MEJORA Y FILTRADO DE IMÁGENES CON FILTRADO ESPACIAL EN MATLAB

Las técnicas de filtrado eliminan ruido de las imágenes a través del filtrado. El mejor método para una situación dada depende de la imagen y del tipo de ruido o degradación. Así, a modo de ejemplo, tenemos:

-Los filtros lineales proporcionan simplicidad y velocidad, y son los más útiles cuando el ruido está limitado a una banda de frecuencia conocida.

-Los filtros mediana son muy efectivos para eliminar el ruido '*salt & pepper*', píxeles a uno y cero que se originaron durante la traducción y la discretización.

-Los filtros Wiener son filtros lineales adaptativos basados en las características de varianza locales de la imagen. Los filtros Wiener suavizan gradualmente cambiando áreas de una imagen donde el ruido es muy aparente, pero manteniendo áreas donde los detalles están presentes y el ruido es menos aparente.

Todas las funciones de restauración de imágenes trabajan con imágenes de intensidad. Para aplicar estos algoritmos a imágenes de color, habría que procesar las componentes rojo, verde y azul separadamente.

Comprender filtros espaciales en MATLAB.

En MATLAB, un filtro FIR bidimensional es un array ordenado de coeficientes, h(n1,n2), los cuales pueden introducirse en MATLAB como una matriz:

```
h =[  9   4  -3  -5   6
      6  -7  12   7   0
      2   4  -1   4   1
      5   0  -2  -4   5];
```

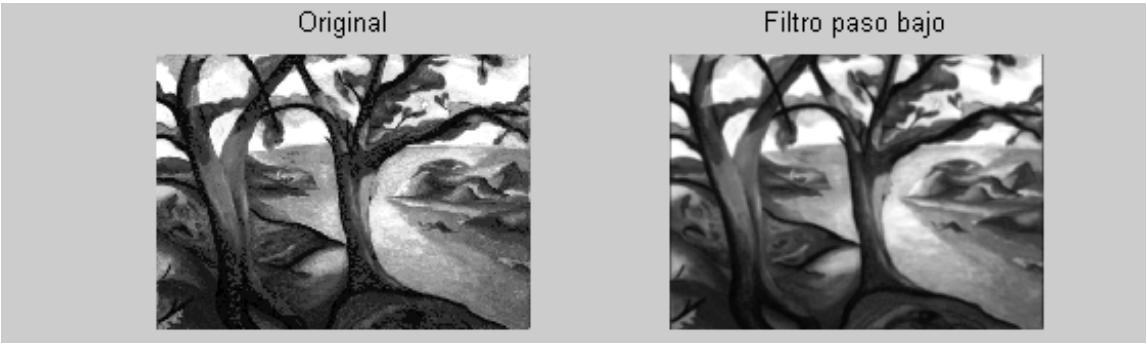
Esta representación del filtro es conocida como "máscara". La función de la toolbox *filter2* implementa el filtrado FIR bidimensional aplicando la máscara a los datos de la imagen.

Diseño de filtros espaciales

Se va a proceder a filtrar, en el dominio del espacio, una imagen con diversas máscaras observando el efecto que producen. Primeramente, se va a utilizar un filtro de promediado en el que el valor que se le da a un píxel es la media de los del entorno. Para ello, haga un programa con los siguientes comandos:

```
% Filtrado espacial de una imagen con máscara
load trees;
I=ind2gray(X,map); %Se trabaja con una imagen a nivel de gris double [0 1]
h=1/9*[1 1 1;1 1 1; 1 1 1] ; %Definición de la máscara. Filtro paso bajo
Y=filter2(h,I); %Por defecto, obtiene filtrado tras centrar "h" en píxeles dentro de la imagen de
                % entrada.
 % Y=filter2(h,I,'full'); %Obtiene el resultado completo del filtrado. Mas informacion: help filter2
figure, subplot(1,2,1),imshow(I),title('Original');
subplot(1,2,2),imshow(Y),title('Filtro paso bajo');
```

El resultado[1] se puede apreciar en la siguiente figura:



Se puede obtener la respuesta frecuencial del filtro mediante la función:

```
figure, freqz2(h)    %Escriba help freqz2 para obtener mas informacion sobre esta funcion
```

Realice el mismo proceso utilizando las máscaras mostradas en la página siguiente (SMOOTH, SHARPEN...) y extraiga conclusiones.

