UNIVERSIDAD POLITÉCNICA DE MADRID

DEPARTAMENTO DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, AUTOMÁTICA Y FÍSICA APLICADA

Prácticas Visión Artificial

Práctica 4

Prácticas de procesado de las imágenes

4 Técnicas de procesamiento de las imágenes I

En esta práctica se tratará de experimentar con las técnicas clásicas del procesado de imágenes empleando *"Image Procesing Toolbox"* de Matlab. Se revisarán tanto las que derivan de Procesamiento Digital de Señales como aquellas de carácter más heurístico. En primer lugar se analizarán los conceptos de convolución discreta 2D y de respuesta en frecuencia. Posteriormente se pasará a las técnicas de preprocesado de realzado. Por último se experimentará con la herramienta de depuración de Matlab.

4.1 Técnicas del procesamiento digital de señales

La operación de convolución discreta corresponde con el procesado lineal de las imágenes. Se fundamenta en aplicar una máscara de convolución que corresponda a un tipo de filtro determinado. Los filtros paso bajo suavizarán las imágenes y los filtros paso alto permitirán determinar los bordes.

El filtro binomial es un filtro de fase lineal empleado para el suavizado y eliminación del ruido. En 1D tiene el aspecto de:

```
bino1D = [1 2 1]./4;
[G,W] = freqz (bino1D, 1,64);
figure(1);
plot(W,abs(G));
figure(2);
plot(W,angle(G).*(180/pi));
```

Su extensión a 2D será:

```
bino2D = conv2(bino1D,bino1D');
resp_frBino2D = fft2(bino2D,64,64);
figure(1);
surf(abs(resp_frBino2D));
figure(2);
surf(fftshift(abs(resp_frBino2D)));
```

Interprete la respuesta en frecuencia. ¿Cuál es el efecto de ***fftshift()*** ?. Decida sobre la simetría de la transformada de Fourier y qué zonas corresponde a la baja frecuencia y qué otras a la alta.

Para ver el efecto de la máscara de convolución aplíquese a la imagen del hombre de la cámara:

```
imgEnt = imread('cameraman.tif');
imgSal = imfilter(imgEnt,bino2D);
imshow([imgEnt,imgSal]);
```

Observe el efecto de los bordes. Utilice las condiciones de Neumann (replicar los bordes). Considérese la máscara de Sobel y aplíquese con la anterior imagen:

```
maskSobel = fspecial('sobel')
imgSal = imfilter(imgEnt,maskSobel,'replicate');
imshow([imgEnt,imgSal]);
```

Efectivamente, la máscara es utilizada para implementar la derivada en el eje X. Varíe el código para obtener la derivada en el eje Y. Presente los resultados de manera que quede como en la siguiente figura:

4.2 Transformadas de Fourier

Para una mejor comprensión de las transformadas de Fourier, hágase la antitransformada de la segunda componente y de la dieciseisava en la dirección de franjas verticales

```
V=zeros(64,64);V(1,1+2)=2000;V(1,64-2)=2000;
figure(1);
surf(fftshift(V));title('Componente 2 eje de las frec. vertical');
figure(2);imshow(abs(ifft2(V,64,64)));

figure(1);
V=zeros(64,64);V(1,1+16)=2000;V(1,64-16)=2000;
surf(fftshift(V));title('Componente 16 eje de las frec. vertical');
figure(2);imshow(abs(ifft2(V,64,64)));
```

Modifique la transformada de Fourier para que aparezcan franjas horizontales.

Véase la utilidad de eliminar las componentes de alta frecuencia de una imagen contaminada por el ruido:

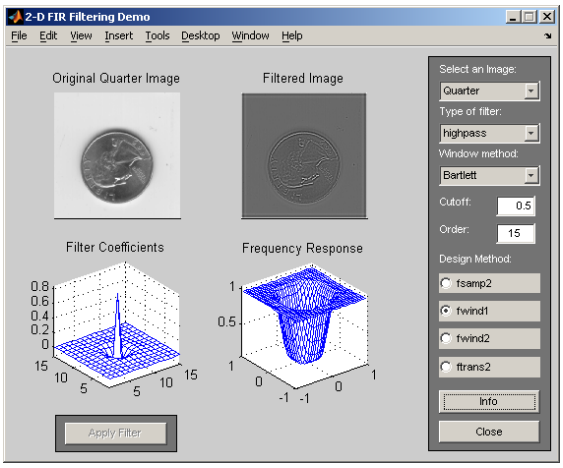
```
%% Filtro paso bajo
imagen=(imread('cameraman.tif'));
n = size(imagen,1);
k = round(.8*n); k = round(k/2)*2;
V=fft2(double(imagen));
V(n/2-k/2+2:n/2+k/2, n/2-k/2+2:n/2+k/2) = 0;
```

```
imshow([imagen,uint8(ifft2((V)))])
```

