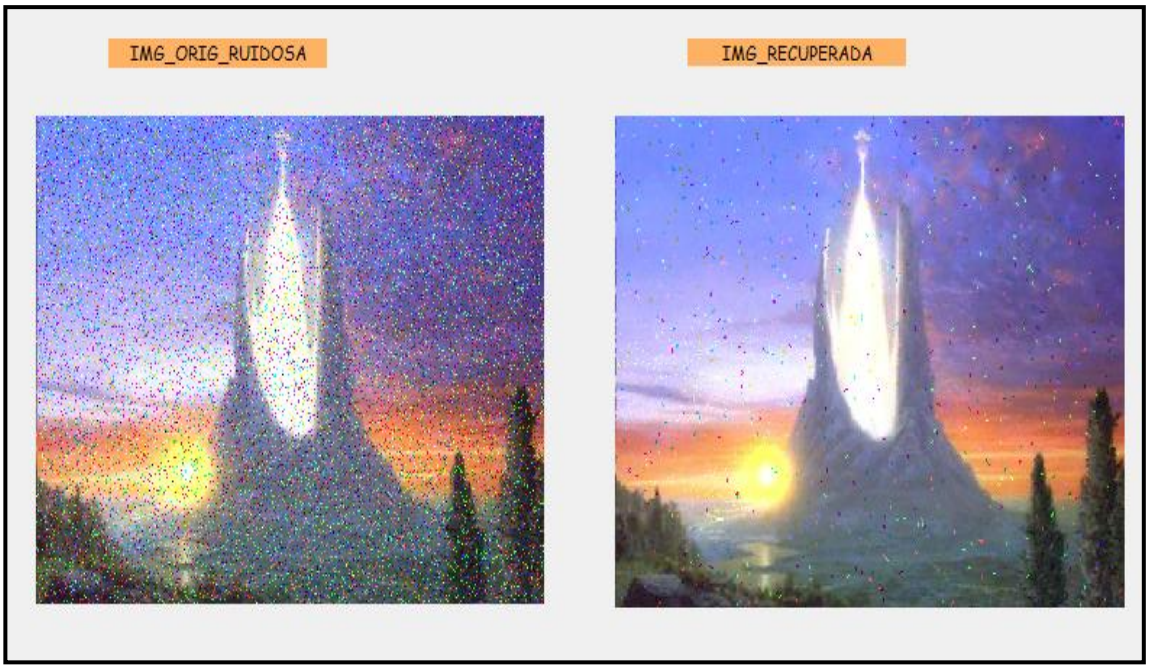


Figura 25. Imágenes RUIDOSA y FILTRADA

3. APLICACIÓN DE RUIDO

Este módulo se utilizará para aplicar ruido a la imagen cuando ésta no lo lleva originalmente.

Se podrá aplicar ruido Impulsivo, ruido Gaussiano, ambos, o ruido uniforme. Si seleccionamos cualquiera de las tres primeras opciones, podremos indicar el porcentaje de ruido (ej.: 5% daremos un valor de 0.05) en cada casillero que se activará según corresponda.

1. Seleccionamos el tipo de ruido:

Figura 26. Panel de ruido

2. Introducimos el valor de la densidad del ruido.

Figura 27. Introducir densidad de ruido

3. Aplicamos ruido pulsando el botón correspondiente.

4. FILTRADO DE IMÁGENES

En el módulo presente disponemos de 4 tipos de filtrado: filtraremos con PGFM para ruido Impulsivo, filtro Difusión para ruido Gaussiano, filtrado para ruido Impulsivo y Gaussiano, y filtro de tipo Coseno.

Para el filtro PGFM dispondremos de un módulo secundario con 3 casillas correspondientes a los 3 parámetros que podremos utilizar al aplicar el filtrado: D, N y K.

La cifra que introduzcamos podrá ser aplicada o no en función de la imagen con la que trabajemos [6]. Los valores que aparecen por defecto son D=0.95, N=3 y K=1024, como hemos comentado anteriormente en el filtrado PGFM.

Más adelante, en el módulo correspondiente, tenemos la opción de guardar los parámetros utilizados para posibles referencias futuras.

1. Seleccionamos el tipo de filtro (PGFM, Difusión, PGFM & Difusión, Coseno):

Figura 28. Escoger filtrado

2. Si el filtro seleccionado es para ruido impulsivo, indicaremos los parámetros necesarios.

Figura 29. Parámetros para filtro PGFM

3. Aplicaremos el filtrado con el botón correspondiente.

5. CÁLCULO DEL ERROR

Este módulo muestra automáticamente los resultados obtenidos al aplicar cualquiera de los filtros. Realizará una comparación de la imagen original con la imagen recuperada y mostrará los valores de PSNR, MSE, MAE y NCD

En el módulo correspondiente tenemos la opción de guardar los parámetros utilizados para posibles referencias futuras.

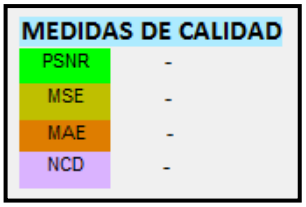


Figura 30. Panel medidas de calidad

1. Una vez filtrada la imagen, aparecen los resultados de las medidas de calidad. Como explicamos en el punto del módulo 8, tendremos la opción de guardarlos en una tabla Excel.

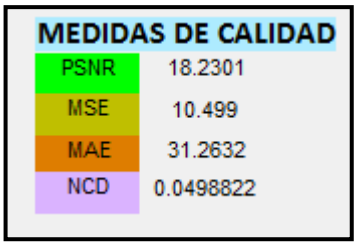


Figura 31. Ejemplo resultado medidas de calidad

6. GUARDADO DE IMÁGENES

Tenemos la opción de guardar las imágenes con las que trabajamos: imagen ruidosa y la imagen que recuperamos.

