ESCUELA SUPERIOR POLITECNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"RESTAURACION DE IMAGENES EN BLANCO Y NEGRO A TRAVES DE LA DETECCION Y MEJORAMIENTO DE SUS BORDES"

INFORME DE MATERIA DE GRADUACION

Previo a la obtención del título de:

INGENIERO EN ELECTRONICA Y TELECOMUNICACIONES

Presentado por:

Germán Andrés López Peña

Guayaquil – Ecuador

2009

DEDICATORIA

A mi familia,

Amigos y todos

Los que de una u otra forma,

Me ayudaron a culminar mi carrera,

En especial a mi prometida

Anabel

TRIBUNAL DE GRADUACION

----------	**----------**
Ing. Patricia Chávez Burbano Directora	Ing. Juan Carlos Avilés Delegado del Decano

DECLARACION EXPRESA

"La responsabilidad del contenido de este Reporte de Materia de Graduación, me corresponde exclusivamente; y el patrimonio intelectual de la misma a la Escuela Superior Politécnica del Litoral".

_____

Germán Andrés López Peña

RESUMEN

Este proyecto tiene como objetivo la restauración de imágenes en blanco y negro. Las imágenes a ser restauradas serán fotografías en blanco y negro las cuales debido a su antigüedad o al proceso de digitalización han sufrido un deterioro en la calidad de las mismas. Esto se lo realizará con una detección y mejoramiento de los bordes, para lo cual se hará uso de los filtros Laplaciano y Canny, además de los filtros wavelet Haar y Sym2.

Durante el proyecto se procesarán cuatro imágenes diferentes con cada de los filtros para determinar con cual de ellos se obtiene un mejor resultado visual. Las imágenes resultantes serán sometidas a una encuesta, a fin de poder esclarecer cual produce el mejor efecto deseado. Adicionalmente se determinará el error medio cuadrático normalizado y el tiempo de procesamiento de los mismos como parámetros adicionales para la elección del filtro adecuado, dependiendo de las limitantes del computador a utilizar.

El proyecto será desarrollado utilizando el programa Matlab, el cual tendrá implementada una interfaz gráfica a fin de facilitar la interacción del usuario con el mismo.

INDICE GENERAL

ABREVIATURAS

DWT2	Discret Wavelet Transform 2D
NMSE	Normalised Mean Square Error

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCION

Las fotos que se han tomado con cámaras convencionales y que han sido escaneadas a fin de poder preservarlas de la degradación habitual debido al medio ambiente, así como las fotos tomadas mediante métodos digitales usualmente requieren ser sometidas a un proceso de restauración a fin de mejorar la calidad de las mismas, las cuales pueden haber sido afectadas por diversos factores durante su captura o proceso de digitalización.

Existen muchas técnicas utilizadas para la restauración digital de imágenes, una técnica muy utilizada en la fotografía es el mejoramiento de los bordes de la imagen. Esta técnica consiste en aislar los bordes de la imagen, amplificarlos y posteriormente volverlos a colocar en la imagen. Existen varios filtros que se emplean para este propósito, pero en este caso utilizaremos solamente cuatro: Gradiente basado en derivada de segundo

orden (Laplaciano), Filtro Canny, Wavelet Haar y Wavelet Sym2 descritos en el capítulo #1.

Estos algoritmos fueron implementados usando Matlab 7.7.0 con el Editor incluido en el programa, así como GUIDE que es una herramienta para el desarrollo de las interfaces gráficas. Los resultados de estos procedimientos son presentados en el Capítulo 2 y los códigos correspondientes en los Apéndices A y B.

El objetivo principal de este proyecto es poner en práctica los conocimientos adquiridos durante el desarrollo de la materia, y mostrar una alternativa diferente a la gran variedad de programas disponibles para el procesamiento de imágenes que existen en la actualidad.

CAPITULO 1

ALGORITMOS DE DETECCION DE BORDES

1.1 **Filtro Laplaciano**

Este filtro se basa en crear una máscara que destaque los píxeles (a través del aumento de su nivel de gris) cuya variación, con respecto a su vecindad, es significativa.

El Laplaciano se define como el diferencial de segundo orden con respecto a dos variables. Ello es:

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} \qquad (1)$$

Donde,

$$\frac{\partial^2 f(x,y)}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y) \qquad (2)$$

$$\frac{\partial^2 f(x,y)}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y) \qquad (3)$$

Reemplazando (2) y (3) en (1) se obtiene:

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

(4)

Esto es equivalente a una máscara de la forma:

$$[h] = \begin{bmatrix} \propto/(1+\propto) & (1-\propto)/(1+\propto) & \propto/(1+\propto) \\ (1-\propto)/(1+\propto) & -4/(1+\propto) & (1-\propto)/(1+\propto) \\ \propto/(1+\propto) & (1-\propto)/(1+\propto) & \propto/(1+\propto) \end{bmatrix} \quad (5)$$

El valor de α esta limitado al rango de 0.0 a 1.0.

1.2 Filtro Canny

El filtro Canny es un filtro utilizado para la detección de bordes, el cual se basa en tres criterios fundamentales:

- Una buena detección. Debe haber una baja probabilidad de seleccionar en forma incorrecta los bordes y de marcar como bordes puntos que no lo son.
- Una buena localización. El punto seleccionado como borde debe encontrarse lo más cerca posible del centro del borde real.
- Unicidad en la obtención de bordes. Esto está implícito en el primer criterio, ya que de haber dos respuestas para un mismo borde uno debe estar equivocado. Sin embargo el aspecto matemático del primer criterio no lo considera, así que se debe hacer implícitamente.

El primer criterio se cumple con la siguiente ecuación:

$$SNR = \frac{\left|\int_{-W}^{+W} G(-x) f(x) dx\right|}{n_0 \sqrt{\int_{-W}^{+W} f^2(x) dx}} \quad (6)$$

Donde G(x) es el filtro Gaussiano que se utiliza para suavizar los bordes y n_0 es la amplitud promedio del ruido.

El segundo criterio se satisface con la siguiente ecuación:

$$Localización = \frac{\left|\int_{-W}^{+W} G'(-x)f'(x)dx\right|}{n_0\sqrt{\int_{-W}^{+W} f'^2(x)dx}} \qquad (7)$$

El tercer criterio se satisface con la siguiente ecuación:

$$Distancia = \pi\left(\frac{\int_{-\infty}^{+\infty} f'^2(x)dx}{\int_{-\infty}^{+\infty} f''^2(x)dx}\right) \qquad (8)$$

1.3 Filtro Wavelet Haar

La transformada wavelet Haar consisten en un breve impulso positivo seguido de un breve impulso negativo. La transformada Haar es una función ortogonal, lo cual junto con la existencia de una función de escalabilidad son los criterios que se asocian en los filtros wavelet con un algoritmo rápido y pequeño. La transformada Haar discreta además se beneficia de las propiedades de las transformaciones ortogonales. La inversa de una transformación ortogonal es también particularmente fácil de implementar porque esta es la transpuesta de una transformación directa.

La transformada wavelet Haar es una transformada lineal separable basada en la función escalón, g(x), y en la función Haar, h(x) expresadas como:

$$g(x) = \begin{cases} 1\ ;\ 0 \le x \le 1 \\ 0\ ;\ otro\ caso \end{cases} \qquad (9)$$

$$h(x) = \begin{cases} 1\,; 0 \le x \le \frac{1}{2} \\ -1; \frac{1}{2} \le x \le 1 \\ 0\,; otro\ caso \end{cases} \qquad (10)$$

1.4 Filtro Wavelet Sym2

Los filtros simétricos son generalmente llamados filtros de fase lineal; si un filtro no es simétrico, entonces su desviación de simetría es juzgada por como su fase se desvía de una función lineal. Para hacer un filtro "cercano" a la simetría, la idea es modificar la fase para hacerla "casi" lineal.

Teniendo:

$$P(y) = \sum_{k=0}^{N-1} C_k^{N-1+k} y^k \qquad (11)$$

Donde C_k^{N-1+k} denota a los coeficientes binomiales.

Entonces,

$$\left|m_{0(w)}\right|^2 = (cos^2(w/2))^N \mathrm{P}(sin^2(\mathrm{w}/2)) \qquad (12)$$

La idea es expresar es reconstruir el m_0, considerando $\left|m_{0(w)}\right|^2$ como una función W de $z = e^{iw}$

Entonces podemos factorizar W de diversas maneras a la forma $W(z) = U(z)\overrightarrow{U(z)}$ debido a que las raíces de W con módulos no iguales a 1 van en par. Si una de las raíces es z1, entonces 1/z1 también es una raíz.

Seleccionando U de tal forma que los módulos de sus raíces sean menor que 1, construimos Daubechies wavelets dbN. El filtro U es un "filtro de fase mínima"

Haciendo otra elección, obtenemos filtros más simétricos; estos son los filtros symlets.