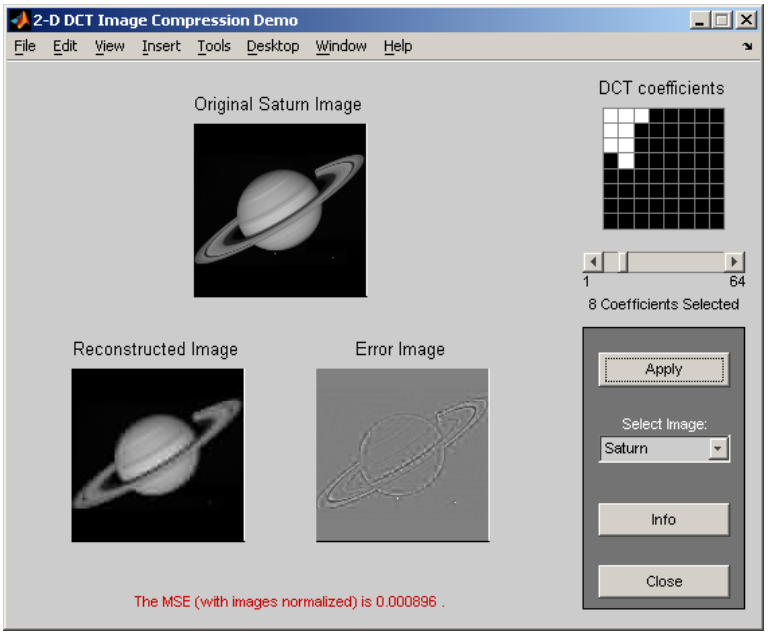
Y de cancelar las componentes de baja frecuencia:

```
%% Filtro paso alto
imagen= imread('cameraman.tif');
V=fft2(double(imagen));
V(1:10,1:10)=0;V(256-9:256,1:10)=0;
V(1:10, 256-9:256)=0;V(256-9:256,256-9:256)=0;
imshow([imagen,uint8(ifft2((V)))]);
```

No obstante, este proceder no es adecuado por el alto coste computacional. Es preferible diseñar filtros e implementarlos mediante máscaras de convolución. Emplearemos la demo *firdemo* para ver la potencia del diseño de los filtros, su respuesta en frecuencia y sus resultados al convolucionarlos con las imágenes



Las transformas de Fourier si que son aplicadas para la compresión de las imágenes. Aunque realmente se emplean las transformadas discretas del coseno. Mediante la demo *dctdemo* se puede observar cómo sólo almacenado unas pocas componentes es posible la reconstrucción con muy poca perdida de la información.



4.3 Técnicas de manipulación del histograma

En primer lugar se va a tratar de las operaciones de modificación del brillo, del contraste y de corrección de gamma mediante las funciones de transferencia del histograma. Capture una imagen de resonancia magnética y mejore su contraste:

```
%% Manipulación del histograma
imagen=imread('mri.tif');
imhist(imagen);
max(imagen(:))
pause;

imagen_sal=imadjust(imagen);
imhist(imagen_sal);
imshow([imagen,imagen_sal]);
[std2(imagen),std2(imagen_sal)]
```

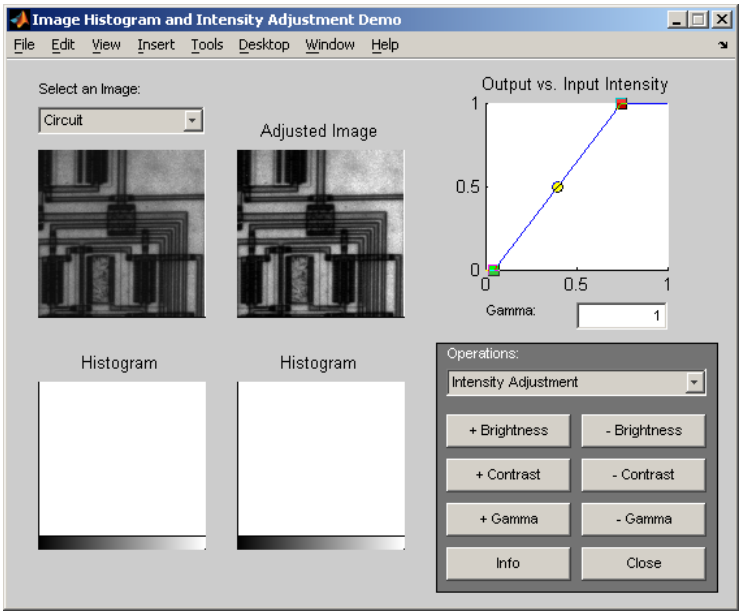
La ecualización del histograma es efectuada por la función *histeq*:

```
%% Ecualización
imagen=imread('tire.tif');
imhist(imagen);
imagen_sal=histeq (imagen);
imhist(imagen_sal);
imshow([imagen,imagen_sal]);
```

Para las operaciones de realce mediante el énfasis sobre las componentes de alta frecuencia se recurrirá al algoritmo de *unsharp masking*:

```
%% Unsharp
imagen=imread('cameraman.tif');
H=fspecial('unsharp')
imagen_sal=imfilter (imagen, H);
imshow([imagen,imagen_sal]);
```

Las técnicas de procesamiento punto a punto basadas en la adecuación del rango dinámico, así como las técnicas de ecualización del histograma pueden ser observadas en la demo *imadjdemo*. Cargue cualquier imagen y haga variaciones sobre el brillo, el contraste y la corrección de gamma. Posteriormente, empleé las técnicas de ecualización del histograma. Por último, efectué la transformación necesaria para conseguir el negativo de una imagen.