Se abrirá una ventana en la que seleccionamos el directorio o carpeta dónde deseamos guardarla para posteriores consultas y/o comparaciones.

Figura 32. Salvar imágenes

1. Escogemos qué ventana guardar.

Figura 33. Imagen a guardar

2. A continuación se abrirá una ventana y guardaremos la imagen escogida en el directorio que queramos.

Figura 34. Ventana de guardar imagen

7. PARÁMETROS UTILIZADOS

Como su nombre indica, muestra los valores que han sido utilizados por última vez al aplicar el filtrado PGFM para ruido Impulsivo. Este módulo es útil para no confundir con los valores que se estén modificando en el módulo de filtrado y que aún no hayan sido aplicados.

1. En este módulo del programa se muestran los parámetros con los que se ha filtrado la imagen (si se trata de PGFM).

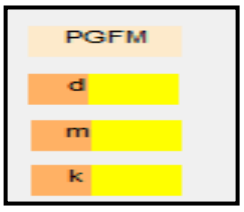


Figura 35. Muestra los últimos parámetros usados en filtro PGFM

8. GUARDADO DE PARÁMETROS

Crearemos un archivo de tablas en el que guardará la siguiente información para posterior consulta, pudiendo conocer así qué parámetros se ajustarán mejor a cada imagen:

- Tipo de color

- Nombre de la imagen
- Porcentajes de ruido aplicado
- Parámetros de filtrado PGFM
- Resultados cálculo error (PSNR, MSE, MAE y NCD)

1. Pulsamos en el botón SALVAR

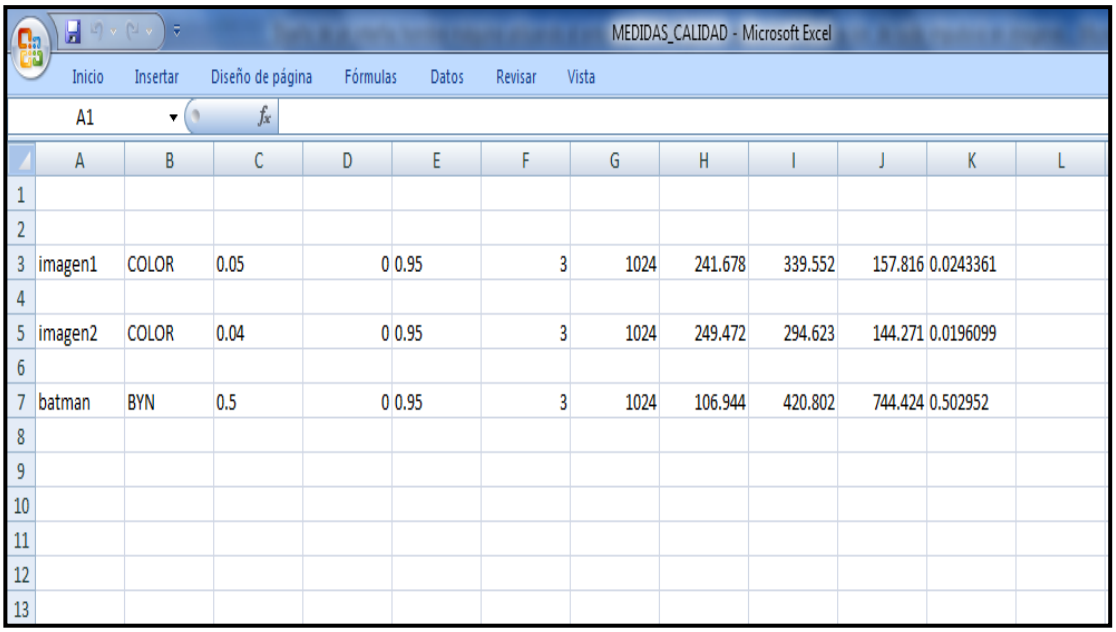
Figura 36. Botón Guardar datos

2. Se abrirá una ventana dónde introduciremos el nombre de referencia (entre 1 y 7 caracteres).

Figura 37. Introducir nombre para clasificación

3. Se habrá creado un archivo llamado *MEDIDAS_CALIDAD.txt*

(Para poder trabajar con él en Microsoft Office Excel se explica cómo abrirlo en el ANEXO)



	A	B	C	D	E	F	G	H	I	J	K	L
1												
2												
3	imagen1	COLOR	0.05	0	0.95	3	1024	241.678	339.552	157.816	0.0243361	
4												
5	imagen2	COLOR	0.04	0	0.95	3	1024	249.472	294.623	144.271	0.0196099	
6												
7	batman	BYN	0.5	0	0.95	3	1024	106.944	420.802	744.424	0.502952	
8												
9												
10												
11												
12												
13												

Figura 38. Ejemplo archivo Medidas

9. AYUDA

Mientras trabajamos con nuestro programa podemos abrir una ventana dónde se resumen los pasos a seguir. Para más ayuda se consultará la guía adjunta en documento pdf.