CAPITULO 2

IMPLEMENTACION Y RESULTADOS EXPERIMENTALES

2.1 Implementación

Se procedió a implementar los algoritmos descritos en el capítulo 1 basados en las ecuaciones (5), (6), (7), (8), (9), (10) y (12) usando Matlab 7.7.0 con la herramienta de procesamiento de imágenes. Se desarrolló un programa interactivo que permite el cambio de parámetros para los algoritmos, las imágenes a procesar y los nombres con el que se guardarán las imágenes resultantes de estos procesos. Una vez que se hayan copiados los archivos correspondientes a la carpeta de trabajo de Matlab, se debe tipear en la ventana de comandos la palabra "Presentacion" lo cual dará inicio al programa, donde se podrá elegir el proceso de filtrado al cual se desea someter a una imagen en particular.

Para las diversas pruebas realizadas se configuraron diversos parámetros en cada uno de los algoritmos para mejorar la eficiencia de los mismos:

- Las imágenes que se ingresen para ser procesadas deberán estar en escala de grises, ya que de lo contrario se producirá un mensaje de error y el programa se interrumpirá.
- El filtro Canny, se lo implementó utilizando la función edge de Matlab, la cual permite especificar los parámetros de umbral de convergencia del filtro, pero estos valores serán omitidos a fin de que el programa escoja los valores correspondientes en forma automática.
- El filtro wavelet Haar, se lo implementó usando la función dwt2 de Matlab y puede escogerse entre tres niveles de filtrado.
- El filtro wavelet sym2, se lo implementó usando la función dwt2 de Matlab y puede escogerse entre tres niveles de filtrado.

Adicionalmente, para las pruebas realizadas se utilizaron cuatro imágenes (Fig. 2.1, 2.2, 2.3 y 2.4): dos en las que se muestran imágenes de personas y dos en las que se presentan únicamente objetos, a fin de poder observar la forma en que las imágenes se ven afectadas por cada uno de los filtros y determinar cual es el más conveniente para cada caso.

Para el caso del filtro Laplaciano se realizaron pruebas con la imagen de la figura 2.4 variando los valores de alpha, para poder determinar cual produce mejores efectos en la imagen resultante.

Como medida para determinar las variaciones entre la imagen original y la resultante se decidió utilizar el error cuadrático normalizado (NMSE).

Fig. 2.1 Imagen de prueba 1

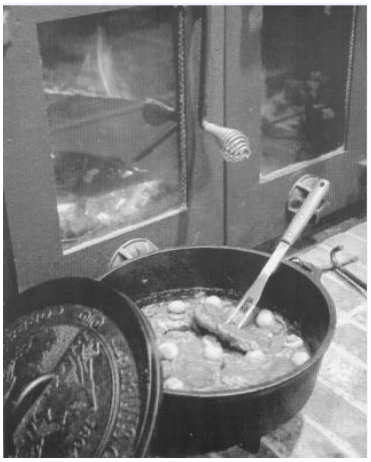

Fig. 2.2 Imagen de prueba 2

Fig. 2.3 Imagen de prueba 3

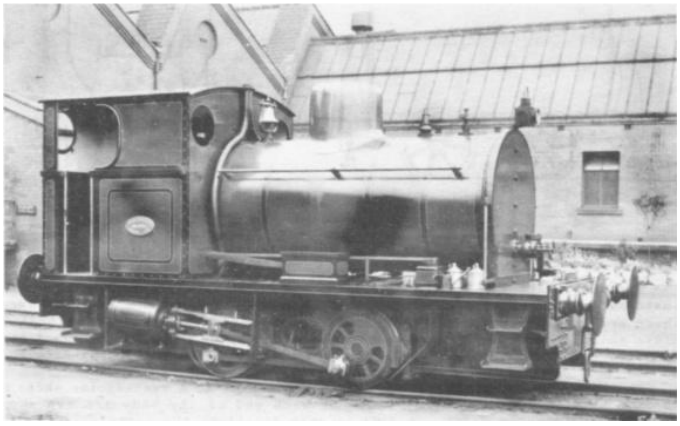

Fig. 2.4 Imagen de prueba 4

2.2 Resultados experimentales

2.2.1 Filtro Laplaciano con variación de alpha

Para este experimento se utilizó únicamente la imagen de la figura 2.4, la misma que se sometió al proceso de restauración con el filtro Laplaciano mientras, utilizando diferentes valores de alpha. En la Tabla 2.1 se muestran los resultados de NMSE para los diferentes valores de alpha, mientras que en las figuras 2.5, 2.6, y 2.7 se muestra la forma en que varía la imagen de acuerdo a estos cambios, así como los bordes detectados en cada caso.

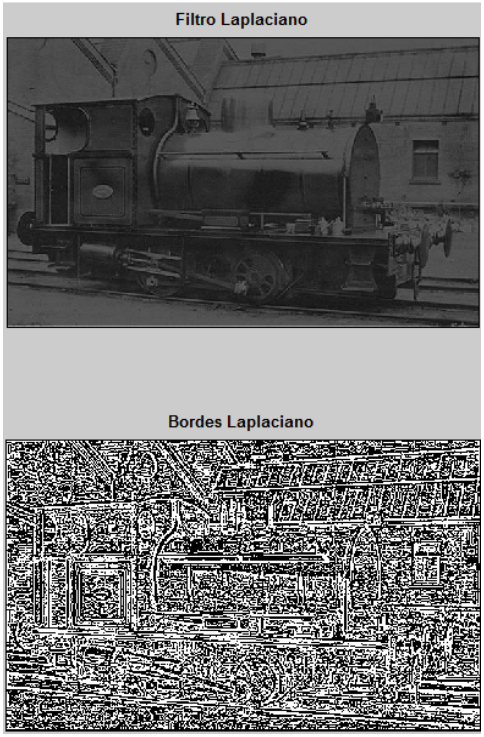

Fig. 2.5 Laplaciano con alpha=0.0

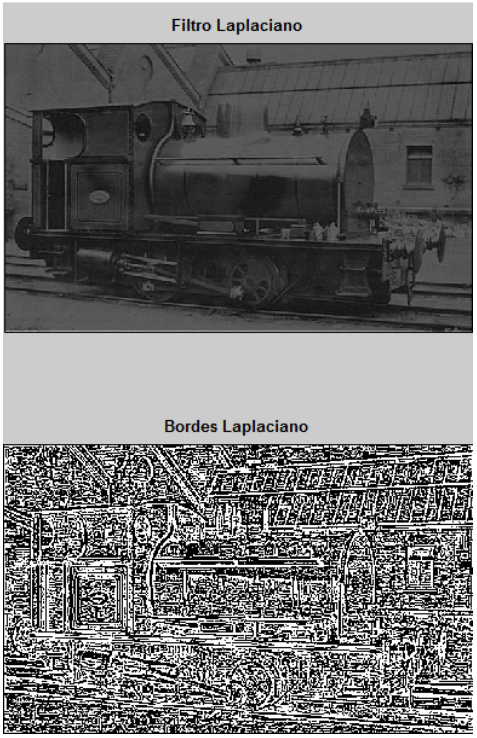

Fig. 2.6 Laplaciano con alpha=0.4

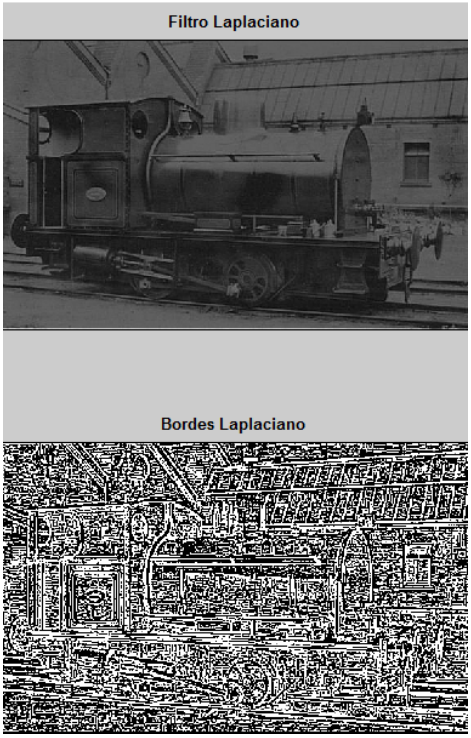

Fig. 2.7 Laplaciano con alpha=1.0

Como se puede apreciar que la imagen de la figura 2.7 presenta una imagen mas clara que las imágenes de las figuras 2.5 y 2.6, así mismo se aprecia que la cantidad de bordes detectados con un alpha=1 es menor que la cantidad de bordes obtenidos con valores de alpha diferentes a 1.