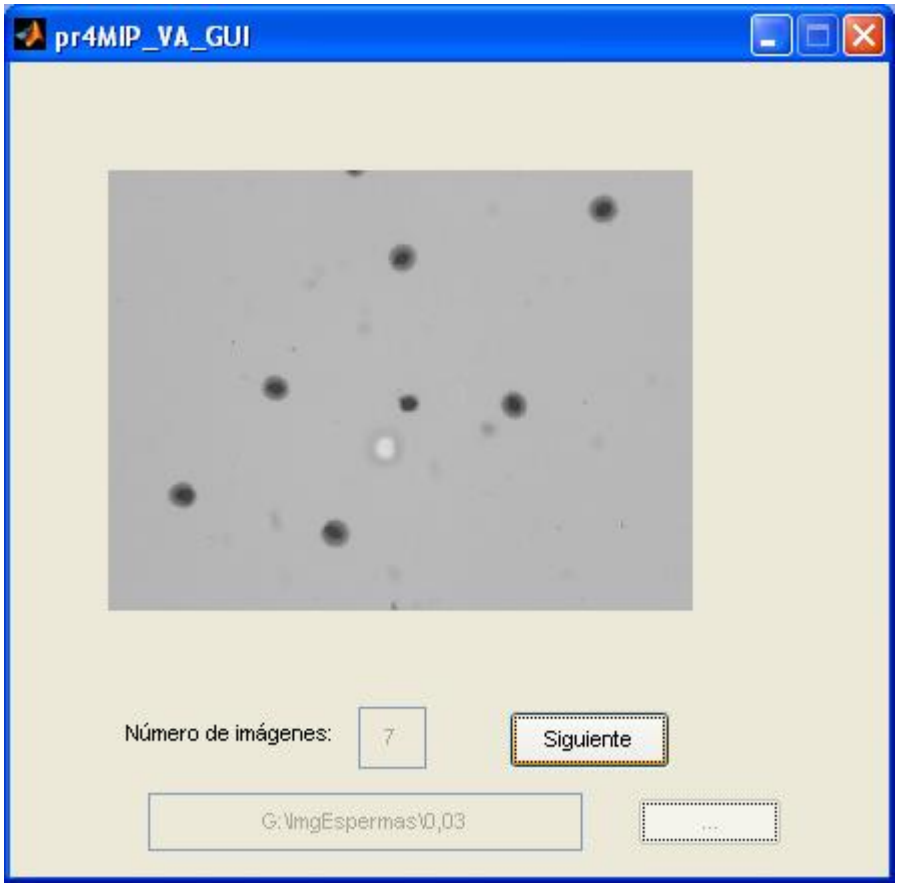


4.4 Depuración en Matlab

Para el análisis y depuración de los ficheros scripts de Matlab se usan las siguientes teclas:

- F5: Ejecución del programa hasta la localización de un punto de ruptura (*breakpoint*).
- F9: Ejecución de una sentencia.
- F10: Ejecución paso a paso
- F11: Ejecutar el interior de una función.

También es posible emplear los iconos y/o menús existentes en la barra de herramientas de depuración. Se empleará el visualizador de imágenes de un directorio para experimentar con las técnicas de depuración. Se ha elegido esta aplicación por su conexión con los trabajos propuestos.



Al ejecutar la aplicación pr4MIP_VA_GUI, todo está desactivado menos el botón para buscar el directorio. Para entender el funcionamiento del programa se colocará un punto de ruptura en '*callback'* de este botón.

```
76    % --- Executes on button press in pushbutton1.
77    function pushbutton1_Callback(hObject, eventdata, handles)
78    % hObject    handle to pushbutton1 (see GCBO)
79    % eventdata  reserved - to be defined in a future version of MATLAB
80    % handles    structure with handles and user data (see GUIDATA)
81    data = guihandles(gcbf);
82    directorio = uigetdir(pwd,'Directorio de imágenes');
83
84    if(directorio==0)
85        return;
86    end
87    set(data.edit1,'Enable','on');
88    set(data.edit1,'String',directorio);
89    set(data.edit1,'Enable','off');
90
```

Pulse F5 y procede a ejecutar el programa paso a paso con F10. Observe las variables del programa en el '*Workspace'* y analice su comportamiento.

4.4.1 Ejercicios propuestos

1. Documentar el funcionamiento de la aplicación.

2. Depurarlo hasta su funcionamiento óptimo.