Figura 39. Botón Ayuda

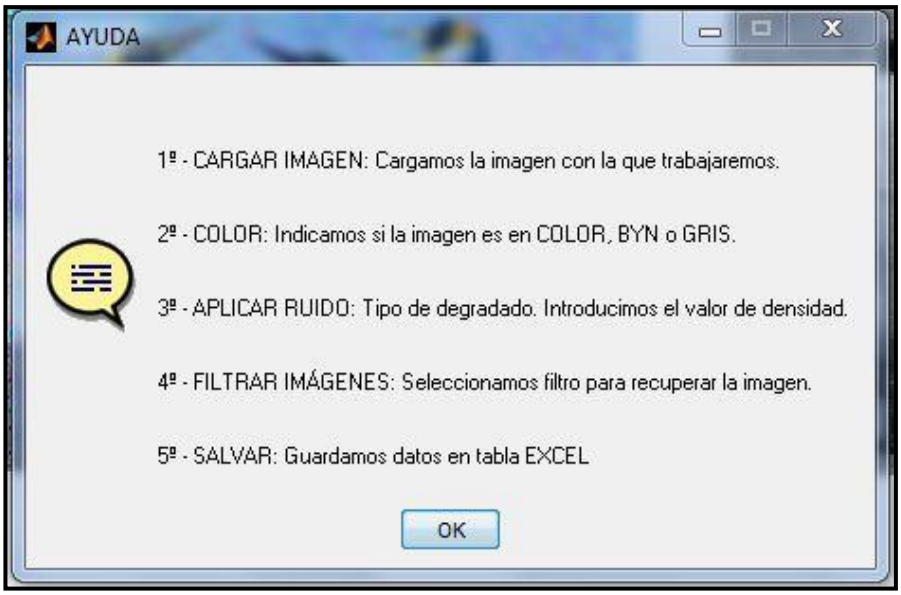


Figura 40. Mini-panel de ayuda

10. SALIR

Cuando terminemos de utilizar el programa, pulsaremos en el botón SALIR.

Figura 41. Botón Salir del programa

Figura 42. Salir Sí/No

Como ya se ha comentado antes, para la elaboración del proyecto y desarrollo de GUI en Matlab se ha consultado los manuales [1], [2] y [3].

CONCLUSIONES Y LÍNEAS FUTURAS

En este proyecto se ha presentado el desarrollo de una interfaz programada mediante Matlab™ (R2010b, 7.11), para la manipulación de imágenes, pudiendo añadir ruido para degradarlas, trabajar con los filtros y posteriormente recuperarlas.

Se han implantado tres filtros: el filtro PGFM, el filtro Difusión y el filtro Coseno. Con el GUI desarrollado podemos testear la efectividad de los filtros y así estudiar los parámetros óptimos para cada tipo de imagen.

Como el filtro de ruido Gaussiano está hecho para imágenes en grises (1-D) dará como resultado una imagen en grises y no la recuperará en color.

Dicha aplicación se ha testeado utilizando principalmente imágenes con formato BMP, JPEG y GIF, pero pueden utilizarse otros formatos que Matlab y/o que los algoritmos implementados soporten.

Respecto al guardado en Excel de parámetros de filtrado de ruido y a las medidas de calidad, se ha tenido que guardar primero en formato de texto y luego seguir los pasos ya descritos para importar a tablas. Esto se debe a que Matlab no está preparado completamente para trabajar completamente con Office ya que no permite una configuración óptima para trabajar con este programa.

El diseño de la GUI se ha llevado a cabo de manera que sea agradable y sencilla para el usuario final. Matlab no da mucha opción a grandes retoques y estilismos, ya sea en fondos de menú o botones. Si quisiéramos mejorar el aspecto de nuestro programa, se recomienda editar previamente las imágenes que correspondan al fondo de la interfaz mediante programas especializados, como Photoshop o Gimp. Considero que tampoco es necesario invertir mucho tiempo en un diseño estilizado; la importancia de nuestro objetivo reside en realizar las pruebas con los diferentes filtrados y parámetros.

Como líneas futuras, cabe decir que nuestro programa queda abierto a nuevas actualizaciones e implantaciones de los algoritmos de ruido y de filtrado.

En futuras actualizaciones de este GUI se espera actualizar el filtro difusivo (para ruido Gaussiano) para poder trabajar en imágenes a color, no sólo en grises.

Esperamos también poder trabajar directamente con Microsoft Office Excel desde Matlab™, si las nuevas actualizaciones lo permiten, ya que en la versión utilizada (7.11.0) aún dispone de pocas opciones para ello.

Por último, la idea de desarrollar esta aplicación fue la de facilitar al grupo de investigación el análisis heurístico del comportamiento de los diferentes parámetros que intervienen en los filtros descritos, objetivo que se ha logrado.

BIBLIOGRAFÍA

[1] MARCHAND PATRICK y HOLLAND, O.THOMAS (2003). *Graphics and GUIs with MATLAB*. Boca Raton: Chapman & Hall/CRC.

[2] BIRAN, ADRIAN B. y BREINER, M. (2011). *What every engineer should know about MATLAB and Simulink.* Boca Raton, Fla; London: CRC Press.

[3] SCOTT, T.SMITH (2006). MATLAB *Advanced GUI Development.* Indianapolis: Dog Ear Publishing.

[4] SÁNCHEZ, M.G., VIDAL GIMENO, V., VERDÚ, G., MAYO, P., RÓDENAS, F. y GINESTAR, D. (2011) *Un Método Híbrido de Restauración de Imágenes Médicas con Ruido Gausiano e Impulsivo.* Los Ángeles, California, USA: EMBC 2012.

[5] SÁNCHEZ, M.G., BATALLER MASCARELL, J., VIDAL GIMENO, V. y ARNAL, J. (2010). *Estudio sobre la utilización de CUDA para programas de corrección de ruido impulsivo en imágenes.* Valencia, España: Proceedings de CEDI 2010.

[6] SÁNCHEZ, M.G., VIDAL GIMENO, V., BATALLER MASCARELL, J. y RIVERA, A. (2011). *Reducción de ruido impulsivo Fijo y Uniforme en imágenes digitales usando las GPUs*. La Laguna, Tenerife, España: Proceedings de Jornadas de Paralelismo 2011.

[7] SAHOO, A. & PRATAP, M. (2009). Fuzzy Weighted Adaptive Linear Filter for Color Image Restoration Using Morphological Detectors, *International Journal on Computer Science and Engineering* Vol.1(3), 2009, 217-221.