Alpha	**NMSE**
0.0	$1.979\ x\ 10^{-6}$
0.4	$1.896\ x\ 10^{-6}$
1.0	$1.768\ x\ 10^{-6}$

Tabla 2.1 Variaciones del NMSE en el filtro Laplaciano

2.2.2 Filtro Laplaciano y filtro Canny

Para este experimento se utilizaron las imágenes de las figuras 2.1, 2.2, 2.3 y 2.4 todas evaluadas con un alpha=1. En la Tabla 2.2 se muestran los resultados de NMSE obtenidos en cada caso, mientras que en las figuras 2.8, 2.9, 2.10 y 2.11 se muestran las imágenes resultantes así como los bordes detectados por ambos filtros. Adicionalmente en la Tabla 2.3 se muestra el tiempo de procesamiento de los procesos en cada imagen.

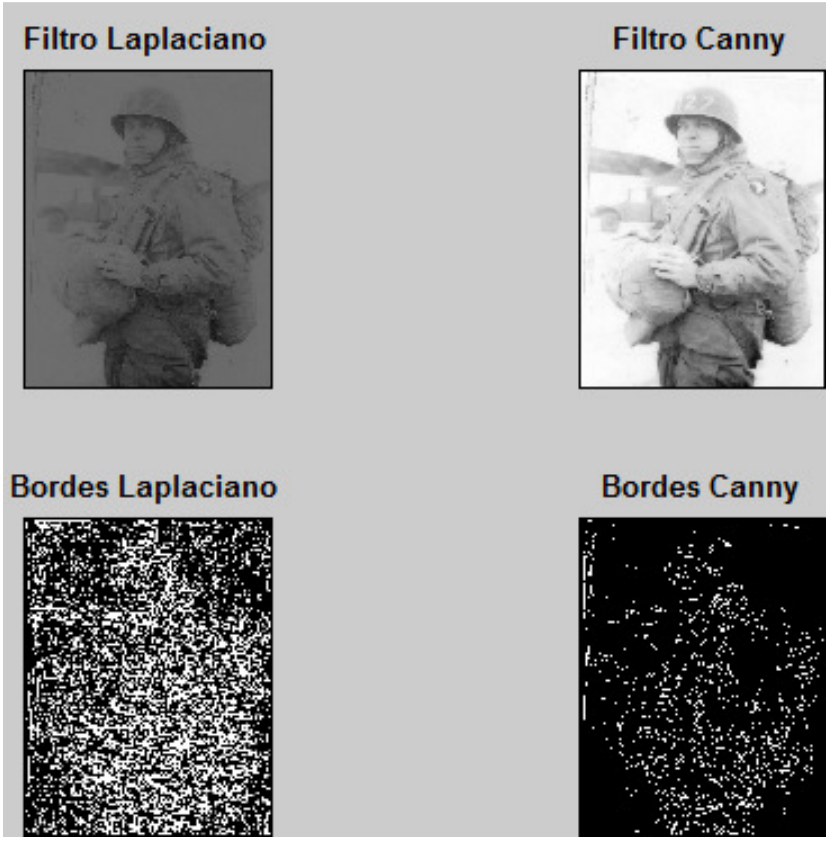

Fig. 2.8 Imagen de prueba 1 procesada con los filtros Laplaciano y Canny

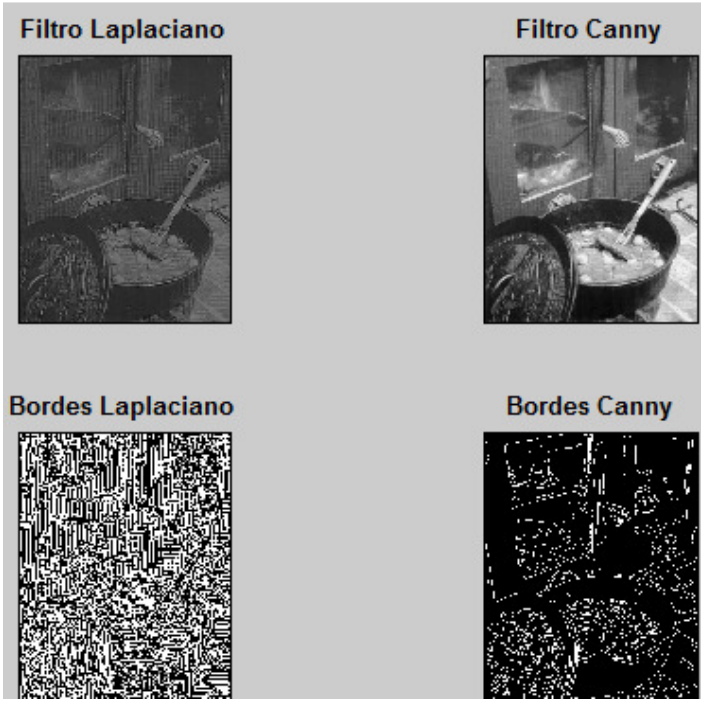

Fig. 2.9 Imagen de prueba 2 procesada con los filtros Laplaciano y Canny

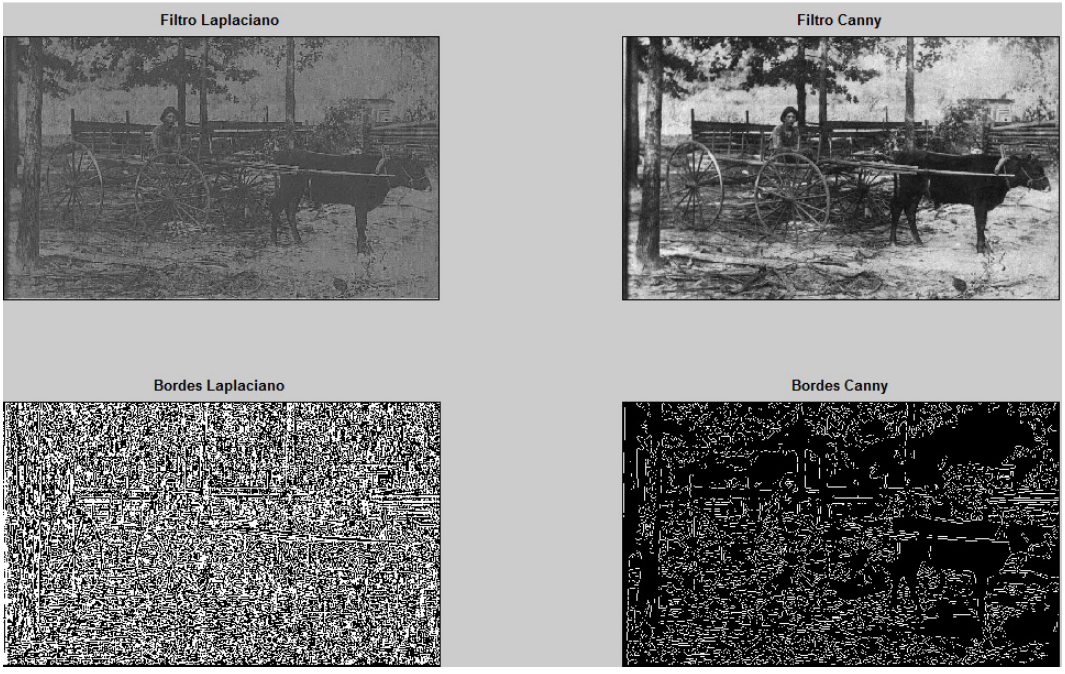

Fig. 2.10 Imagen de prueba 3 procesada con los filtros Laplaciano y Canny

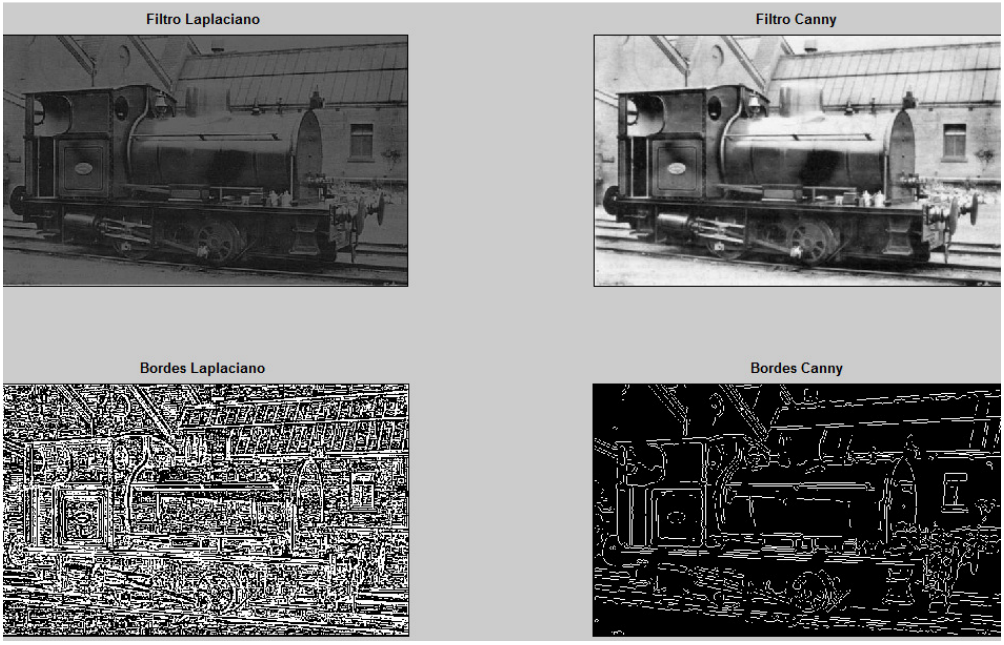

Fig. 2.11 Imagen de prueba 4 procesada con los filtros Laplaciano y Canny

	NMSE Laplaciano	**NMSE Canny**
Imagen 1	$0.586\ x\ 10^{-6}$	$1.584\ x\ 10^{-9}$
Imagen 2	$1.588\ x\ 10^{-6}$	$0.172\ x\ 10^{-6}$
Imagen 3	$0.336\ x\ 10^{-6}$	$0.102\ x\ 10^{-6}$
Imagen 4	$1.767\ x\ 10^{-6}$	$0.170\ x\ 10^{-6}$