Para ello tenga en cuenta que, tras aplicar algunas máscaras, es posible que la imagen de salida, de tipo double, denominada Y en el ejemplo anterior, tenga niveles fuera del intervalo [0—1] o [0—255] (según la codificación de la imagen de entrada) pues algunos de estos filtros dan valores (bien positivos o negativos) superiores en módulo al nivel de gris máximo cuando detectan algún punto aislado, línea etc. Es fácil de ver si el rango de la imagen de salida está fuera del rango [0—1] o [0—255] viendo el valor mínimo total de la imagen con *min(min(Y))* y el máximo con *max(max(Y)*.

[1] Compruebe el resultado y tamaño de Y=filter2(h,I,'full') en comparación con Y=filter2(h,I)

Como solución:

1) Puede visualizar la nueva imagen con imagesc que escala los datos, aunque de forma interna y sólo de cara a la visualización, para usar todo el *colormap* (mapa de colores).

```
figure, colormap(gray(256)), imagesc(Y)
```

2) Puede normalizar la imagen de salida al rango [0-1], restando a cada píxel el valor mínimo de toda la imagen y dividiendo por el rango o diferencia entre el valor máximo y mínimo, y luego visualizar del modo normal con *imshow*.

```
n_Y=(Y-min(min(Y)))/(max(max(Y))-min(min(Y)));
figure, imshow(n_Y)
```

SMOOTH

$$1/12\begin{bmatrix}1&2&1\\1&2&1\\1&2&1\end{bmatrix}$$

SHARPEN

$$\begin{bmatrix}-1&-1&-1\\-1&9&-1\\-1&-1&-1\end{bmatrix}$$

HORIZ_EDGE

$$\begin{bmatrix}-2&-2&-2\\0&0&0\\2&2&2\end{bmatrix}$$

VERT_EDGE

$$\begin{bmatrix}-2&0&2\\-2&0&2\\-2&0&2\end{bmatrix}$$

LAPLACIANA

$$\begin{bmatrix}0&-1&0\\-1&4&-1\\0&-1&0\end{bmatrix}$$

EDGE

$$\begin{bmatrix}1&1&1\\0&0&0\\-1&-1&-1\end{bmatrix}+\begin{bmatrix}-1&0&1\\-1&0&1\\-1&0&1\end{bmatrix}$$

En el último filtro como el segundo y sumar los resultados obtenidos para detectar bordes (horizontales y verticales):

```
close all
load gatlin; im=ind2gray(X,map);figure,imshow(im)
h=[ 1 1 1; 0 0 0; -1 -1 -1]; %filtro detector de bordes horizontales
Y=filter2(h,im); %imagen bordes horizontales
bordes_hori=(Y-min(min(Y)))/(max(max(Y))-min(min(Y))); %imagen bordes horizont.
normalizada
figure, imshow(bordes_hori);
Y=filter2(h',im); %imagen bordes verticales pues se filtro con h' = filtro de bordes verticales
bordes_verti=(Y-min(min(Y)))/(max(max(Y))-min(min(Y))); %imagen bordes verticales
normalizada
figure, imshow(bordes_verti)
Y=bordes_hori+bordes_verti; %imagen bordes
bordes=(Y-min(min(Y)))/(max(max(Y))-min(min(Y))); %imagen bordes normalizada
figure,imshow(bordes)
```

Filtrado de imágenes en color

Para filtrar una imagen en color se puede filtrar cada una de sus componentes R, G, B separadamente con un filtro (igual o distinto) para cada una de ellas.

```
rgb=imread('flowers.tif'); %Los valores R,G,B de la imagen rgb están entre 0 y 255 (uint8)
[filas, columnas, profundidad_color]=size(rgb) %Tamano de la imagen
h=ones(5,5)/25;            %Mascara. Filtro paso bajo
r=rgb(:,:,1);     %Los valores r están entre 0 y 255 (uint8)
g=rgb(:,:,2);     %Los valores g están entre 0 y 255 (uint8)
b=rgb(:,:,3);     %Los valores b están entre 0 y 255 (uint8)
n_r=filter2(h,r);  %Los valores n_r están entre 0'0 y 255'0 (double)
n_g=filter2(h,g);  %Los valores n_g están entre 0'0 y 255'0 (double)
n_b=filter2(h,b);  %Los valores n_b están entre 0'0 y 255'0 (double)
rgb1=zeros(filas,columnas,profundidad_color);
% rgb1 = imagen inicializada a ceros donde meter componentes filtradas
rgb1(:,:,1)=n_r;
rgb1(:,:,2)=n_g;
rgb1(:,:,3)=n_b;
rgb1=uint8(rgb1);    %Convierto la imagen a valores entre 0 y 255 de tipo uint8 (en vez de
double)
figure,imshow(rgb),title('Imagen original');
figure,imshow(rgb1),title('Imagen filtrada');
```