[8] ESTRUCH CAMARENA, J.G. (2009). *Aplicació de mètriques fuzzy en la millora computacional d'algorismes de filtratge d'imatges en color.* Valencia: UPV – Tesis Doctoral RIUNET

[9] ESTRUCH FUSTER, V.D., GREGORI GREGORI, V., SAPENA PIERA, A. (2003). *Estadística*. Valencia: EDITORIAL UPV

[10] PLATANIOTIS, K.N y VENETSANOPOULOS, A.N (2000). *Color image processing and application.* Barcelona: Springer

[11] NIXON, M. y AGUADO, A. (2008). *Feature Extraction & Image Processing.* Oxford: Academic Press, Elsevier

[12] PETROU, M. y BOSDOGIANNI, P. (1999). *Image Processing: the fundamentals*. England: Wiley Editorial

ANEXO

GUÍA PARA ABRIR EL ARCHIVO *MEDIDAS_CALIDAD.txt*

El archivo generado por nuestro programa para guardar los parámetros y las medidas de calidad se genera en formato texto (.txt). Pero si queremos trabajar correctamente con él mediante tablas en Microsoft® Office Excel® (en adelante Excel) debemos seguir los siguientes pasos (donde la acción la resaltaremos en color naranja en cada ilustración):

1. Abriremos Excel desde cualquiera de sus accesos:

Figura 43. Icono de Excel® 2007

Desde el menú de usuario o Inicio en Windows 7.