Tabla 2.2 Comparación del NMSE del filtro Laplaciano y Canny

Como se puede apreciar en cada una de las imágenes el filtro Canny resulta ser más efectivo que el filtro Laplaciano, ya que las imágenes resultantes son más claras, además de que el NMSE producido en

cada caso que se utilizó el filtro Canny es menor que el producido por el filtro Laplaciano.

	Tiempo Laplaciano (s)	Tiempo Canny (s)
Imagen 1	0.0421	1.5712
Imagen 2	0.0142	0.3549
Imagen 3	0.0445	0.8918
Imagen 4	0.0193	0.5762

Tabla 2.3 Tiempos de procesamiento del filtro Laplaciano y Canny

De la tabla 2.3 podemos ver que el tiempo de procesamiento del filtro Laplaciano es considerablemente menor que el del filtro Canny.

2.2.3 Filtros Wavelet Haar y Wavelet Sym2

Para este experimento se utilizaron las imágenes de las figuras 2.1, 2.2, 2.3 y 2.4 todas utilizando un filtrado de primer nivel. En la Tabla 2.4 se muestran los resultados del NMSE obtenidos en cada caso, mientras que en las figuras 2.12, 2.13, 2.14 y 2.15 se muestran las imágenes resultantes así como los bordes detectados por ambos filtros.

Adicionalmente en la Tabla 2.5 se muestra el tiempo de procesamiento de los procesos en cada imagen.