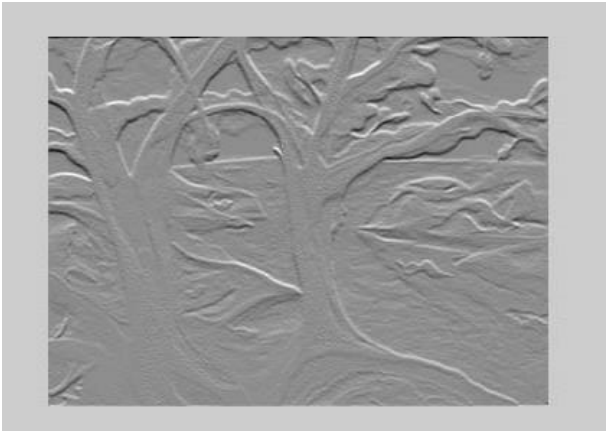
O se puede aplicar el mismo filtro a las 3 componentes como se hace en el siguiente ejemplo (con una única función) con la función imfilter (versión 6.1 de Matlab o superior):

```
rgb=imread('flowers.tif'); %Los valores R,G,B de la imagen rgb están entre 0 y 255 (uint8)
h=ones(5,5)/25;            %Mascara. Filtro paso bajo
rgb2=imfilter(rgb,h);     %Los valores R,G,B la imagen rgb2 están entre 0 y 255 (uint8)
%imfilter se puede usar tambien en imagenes en niveles de gris en lugar de filter2. Vea: help
imfilter
figure,imshow(rgb),title('Imagen original');
figure,imshow(rgb2),title('Imagen filtrada');
```

Otros filtros lineales especiales.

Algunos filtros mejoran las imágenes de manera especial. Por ejemplo, aplicando el filtro *Sobel* en una única dirección, se puede alcanzar un efecto de realce. La rutina *fspecial* produce varios tipos de filtros predefinidos. Después de crear un filtro con *fspecial*, se aplica a los datos de la imagen utilizando *filter2*. Este ejemplo aplica un filtro *Sobel* a la imagen *trees*.

```
load trees
I=ind2gray(X,map); %Imagen de intensidad double (rango de 0 a 1)
h=fspecial('sobel') %Observe como es un filtro detector de bordes horizontales
J=filter2(h,I);
figure, colormap(gray(128))
imagesc(J)
axis off
```



La imagen obtenida tras el filtro de *Sobel* contiene valores fuera del rango 0.0 a 1.0 de la imagen de partida; *imagesc* es por lo tanto la mejor elección para visualizarla, ya que hace corresponder los valores de intensidad al rango completo de salida. Otra opción es normalizar la imagen obtenida al rango 0.0 a 1.0 (double) y visualizarla entonces con *imshow*, tal y como se vio en la práctica anterior.

Los distintos filtros que se pueden obtener con la función *fspecial* son:

'gaussian' for a Gaussian lowpass filter
'sobel' for a Sobel horizontal edge-emphasizing filter
'prewitt' for a Prewitt horizontal edge-emphasizing filter
'laplacian' for a filter approximating the two-dimensional Laplacian operator
'log' for a Laplacian of Gaussian filter
'average' for an averaging filter
'unsharp' for an unsharp contrast enhancement filter

Filtrado de mediana.

El filtrado mediana (mediante la función *medfilt2*) es útil para eliminar valores de píxeles extremos. Se trata de un tipo de filtro no lineal útil eliminar el ruido '*salt & pepper*'.

Como ejemplo se añadirá ruido del tipo "*salt&peper*" a una imagen y posteriormente se filtrará mediante un filtro media y un filtro mediana para que se observe la diferencia.

```
I=im2double(imread('eight.tif'));
J=imnoise(I,'salt & pepper',0.02);  %Añade a imagen I ruido sal y pimienta de densidad
0.02
figure, imshow(I); title('imagen original ');
figure, imshow(J);title('imagen con ruido sal y pimienta');
K=filter2(fspecial('average',3),J); %Filtro J con filtro promedio en entorno de vecindad
3x3
L=medfilt2(J,[3 3]); %Filtro imagen J con filtro mediana en entorno de vecindad 3x3
```

```
figure, imshow(K); title('Resultado filtro media ');
figure, imshow(L); title('Resultado filtro mediana ');
```