Figura 44. Menú Inicio Windows 7

Figura 45. Intro del programa

2. Una vez abierto el programa, iremos a la opción de ABRIR archivo:

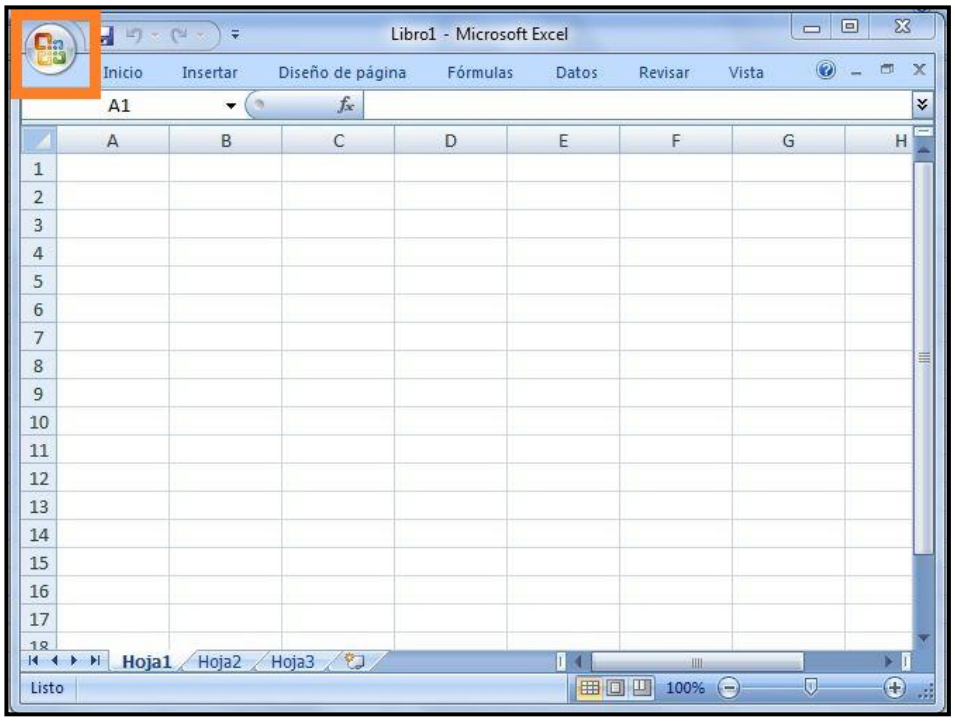


Figura 46. Botón de Office en el menú principal

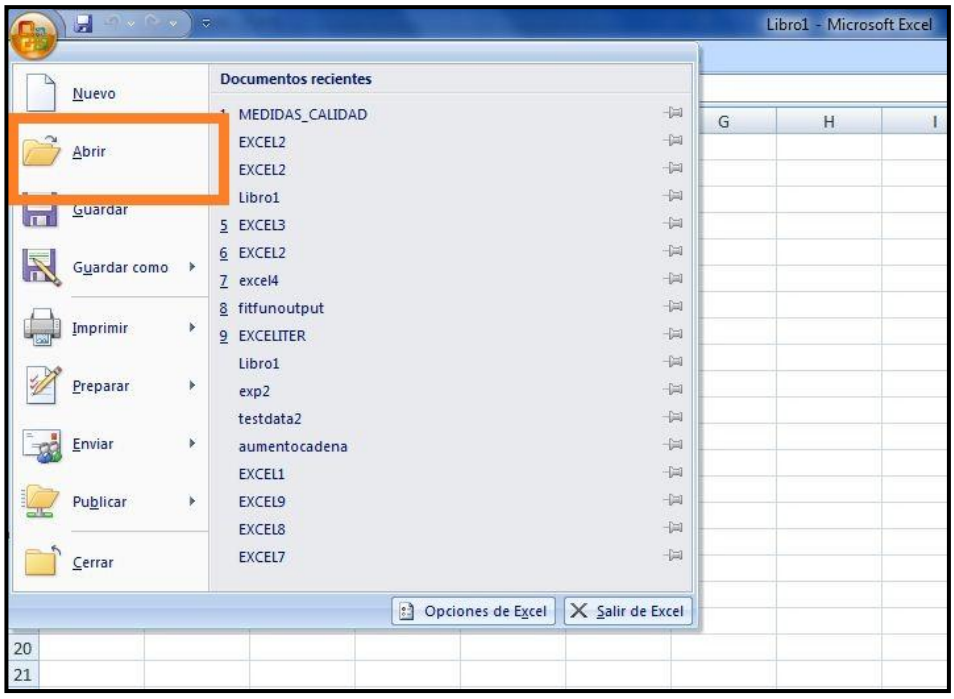


Figura 47. Abrir documento

3. Seleccionaremos "Archivos de texto" como tipo de archivo, y escogeremos MEDIDAS_CALIDAD.txt.

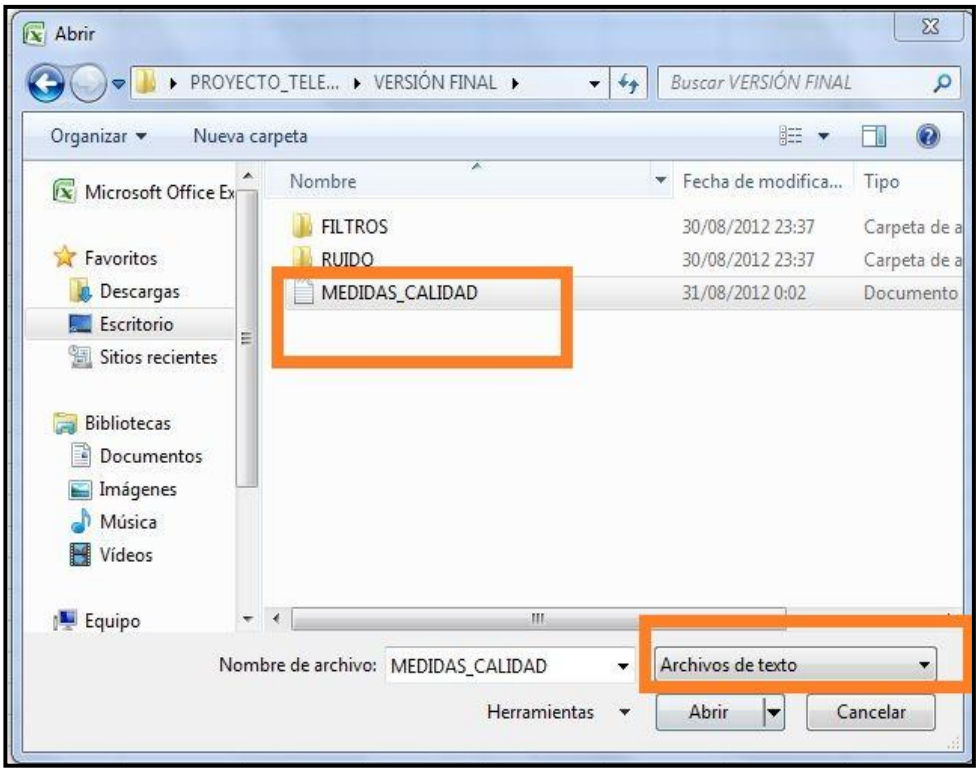


Figura 48. Buscar archivos de texto

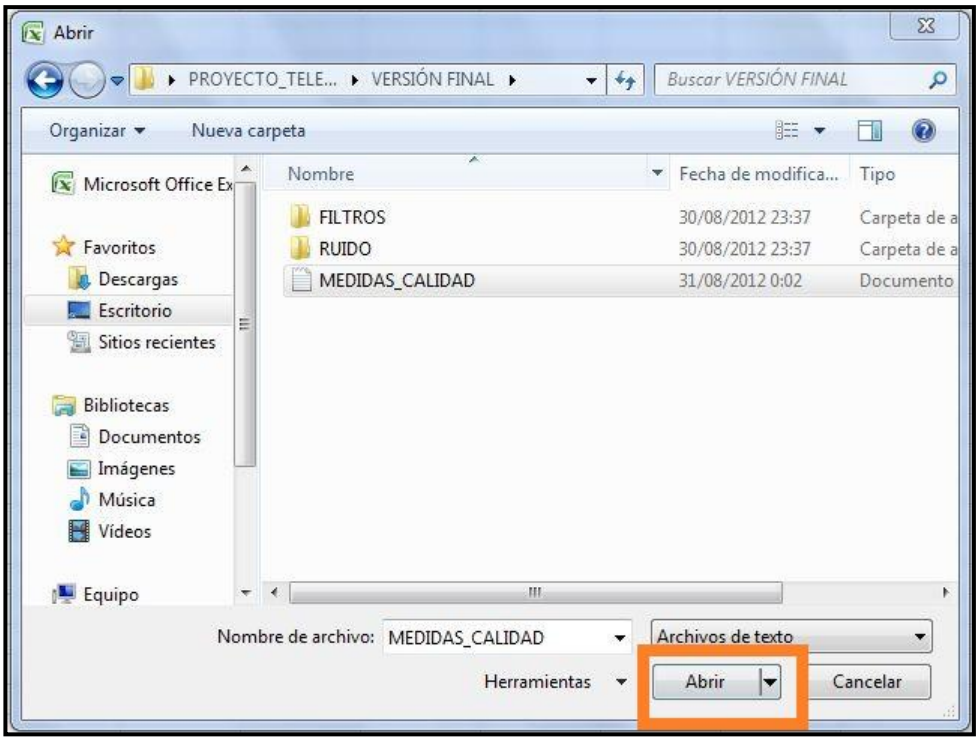


Figura 49. Abrir archivo

4. Se abrirá el asistente para importar texto.

Seleccionaremos la precisión: *Delimitados*

Y pulsaremos Siguiente.