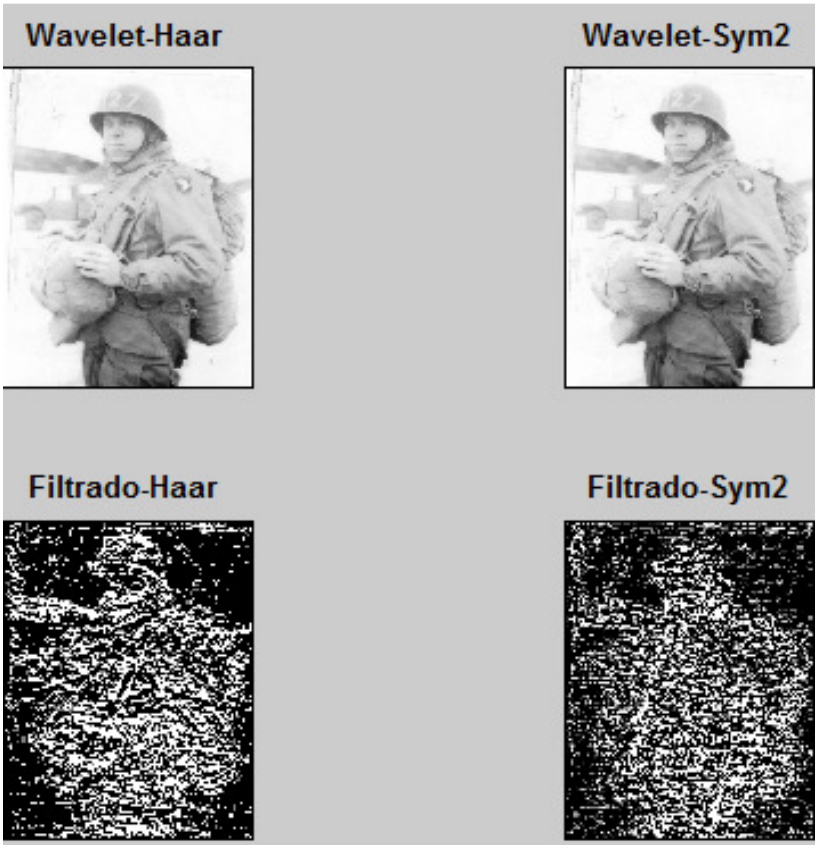

Fig. 2.12 Imagen de prueba 1 procesada con los filtros wavelet Haar y Sym2

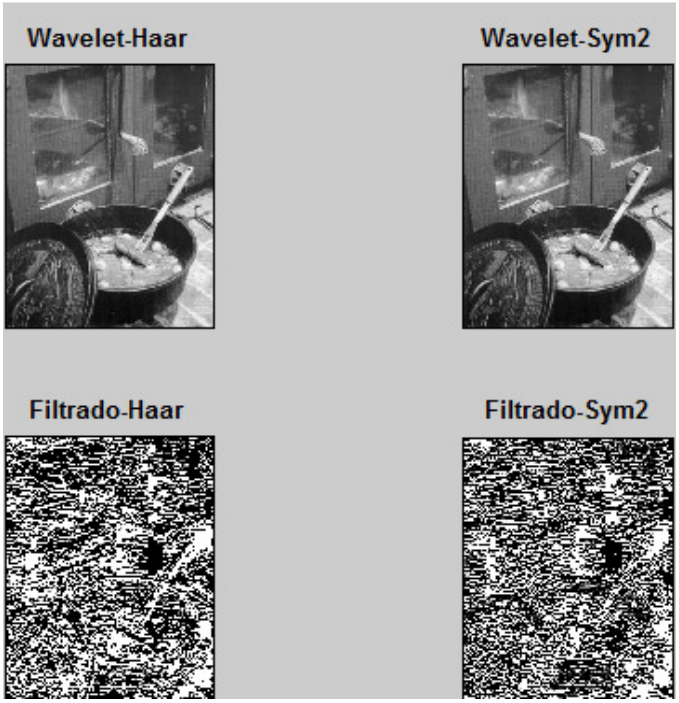

Fig. 2.13 Imagen de prueba 2 procesada con los filtros wavelet Haar y Sym2

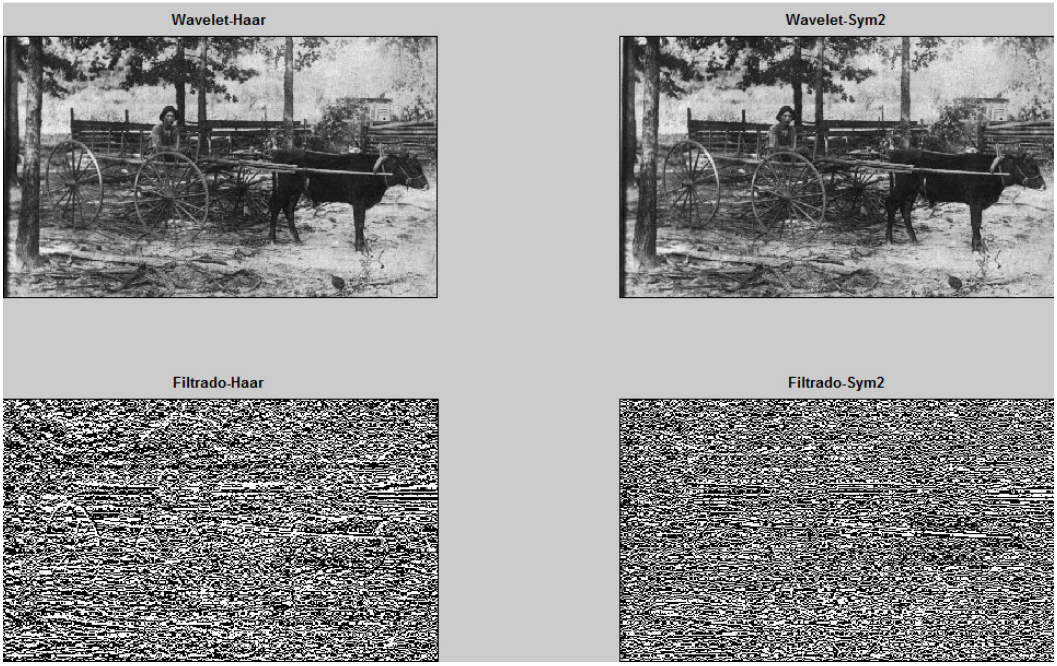

Fig. 2.14 Imagen de prueba 3 procesada con los filtros wavelet Haar y Sym2

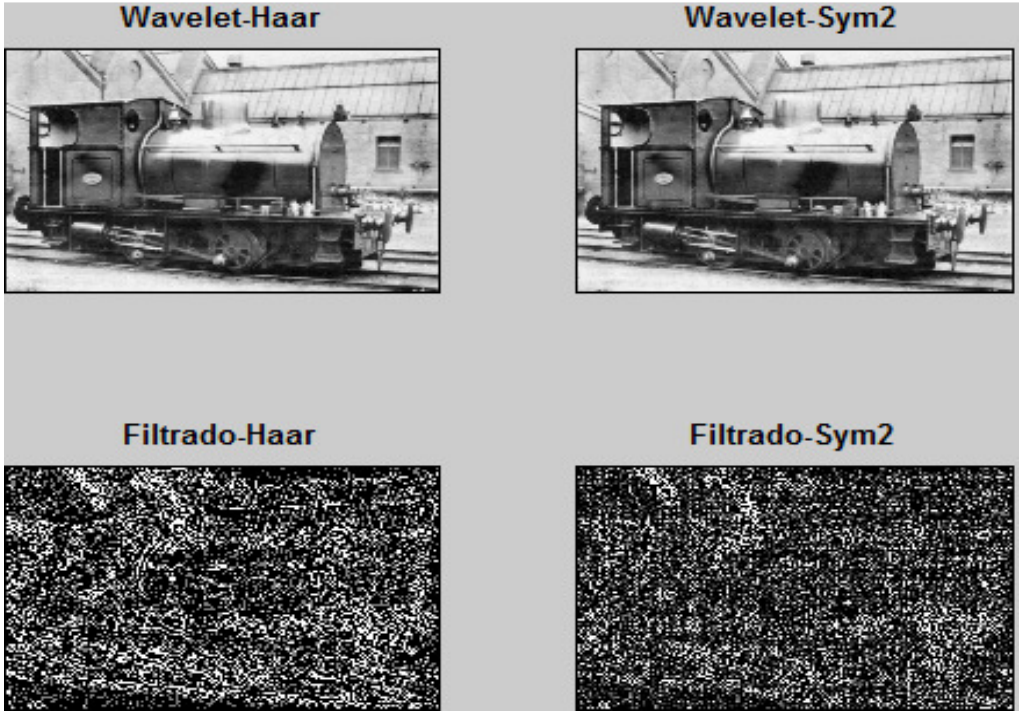

Fig. 2.15 Imagen de prueba 4 procesada con los filtros wavelet Haar y Sym2

	NMSE Wavelet Haar	**NMSE Wavelet Sym2**
Imagen 1	$0.181\ x\ 10^{-36}$	$0.637\ x\ 10^{-33}$
Imagen 2	$0.806\ x\ 10^{-36}$	$81.148\ x\ 10^{-33}$
Imagen 3	$0.334\ x\ 10^{-36}$	$41.285\ x\ 10^{-33}$
Imagen 4	$0.506\ x\ 10^{-36}$	$0.109\ x\ 10^{-33}$

Tabla 2.4 Comparación del NMSE del filtro wavelet Haar y Sym2

Como se puede apreciar en cada caso las imágenes resultantes muestran una mejor calidad visual que la original, aunque ambas son muy parecidas entre sí al usarse los filtros wavelet de primer orden.

Al momento de la descomposición de las imágenes se observa que se obtiene una mejor detección de bordes con el filtro wavelet Haar, razón por la que al momento de la reconstrucción de la imagen el NMSE es menor cuando se utiliza el filtro wavelet Haar.

	Tiempo Wavelet Haar (s)	**Tiempo Wavelet Sym2 (s)**
Imagen 1	0.4251	0.5092
Imagen 2	0.0958	0.1091
Imagen 3	0.2448	0.3042
Imagen 4	0.1740	0.2072

Tabla 2.5 Tiempo de procesamiento del filtro wavelet Haar y Sym2

De la tabla 2.5 podemos ver que el tiempo de procesamiento del filtro wavelet Haar es menor que el del filtro Sym2, sin embargo la diferencia máxima apenas es de 6 ms entre ambos.

CONCLUSIONES Y RECOMENDACIONES

1. Basados en el NMSE el filtrado wavelet Haar presenta un mejor desempeño que los otros filtros, pero visualmente los mejores resultados los produce el filtro Canny.

2. Para determinar los mejores resultados visuales, se mostró a un grupo de 20 personas de entre 20 y 40 años, las figuras de prueba 1 y 4, luego de haber sido restauradas utilizando los 4 métodos descritos anteriormente. Los resultados de esta encuesta se encuentran detallados en el Apéndice E. En ambos casos la mayoría de las personas escogió la imagen resultante después de haber sido

restaurada utilizando el filtro Canny como la que tenía mejor resultado visual.

3. El filtro Laplaciano resulta ser el menos eficiente debido a que el algoritmo del mismo solo detecta variaciones de intensidad en la escala de grises, por lo cual se obtienen varios falsos positivos de los bordes, lo que visualmente genera un oscurecimiento de la imagen. Además se pudo observar que las variaciones de alpha inciden en la detección de una mayor cantidad de falsos positivos mientras menor sea el valor del mismo, obteniéndose los mejores resultados con un alpha=1. Sin embargo el filtro Laplaciano es el que utiliza un menor tiempo de procesamiento que los demás filtros, siendo el filtro Canny el que presentó los tiempos de procesamiento más elevados de todos.

4. Resumiendo, al ser el objetivo principal de este proyecto la obtención de una buena imagen restaurada, el filtro a utilizar sería el de Canny, ya que visualmente es el que produjo mejores resultados, aunque el valor del NMSE y el tiempo de procesamiento son menores en el filtro wavelet Haar. En el caso de que la capacidad de procesamiento sea una limitante lo ideal sería utilizar el filtro wavelet Haar, ya que visualmente fue la segunda opción más elegida en la encuesta.

APENDICES

Apéndice A: Algoritmo del Filtro Laplaciano y Filtro Canny

```
function varargout = Filt_Lapl(varargin)
%FILT_LAPL M-file for Filt_Lapl.fig
%      FILT_LAPL, by itself, creates a new FILT_LAPL or raises the existing
%      singleton*.
%
%      H = FILT_LAPL returns the handle to a new FILT_LAPL or the handle
to
%      the existing singleton*.
%
%      FILT_LAPL('Property','Value',...) creates a new FILT_LAPL using the
%      given property value pairs. Unrecognized properties are passed via
%      varargin to Filt_Lapl_OpeningFcn.  This calling syntax produces a
%      warning when there is an existing singleton*.
%
%      FILT_LAPL('CALLBACK') and FILT_LAPL('CALLBACK',hObject,...) call
the
%      local function named CALLBACK in FILT_LAPL.M with the given input
%      arguments.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Filt_Lapl

% Last Modified by GUIDE v2.5 25-Feb-2009 22:47:37

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Filt_Lapl_OpeningFcn, ...
                   'gui_OutputFcn',  @Filt_Lapl_OutputFcn, ...
                   'gui_LayoutFcn',  [], ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
  gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Filt_Lapl is made visible.
function Filt_Lapl_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   unrecognized PropertyName/PropertyValue pairs from the
%            command line (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

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


% --- Outputs from this function are returned to the command line.
function varargout = Filt_Lapl_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



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

```
handles.Archivo=get(hObject, 'String'); %Almacenar en identificador
guidata(hObject,handles); %Salvar datos de la aplicación


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


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

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



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

% Hints: get(hObject,'String') returns contents of Val_alpha as text
%        str2double(get(hObject,'String')) returns contents of Val_alpha as a
double
Val=get(hObject,'String'); %Almacenar valor ingresado
NewVal = str2double(Val); %Transformar a formato double
handles.Val_alpha=NewVal; %Almacenar en identificador
guidata(hObject,handles); %Salvar datos de la aplicación

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

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



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

handles.Guardar_nombre=get(hObject, 'String'); %Almacenar en identificador
guidata(hObject,handles); %Salvar datos de la aplicación

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


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

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in ok_boton.
function ok_boton_Callback(hObject, eventdata, handles)
% hObject    handle to ok_boton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close all;
nombre = handles.Archivo;
alpha = handles.Val_alpha;
```

```
if (alpha < 0) || (alpha > 1 ) || isnan(alpha)
   errordlg('Alpha debe ser un valor entre 0.0 y 1.0','Error');
end
x = imread(nombre); % Se carga la imagen
figure('Name','Imagen Original','NumberTitle','off'); imshow(x);
x=double(x);

%Aplicacion del filtro Laplaciano a la imagen original
tic;
h = fspecial('laplacian',alpha);
laplac = imfilter(x,h);
time_lap=toc;
v = x - laplac;
min_f=min(min(v)); max_f=max(max(v)); % Obtengo los extremos
step_f=(max_f-min_f)/256; % Intervalo que corresponde a cada nivel
ima=uint8(round((v-min_f)/step_f));

nombre = handles.Guardar_nombre;
imwrite(ima,nombre);
[a,b]=size (nombre);
for c=1:1:b-4
   C(c) = nombre(c);
end
A1 = '_lap';
for d=1:1:4
   A2(d) = nombre(c+d);
end
nombre2 = horzcat(C,A1,A2);
imwrite(laplac,nombre2);

%Aplicacion del filtro Canny
tic;
BW = edge(x,'canny');
time_canny=toc;
A1 = '_can';
nombre3 = horzcat(C,A1,A2);
imwrite(BW,nombre3);
r=x+BW;
min_f=min(min(r)); max_f=max(max(r)); % Obtengo los extremos
step_f=(max_f-min_f)/256; % Intervalo que corresponde a cada nivel
ima3=uint8(round((r-min_f)/step_f));
A1 = '_fil';
Nombre4 = horzcat(C,A1,A2);
```

```
imwrite(ima3,nombre4);

%Se muestran las imagenes generadas
figure('Name','Imagenes Generadas','NumberTitle','off');
subplot(2,2,1), subimage(ima); %Imagen filtrada con Laplaciano
title('Filtro Laplaciano','fontsize',10,'fontweight','bold');
set(gca,'xtick',[],'ytick',[]); %Elimina los ejes de los graficos
subplot(2,2,2), subimage(ima3);%Imagen filtrada con Canny
title('Filtro Canny','fontsize',10,'fontweight','bold');
set(gca,'xtick',[],'ytick',[]); %Elimina los ejes de los graficos
subplot(2,2,3), subimage(laplac);%Bordes generados con el Laplaciano
title('Bordes Laplaciano','fontsize',10,'fontweight','bold');
set(gca,'xtick',[],'ytick',[]); %Elimina los ejes de los graficos
subplot(2,2,4), subimage(BW);%Bordes generados con Canny
title('Bordes Canny','fontsize',10,'fontweight','bold');
set(gca,'xtick',[],'ytick',[]); %Elimina los ejes de los graficos
ima=double(ima);
ima3=double(ima3);
MSE=mean(mean(mean(((x-ima).^2))));
PSNR_Lapl=20*log10(1/sqrt(MSE));
MSE=mean(mean(mean(((x-ima3).^2))));
PSNR_Canny=20*log10(1/sqrt(MSE));
save VAR_Lapl PSNR_Lapl PSNR_Canny time_lap time_canny;
clear; clc; pause, Menu;
%Desarrollado por German Andres Lopez
```

Apéndice B: Algoritmo de los filtros wavelet Haar y wavelet Sym2

```
function varargout = Filt_Wave(varargin)
% FILT_WAVE M-file for Filt_Wave.fig
%      FILT_WAVE, by itself, creates a new FILT_WAVE or raises the existing
%      singleton*.
%
%      H = FILT_WAVE returns the handle to a new FILT_WAVE or the handle
to
%      the existing singleton*.
%
%      FILT_WAVE('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in FILT_WAVE.M with the given input
arguments.
%
%      FILT_WAVE('Property','Value',...) creates a new FILT_WAVE or raises
the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Filt_Wave_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Filt_Wave_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Filt_Wave

% Last Modified by GUIDE v2.5 24-May-2009 13:06:38

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Filt_Wave_OpeningFcn, ...
                   'gui_OutputFcn',  @Filt_Wave_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
```

```
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Filt_Wave is made visible.
function Filt_Wave_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Filt_Wave (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

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


% --- Outputs from this function are returned to the command line.
function varargout = Filt_Wave_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function Archivo_Callback(hObject, eventdata, handles)
% hObject    handle to Archivo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

handles.Archivo=get(hObject, 'String'); %Almacenar en identificador
guidata(hObject,handles); %Salvar datos de la aplicación

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


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

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

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

handles.Guardar=get(hObject, 'String'); %Almacenar en identificador
guidata(hObject,handles); %Salvar datos de la aplicación

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

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

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

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

close all;
nombre = handles.Archivo;
nivel = handles.Nivel_Filtro;
if (nivel == 1 || nivel >4)
    errordlg('Nivel de Filtrado invalido','Error');
else
x = imread(nombre); % Se carga la imagen
figure('Name','Imagen Original','NumberTitle','off');imshow(x);
    if nivel == 2
        %Se utiliza el wavelet con el filtro haar de primer nivel
        tic;
        [b,c,d,e]=dwt2(x,'haar');
        X2 = idwt2(b,c,d,e,'haar');
        time_Haar = toc;
        X = X2;
    elseif nivel == 3
        %Se utiliza el wavelet con el filtro haar de segundo nivel
        tic;
        [b,c,d,e]=dwt2(x,'haar');
        [b2,c2,d2,e2]=dwt2(b,'haar');
        X2 = idwt2(b2,c2,d2,e2,'haar');
        X = idwt2(b,c,d,e,'haar');
        time_Haar = toc;
        e = e2;
    else
        %Se utiliza el wavelet con el filtro haar de tercer nivel
        tic;
        [b,c,d,e]=dwt2(x,'haar');
        [b2,c2,d2,e2]=dwt2(b,'haar');
        [b3,c3,d3,e3]=dwt2(b2,'haar');
        X2 = idwt2(b3,c3,d3,e3,'haar');
        X = idwt2(b,c,d,e,'haar');
        time_Haar = toc;
```

```
    e = e3;
end
min_f=min(min(X2)); max_f=max(max(X2)); % Obtengo los extremos
step_f=(max_f-min_f)/256; % Intervalo que corresponde a cada nivel
ima=uint8(round((X2-min_f)/step_f));
ima2=e;
nombre = handles.Guardar;
imwrite(ima,nombre);
[a,b]=size (nombre);
for h=1:1:b-4
    C(h) = nombre(h);
end
A1 = '_haar';
for d=1:1:4
    A2(d) = nombre(h+d);
end
nombre2 = horzcat(C,A1,A2);
imwrite(c,nombre2);
if nivel == 2
    %Se utiliza el wavelet sym2 de primer nivel
    tic;
    [b,c,d,e]=dwt2(x,'sym2');
    X2 = idwt2(b,c,d,e,'sym2');
    time_sym2 = toc;
    X3 = X2;
elseif nivel == 3
    %Se utiliza el wavelet sym2 de segundo nivel
    tic;
    [b,c,d,e]=dwt2(x,'sym2');
    [b2,c2,d2,e2]=dwt2(b,'sym2');
    X2 = idwt2(b2,c2,d2,e2,'sym2');
    X3 = idwt2(b,c,d,e,'sym2');
    time_sym2 = toc;
    e = e2;
else
    %Se utiliza el wavelet sym2 de tercer nivel
    tic;
    [b,c,d,e]=dwt2(x,'sym2');
    [b2,c2,d2,e2]=dwt2(b,'sym2');
    [b3,c3,d3,e3]=dwt2(b2,'sym2');
    X2 = idwt2(b3,c3,d3,e3,'sym2');
    X3 = idwt2(b,c,d,e,'sym2');
    time_sym2 = toc;
    e = e3;
```

```
    end
    min_f=min(min(X2)); max_f=max(max(X2)); % Obtengo los extremos
    step_f=(max_f-min_f)/256; % Intervalo que corresponde a cada nivel
    ima3=uint8(round((X2-min_f)/step_f));
    A1 = '_sym2';
    nombre3 = horzcat(C,A1,A2);
    imwrite(ima3,nombre3);
    A1 = '_sym2_2';
    nombre4 = horzcat(C,A1,A2);
    imwrite(e,nombre4);
%Se muestran las imagenes generadas
figure('Name','Imagenes Generadas','NumberTitle','off');
subplot(2,2,1), subimage(ima); %Imagen filtrada con wavelet-haar
title('Wavelet-Haar','fontsize',10,'fontweight','bold');
set(gca,'xtick',[],'ytick',[]); %Elimina los ejes de los graficos
subplot(2,2,2), subimage(ima3);%Imagen filtrada con wavelet-sym2
title('Wavelet-Sym2','fontsize',10,'fontweight','bold');
set(gca,'xtick',[],'ytick',[]); %Elimina los ejes de los graficos
subplot(2,2,3), subimage(ima2);%Filtrado wavelet-haar
title('Filtrado-Haar','fontsize',10,'fontweight','bold');
set(gca,'xtick',[],'ytick',[]); %Elimina los ejes de los graficos
subplot(2,2,4), subimage(e);%Filtrado wavelet-sym2
title('Filtrado-Sym2','fontsize',10,'fontweight','bold');
set(gca,'xtick',[],'ytick',[]); %Elimina los ejes de los graficos
x=double(x);
ima=double(X);
ima3=double(X3);
[j k]=size(x);
[l m]=size(ima);
o=min(j,l);
p=min(k,m);
for r=1:1:o
    for s=1:1:p
        x2(r,s)=ima(r,s);
    end
end
for r=1:1:o
    for s=1:1:p
        x3(r,s)=ima3(r,s);
    end
end