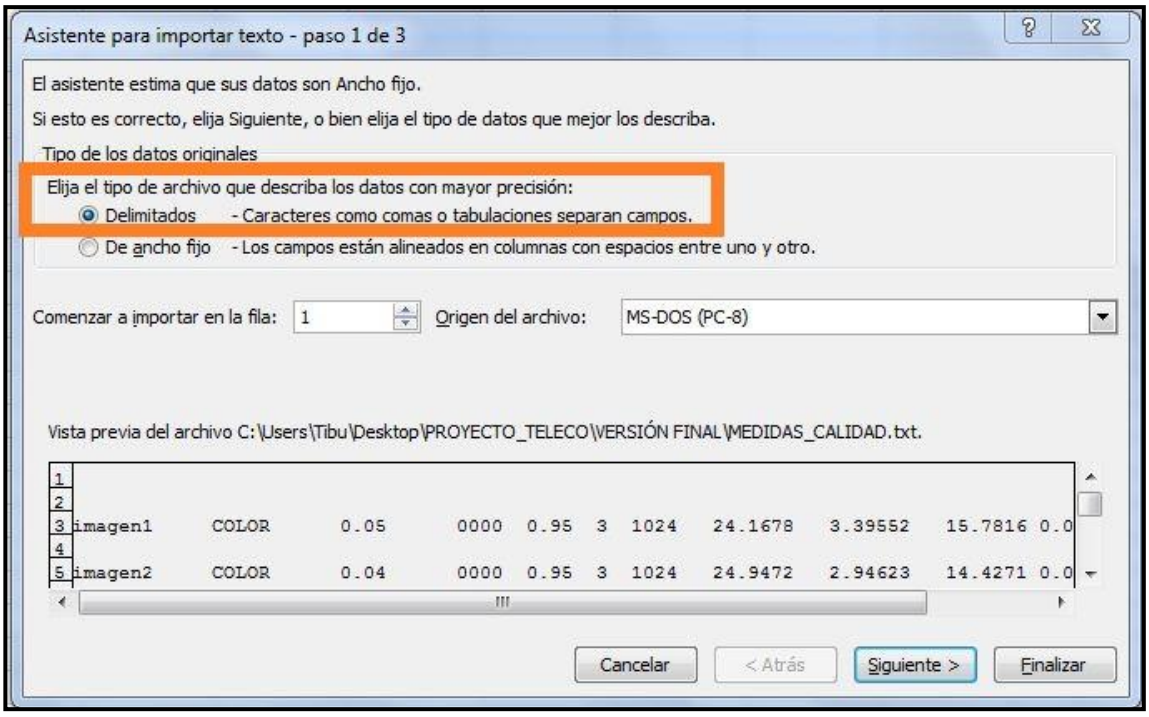


Figura 50. Paso 1 - Importar texto como tabla Excel

5. Seleccionaremos TABULACIÓN y ESPACIO en los separadores, tal y como se muestra en la figura.

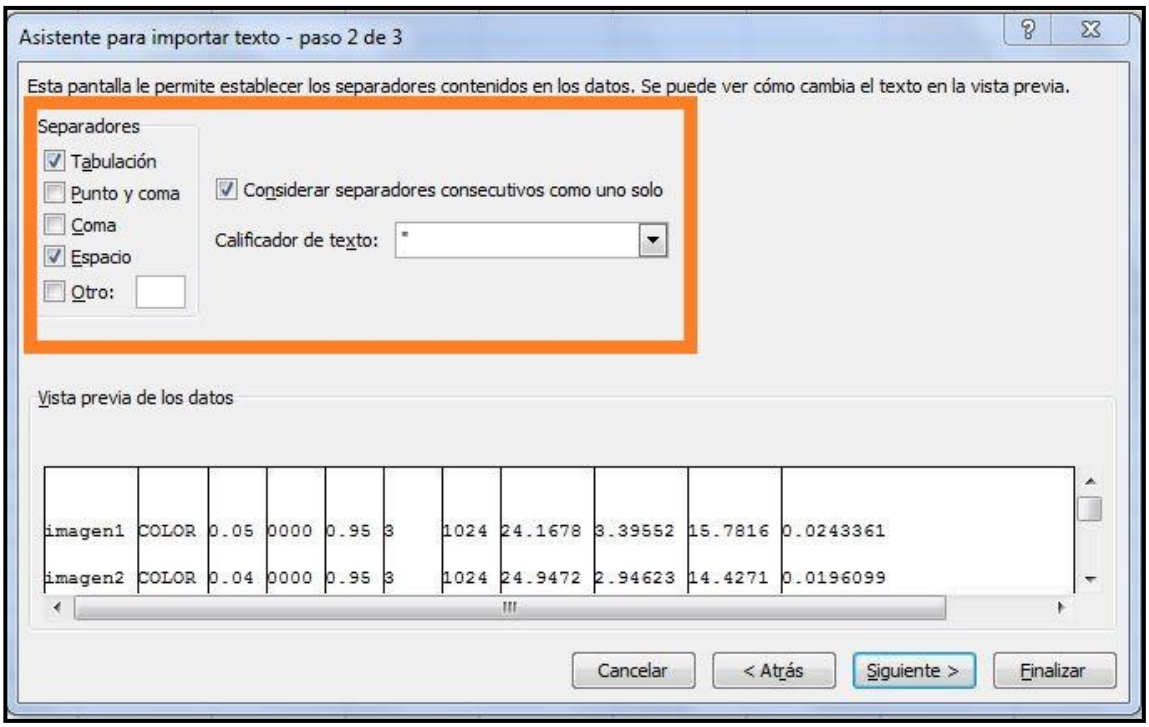


Figura 51. Paso 2 - los separadores

6. Seleccionamos formato GENERAL de los datos en columnas.