Normal=0;
for t=1:1:j
```

```
   for u=1:1:k
      Normal=Normal+(x(t,u).^2);
   end
end

MSE=mean(mean(mean(((x-x2).^2))));
NMSE_Haar=MSE/Normal;
MSE=mean(mean(mean(((x-x3).^2))));
NMSE_Sym2=MSE/Normal;
save VAR_Wave NMSE_Haar NMSE_Sym2 time_Haar time_sym2;
clear; clc;pause, Menu;
end
%Desarrollado por German Andres Lopez
```

Apéndice C: Pantalla de presentación del proyecto

```
function Presentacion
clear,clc,cla,close all
%Creamos figura
figdiag=figure('Units','Pixels',...
    'Position',[0 0 635 425],... %Tamaño de la presentacion
'Number','off',...
'Name','Procesamiento Digital de Señales', ...
'Menubar','none', 'color',[0 0 0]);
%Ubicamos ejes en figura
axes('Units','Normalized','Position',[0 0 1 1]);
%-----Centramos la figura---------
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
%--------------------------------------
%Incluir imagen
%Importamos imagen *.jpg,junto con su mapa de colores
[x,map]=imread('Fondo.jpg','jpg');
%Representamos imagen en figura, con su mapa de colores
image(x),colormap(map),axis off,hold on
%Títulos sobre imagen
%Título
text(30,40,'Restauración de Imagenes','Fontname','Arial','Fontsize',...
    25,'Fontangle','Italic','Fontweight','Bold','color',[1 1 0]);
%Nombre del programador
text(30,80,'Por: German Lopez Peña','Fontname', ...
'Arial','Fontangle','Italic','Fontweight','Bold', ...
'Fontsize',14,'color',[1 1 1]);
%Botón Continuar
botok=uicontrol('Style','pushbutton', ...
'Units','normalized', ...
'Position',[.84 .03 .12 .05], ...
'String','CONTINUAR',...
'Callback','clear all; close all;clc; Menu;'); %Menu es el nombre del
                                        %siguiente programa.
%Desarrollado por German Andres Lopez
```

Apéndice D: Pantalla del Menú de opciones

```
function varargout = Menu(varargin)
% MENU M-file for Menu.fig
%      MENU, by itself, creates a new MENU or raises the existing
%      singleton*.
%
%      H = MENU returns the handle to a new MENU or the handle to
%      the existing singleton*.
%
%      MENU('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MENU.M with the given input arguments.
%
%      MENU('Property','Value',...) creates a new MENU or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Menu_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Menu_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Menu

% Last Modified by GUIDE v2.5 25-Feb-2009 20:14:38

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Menu_OpeningFcn, ...
                   'gui_OutputFcn',  @Menu_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
```

```
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before Menu is made visible.
function Menu_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Menu (see VARARGIN)

background = imread('fondo1.jpg'); %Leer imagen
imshow(background); %Presenta la imagen
% Choose default command line output for Menu
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

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


% --- Outputs from this function are returned to the command line.
function varargout = Menu_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in Wavelet.
function Wavelet_Callback(hObject, eventdata, handles)
% hObject    handle to Wavelet (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clear; clc; close all;
Filt_Wave;   %Llama al programa que ejecuta el filtro mediante Wavelet
```

```
% --- Executes on button press in Filtro_Laplaciano.
function Filtro_Laplaciano_Callback(hObject, eventdata, handles)
% hObject    handle to Filtro_Laplaciano (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clear; clc; close all;
Filt_Lapl;   %Llama al programa que ejecuta el filtro Laplaciano