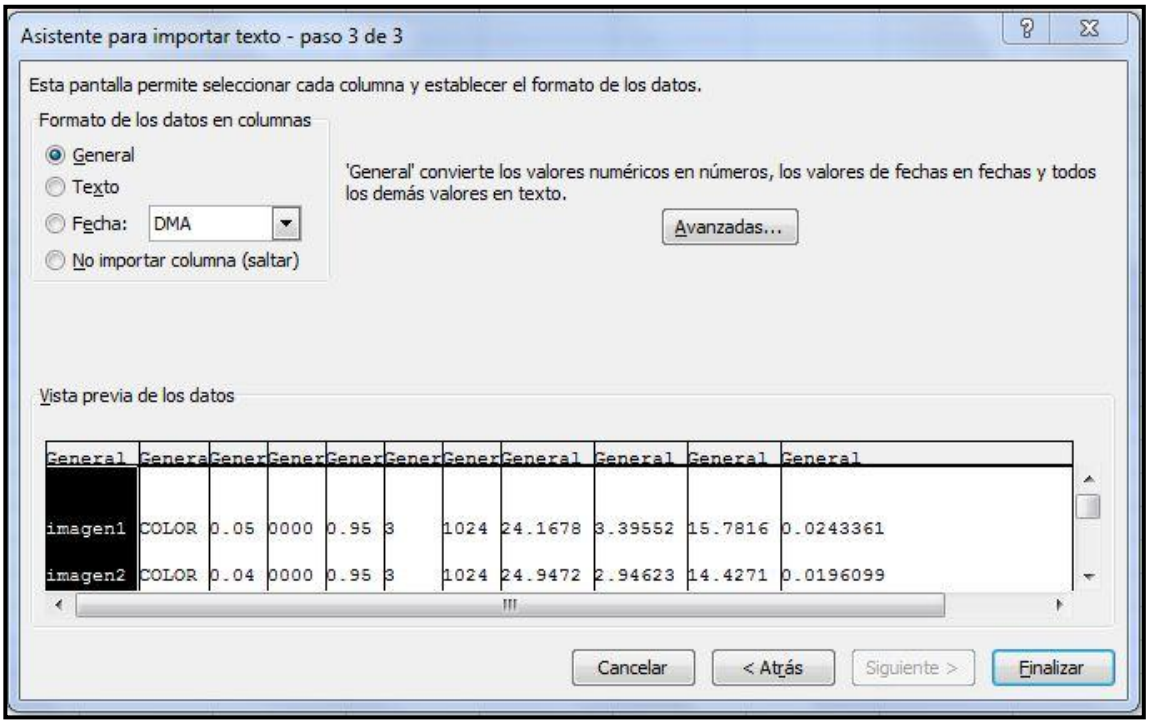


Figura 52. Paso 3 - Tipo de datos y valores

7. En “Avanzadas…” seleccionamos:

Separador decimal: “punto” (.)

Separador de miles: “en blanco” ( )

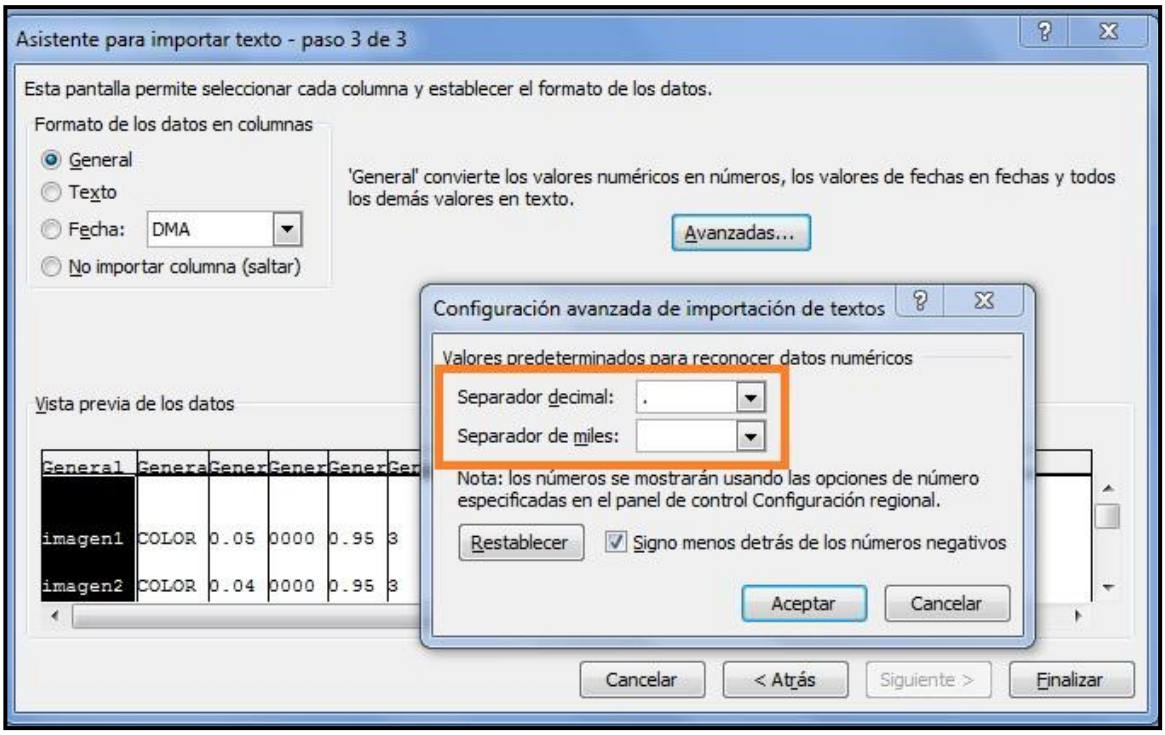
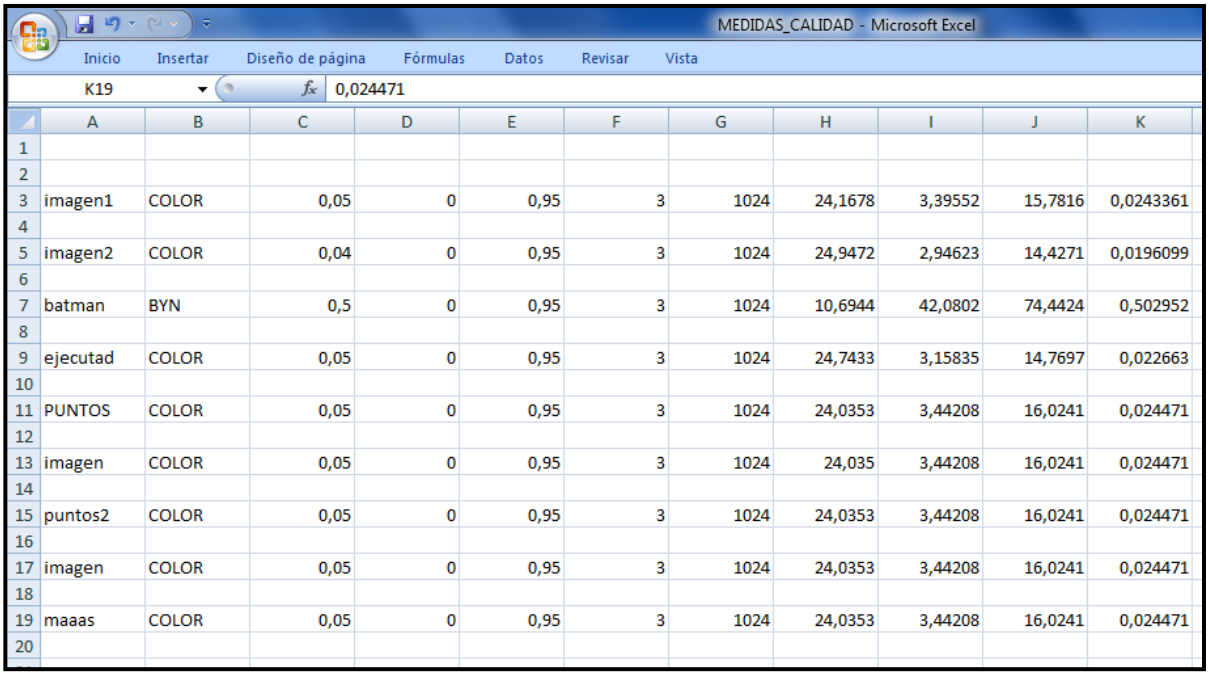