%Desarrollado por German Andres Lopez
```

Apéndice E: Resultados de la encuesta

Para determinar el filtro con el cual se obtiene un mejor resultado visual, se procedió a realizar una encuesta a un grupo de veinte personas de las cuales respondieron un total de catorce. Los siguientes gráficos muestran los resultados obtenidos.

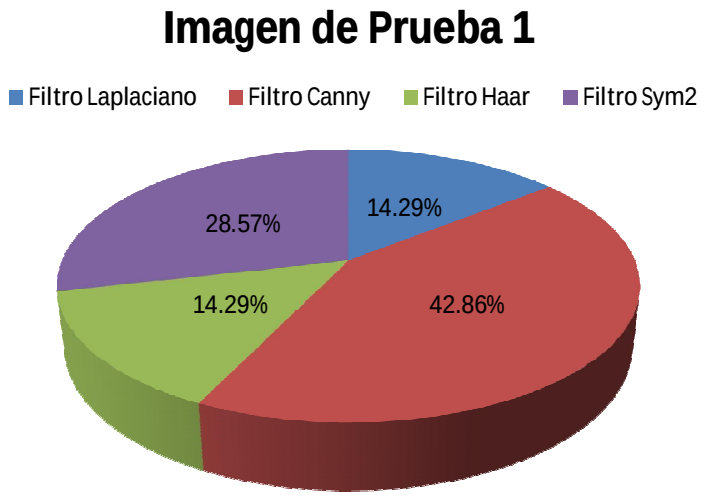

Fig. 4.1 Resultados de la encuesta con la imagen de prueba 1

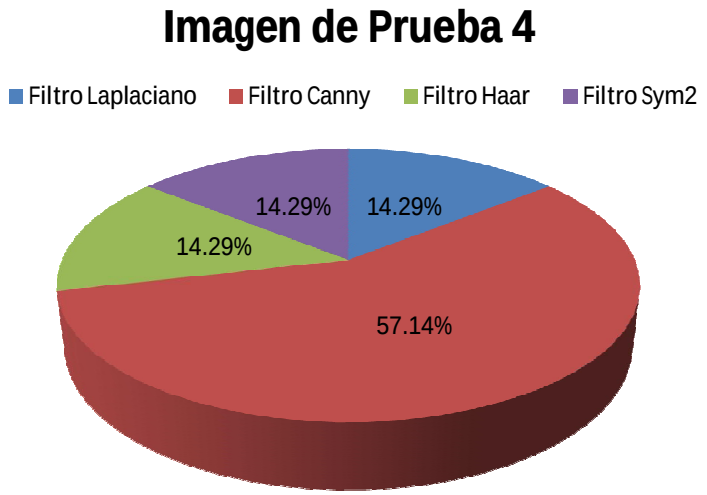

Fig. 4.2 Resultados de la encuesta con la imagen de prueba 4

APENDICE F: MANUAL DEL USUARIO

1. Para el uso del programa se deben copiar en la carpeta de trabajo de Matlab los archivos: Filt_Lapl.fig, Filt_Lapl.m, Filt_Wave.fig, Fig_Wave.m, Fondo.jpg, Fondo1.jpg, Menu.fig, Menu.m y Presentacion.m.

2. En la ventana de comandos tipear Presentacion. Se presentará una pantalla de presentación.

3. Dar clic en el botón CONTINUAR.

4. Se cargará una ventana con el Menú, en el cual se presentan las dos opciones disponibles.

Fig. 5.1 Menú de opciones

Uso del Filtro Laplaciano

1. Para acceder al proceso de restauración que utiliza los filtros Laplacianos y Canny se debe dar clic sobre el correspondiente botón en el Menú.

2. Se cargará una nueva ventana en la cual se debe ingresar en el primer recuadro el nombre de la imagen en blanco y negro que se desea someter al proceso de restauración. Si la imagen a procesar se encuentra en la carpeta de trabajo basta ingresar el nombre del archivo con su respectiva extensión (nombre.ext), de lo contrario se debe ingresar la ruta donde se encuentra ubicado el archivo, por ejemplo C:\Imagenes\imagen.jpg. La lista de extensiones soportadas se detallan en la Tabla 5.1.

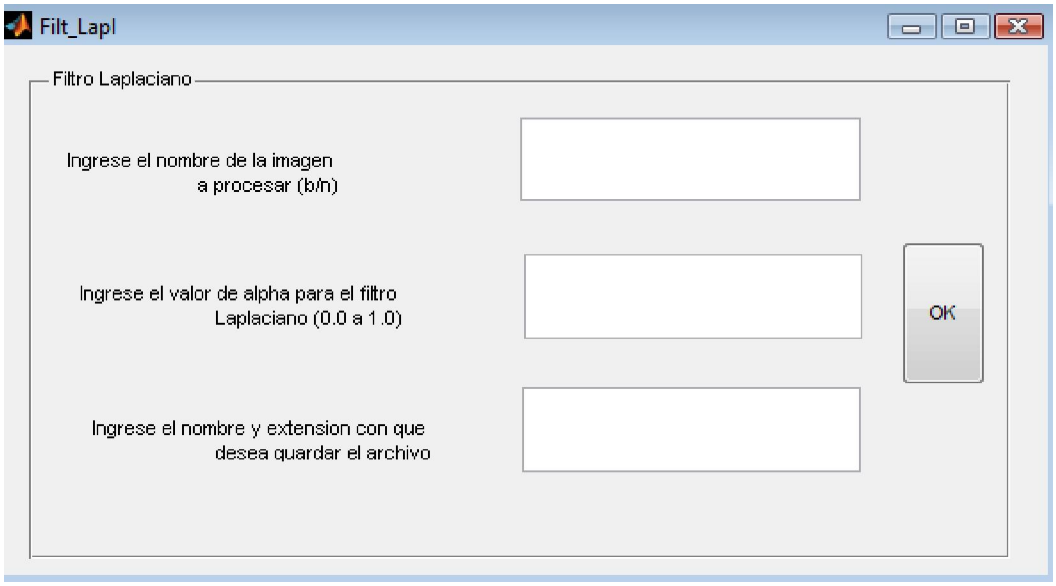

Fig. 5.2 Ventana del Filtro Laplaciano

3. En el segundo recuadro se debe ingresar el valor de alpha usado para crear el Filtro Laplaciano, debe ser un valor comprendido entre 0.0 y 1.0.

4. En el tercer recuadro se debe ingresar el nombre y extensión (nombre.ext) con el que se desea guardar el archivo generado después de haber sometido la imagen original al Filtro Laplaciano. La lista de extensiones soportadas se detallan en la Tabla 5.1.

5. Una vez ingresados estos tres parámetros se debe dar clic en el botón OK, para comenzar a procesar la imagen.

6. Se abrirá una ventana llamada Imagen Original, en la cual se mostrará la imagen ingresada antes de ser procesada.

7. El programa generará en la carpeta de trabajo cuatro archivos correspondientes a igual cantidad de imágenes. La descripción de las imágenes obtenidas se detallan en la Tabla 5.2.

8. Adicionalmente se creará un archivo VAR_Lapl.mat donde se guardarán cuatro variables del programa las cuales son: PSNR_Lapl, PSNR_Canny, time_lap y time canny.

9. Se abrirá una ventana adicional llamada Imágenes Generadas en la cual se mostrarán las 4 imágenes generadas que se detallaron en el paso anterior.

10. Las ventanas se mantendrán abiertas hasta que se las cierre manualmente o al presionar cualquier tecla sobre la ventana de comandos de Matlab, lo cual cerrará ambas ventanas y presentará nuevamente la ventana del Menú.

11. Para finalizar el programa se debe cerrar manualmente la venta del Menú.

Uso del Filtro Wavelet

1. Para acceder al proceso de restauración que utiliza los filtros Wavelet Haar y Wavelet Sym2, se debe dar clic sobre el correspondiente botón en el Menú, lo cual cerrará la ventana actual.

2. Se cargará una nueva ventana en la cual se debe ingresar en el primer recuadro el nombre de la imagen en blanco y negro que se desea someter al proceso de restauración. Si la imagen a procesar se encuentra en la carpeta de trabajo basta ingresar el nombre del archivo con su respectiva extensión (nombre.ext), de lo contrario se debe ingresar la ruta donde se encuentra ubicado el archivo por ejemplo C:\Imagenes\imagen.jpg. La lista de extensiones soportadas se detallan en la Tabla 5.1.

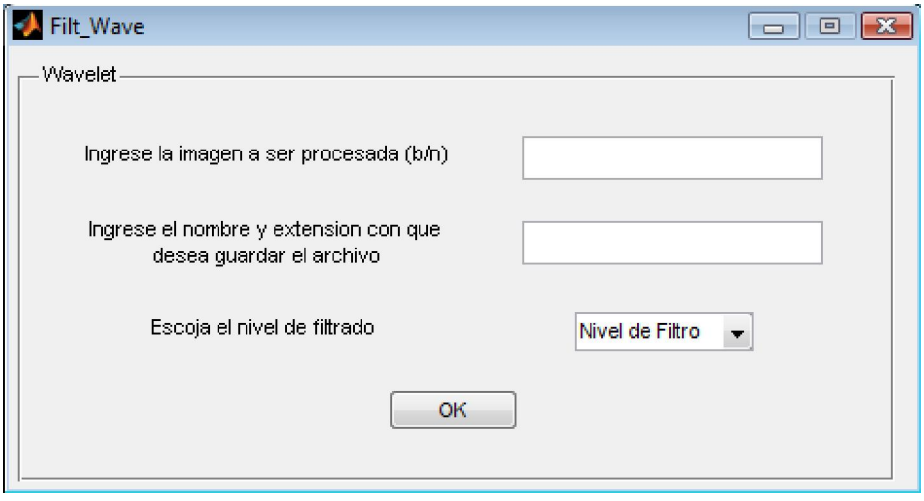

Fig. 5.3 Ventana del Filtro Wavelet

3. En el segundo recuadro se debe ingresar el nombre y extensión (nombre.ext) con el que se desea guardar el archivo generado después de haber sometido la imagen original al Filtro Wavelet. La lista de extensiones soportadas se detallan en la Tabla 5.1.

4. Finalmente se presenta un menú donde se debe escoger el nivel de de filtrado al que se someterá la imagen, tanto para el filtro Haar como para el filtro Sym2.

5. Una vez ingresados estos parámetros se debe dar clic en el botón OK, para comenzar a procesar la imagen.

6. Se abrirá una ventana llamada Imagen Original, en la cual se mostrará la imagen ingresada antes de ser procesada.

7. El programa generará en la carpeta de trabajo cuatro archivos correspondientes a igual cantidad de imágenes. La descripción de las imágenes obtenidas se detallan en la Tabla 5.3.

8. Se abrirá una ventana adicional llamada Imágenes Generadas en la cual se mostrarán las 4 imágenes generadas que se detallaron en el paso anterior.

9. Adicionalmente se creará un archivo VAR_Wave.mat donde se guardarán cuatro variables del programa las cuales son: PSNR_Haar, PSNR_Sym2, time_Haar y time_sym2.

10. Las ventanas se mantendrán abiertas hasta que se las cierre manualmente o al presionar cualquier tecla sobre la ventana de comandos de Matlab, lo cual cerrará ambas ventanas y presentará nuevamente la ventana del Menú

11. Para finalizar el programa se debe cerrar manualmente la venta del Menú.

Formato	Nombre completo
’bmp’	Windows Bitmap (BMP)
’gif’	Graphics Interchange Format (GIF)
’hdf’	Hierarchical Data Format (HDF4)
’jpg’ o ’jpeg ’	Joint Photographic Experts Group (JPEG)
’pbm’	Portable Bitmap (PBM)
’pcx’	Windows Paintbrush (PCX)
’pgm’	Portable Graymap (PGM)
’png’	Portable Network Graphics (PNG)
’pnm’	Portable Anymap (PNM)
’ppm’	Portable Pixmap (PPM)
’ras’	Sun Raster (RAS)
’tif’ o ’tiff’	Tagged Image File Format (TIFF)
’xwd’	X Windows Dump (XWD)

Tabla 5.1 Extensiones de imágenes soportadas por Matlab

Nombre del archivo	Descripción
Nombre.ext	Imagen obtenida después de filtrar la imagen original mediante el filtro laplaciano.
Nombre_lap.ext	Imagen de los bordes detectados mediante el filtro laplaciano.
Nombre_fil.ext	Imagen obtenida después de filtrar la imagen original mediante el filtro Canny
Nombre_can.ext	Imagen de los bordes detectados mediante el filtro Canny.

Tabla 5.2 Detalle de los archivos generados con el filtro Laplaciano

Nombre del archivo	Descripción
Nombre.ext	Imagen obtenida después de filtrar la imagen original mediante el filtro wavelet haar.
Nombre_haar.ext	Imagen de los bordes detectados mediante el filtro wavelet haar.
Nombre_sym2.ext	Imagen obtenida después de filtrar la imagen original mediante el filtro wavelet sym2.
Nombre_sym2_2.ext	Imagen de los bordes detectados mediante el filtro wavelet sym2.

Tabla 5.3 Detalle de los archivos generados con el filtro wavelet

BIBLIOGRAFIA

1. Canny, John. "A Computational Approach to Edge Detection." IEEE Transactions on Pattern Analysis and Machine, Vol. PAMI-8, No. 6, 1986, 679-698.

2. Daubechies, Ingrid. Ten lectures on wavelets. Philadelphia: Society for Industrial and Applied Mathematics, 1994, 194 - 202.

3. Madisetti, Vijay K., and Douglas B. Williams. Digital Signal Processing Handbook. Atlanta: CRC Press LLC, 1999, 1087 - 1105.

4. Borrell, Guillem. Introducción Informal a Matlab. Fe af, 2000. http://iimyo.forja.rediris.es/matlab/cursolatex007.html.

5. Casas, María. Alasbimn Journal 10. Abril 1, 2008. http://www.alasbimnjournal.cl/alasbimn/index.php?option=com_content&task=view&id=351&Itemid=149.

6. Sánchez, Omar. Modelos, Control y Sistema de Visión. Noviembre 1, 2008. http://omarsanchez.net/filtroespa.aspx.