Figura 53. Reconocimiento de datos numéricos

8. Pulsamos FINALIZAR para terminar el asistente.



	A	B	C	D	E	F	G	H	I	J	K
1											
2											
3	imagen1	COLOR	0,05	0	0,95	3	1024	24,1678	3,39552	15,7816	0,0243361
4											
5	imagen2	COLOR	0,04	0	0,95	3	1024	24,9472	2,94623	14,4271	0,0196099
6											
7	batman	BYN	0,5	0	0,95	3	1024	10,6944	42,0802	74,4424	0,502952
8											
9	ejecutad	COLOR	0,05	0	0,95	3	1024	24,7433	3,15835	14,7697	0,022663
10											
11	PUNTOS	COLOR	0,05	0	0,95	3	1024	24,0353	3,44208	16,0241	0,024471
12											
13	imagen	COLOR	0,05	0	0,95	3	1024	24,035	3,44208	16,0241	0,024471
14											
15	puntos2	COLOR	0,05	0	0,95	3	1024	24,0353	3,44208	16,0241	0,024471
16											
17	imagen	COLOR	0,05	0	0,95	3	1024	24,0353	3,44208	16,0241	0,024471
18											
19	maaas	COLOR	0,05	0	0,95	3	1024	24,0353	3,44208	16,0241	0,024471
20											

Figura 54. Ejemplo tabla con las medidas de calidad y parámetros

Ahora ya podemos trabajar con nuestros datos en formato EXCEL.

Las columnas representan los siguientes datos:

- A: Tipo de color
- B: Nombre de la imagen
- C: Porcentajes de ruido Impulsivo aplicado
- D: Porcentajes de ruido Gaussiano aplicado
- E, F, G: Parámetros de filtrado Impulsivo
- H: Resultados PSNR
- I: Resultados MSE
- J: Resultados MAE
- K: Resultados NCD

FUNCIONES PRINCIPALES

INTRODUCIR RUIDO

Ruido Impulsivo:

Z = imnoise(imagen,'salt & pepper',edit3);

Función *Imnoise* para ruido Salt & Pepper, donde *imagen* es la imagen original y *edit3* es el valor de la densidad de ruido.

Ruido Gaussiano:

Z=imnoise(imagen,'gaussian',edit4,edit10)

Función *Imnoise* para ruido Gaussian, donde *imagen* es la imagen original y *edit3 y edit4* son los valores de la media y de la varianza, respectivamente.

Ruido Uniforme:

Z = AddUniformNoise(imagen);

Función *AddUniformNoise* para ruido uniforme, donde *imagen* es la imagen original.

FILTRADO DE IMÁGENES

Filtro PGFM:

Z1 = impulsivofilter(Z,param_d,param_n,param_k);

Función *impulsivofilter* para el filtro PGFM, donde *Z* es la imagen a filtrar y *param_d, param_n y param_k* son los parámetros de configuración del filtro.

Filtro Difusión:

Z1 = gaussfilter(Z);

Función *gaussfilter* para el filtro PGFM, donde *Z* es la imagen ruidosa.

Filtro Coseno:

Z1 = cosenofilter(Z,umbral1,umbral2);

Función *cosenofilter* para el filtro PGFM, donde *Z* es la imagen a filtrar y el resto de variables son los umbrales mínimo y máximo para etiquetar píxeles como ruidosos.

MEDIDAS DE CALIDAD

PSNR:

M_PSNR = PSNRImages(imagen1,imagen2);

MSE:

M_MSE = MSEImages (imagen1,imagen2);

MAE:

M_MAE = MAEImages(imagen1,imagen2);

NCD:

M_NCD = NCDImages (imagen1,imagen2);

Donde imagen1 e imagen2 son las imágenes original y filtrada, respectivamente.