Bloque 2: Operadores lineales

MATERIAL EXPERTOS	Experto 2: Diseño frecuencial de máscaras

Aproximaciones a la realización de operadores locales lineales

La aplicación de un operador sobre una imagen que denominaremos original ($\psi[n,m]$), da lugar a una nueva imagen que denominaremos procesada ($\theta[n,m]$), Un operador local toma como entrada el valor de un entorno de píxeles en el entorno de uno dado y genera un valor resultante en el píxel homólogo de la imagen resultante. Por simplicidad asumiremos que el entorno es rectangular, de dimensiones M', N', según indica la Figura 1.

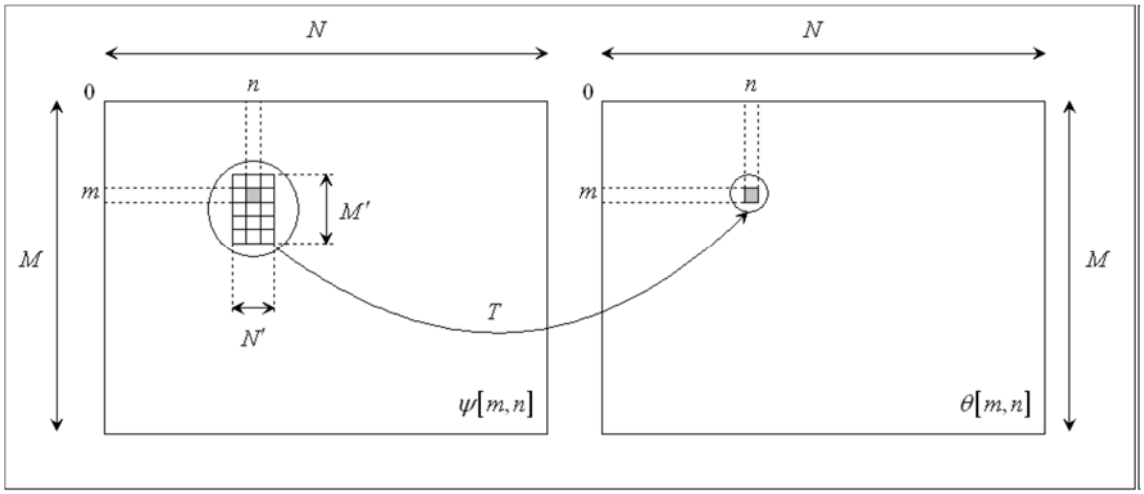

Figura 1: Esquema genérico de aplicación de un operador local

Según se ha visto en la parte teórica de la asignatura, si la operación que se efectúa sobre los píxeles del entorno es lineal, aplica la Teoría de Sistemas Lineales multidimensionales. MatLab ofrece herramientas específicas para esta situación.

Diseño frecuencial de máscaras

Como han visto en teoría, el diseño de filtros en frecuencia puede servir como banco de pruebas para el diseño de filtros espaciales. Asimismo, en teoría se desarrolló una metodología para este proceso que aquí incluimos y desarrollamos por medio de un ejemplo.

Metodología:

I. *Generación del filtro frecuencial con los parámetros deseados.*

II. *Obtención de su respuesta impulsiva.*

III. *Selección de rango –anchura de la máscara- y obtención de valores enteros en el rango deseado cuyo error de redondeo de lugar a mínimo error cuadrático medio.*

IV. *Ajuste de la respuesta de la máscara a una señal constante.*

Ejemplo: Diseño a partir de filtros ideales cuadrados centrados

I. *Generación del filtro frecuencial con los parámetros deseados.*

Genere un filtro paso bajo ideal en la región $0 \leq x \leq 1,\ 0 \leq y \leq 1$ con las siguientes características:

a. Frecuencia de muestreo en el retículo ortogonal: *fs* = 400 .

```
fs=400;
```

b. Retículo ortogonal de vectores: $\vec{v}_1 = (1/fs\,, 0), \vec{v}_2 = (1/fs\,, 0)$

Divida el retículo en *fs* +1 = 401 **muestras, es decir, conserve la última muestra de manera que el número de muestras sea impar en ambas dimensiones.**

```
v1=1/fs; v2=1/fs;
x=[0:v1:1];
y=[0:v2:1];
[X,Y]=meshgrid(x,y);
```

c. Posición central del filtro (en el centro de la imagen): $f_{0x} = 0.5, f_{0y} = 0.5$

```
f0x=0.5;
f0y=0.5;
Xc = (X - f0x);Yc = (Y - f0y);
```

d. Frecuencia de corte (horizontal y vertical): $D_0 = fs/4$

```
D0=fs/4;
% filtro paso bajo ideal con frecuencia de corte D0
f_filter =double(abs(Xc) <= D0/fs & abs(Yc) <= D0/fs );
```

II. *Obtención de su respuesta impulsiva.*

Calcule la respuesta al impulso del filtro paso bajo ideal definido Para ello recuerde mover primero el centro del filtro (frecuencia cero) al origen de la matriz y deshacer esta operación en la transformada inversa de Fourier obtenida mediante la función `ifft2`. Puede utilizar para ello las funciones `ifftshift` y `fftshift` de Matlab, es decir, si denominamos al filtro en frecuencia `f_filter`:

```
f_filter_d = ifftshift(f_filter);
h_d = ifft2(f_filter_d);
h = fftshift(h_d);
```

Compruebe que la respuesta al impulso obtenida *h*[*x*, *y*] es real en todo su rango de definición.

III. *Selección de rango –anchura de la máscara- y obtención de valores enteros en el rango deseado cuyo error de redondeo de lugar a mínimo error cuadrático medio.*

Primero localice la posición del centro *cx*,*cy* del filtro espacial *h*[*x*, *y*] . Note que dada la definición del filtro en frecuencia este valor puede obtenerse como:

$$c_x = 1 + \frac{fs}{2} \qquad c_y = 1 + \frac{fs}{2}$$

A continuación defina la anchura del filtro *w*, por ejemplo para obtener una máscara de filtro de *3x3*, fije *w* = 1 y obtenga la máscara de filtrado recortando la respuesta a partir de la expresión:

```
filter_mask=h(cy-w:cy+w,cx-w:cx+w);
```

A continuación obtenga la versión en número enteros de la máscara obtenida. Para ello:

a. Obtenga primero el mínimo valor en `filter_mask` y divida la máscara por ese valor, así el menor valor será uno.

```
      [m,j] = min(filter_mask(:));
filter_mask = filter_mask./m;
```

b. Defina el máximo factor de ponderación utilizable, para ello, note que el rango de la máscara tras este proceso deberá estará comprendido entre los 256 niveles: [- 128,128) permitiendo que la máscara contenga valores negativos. Así, el máximo factor de ponderación utilizable: `R` podrá ser aproximado mediante la expresión:

```
R = ceil(127./max(abs(filter_mask(:))));
```

c. Recorra mediante un bucle los enteros comprendidos entre 1 y R. Calcule la versión en números enteros de `filter_mask` resultante de aplicar la ponderación correspondiente en cada iteración del bucle y mida el error cuadrático medio respecto de la máscara original:

```
% búsqueda de la versión entera que minimiza el MSE
mse = Inf.*ones(1,R);
for j =1:R,
    dif = (filter_mask - double(round(filter_mask.*j)./j)).^2;
    mse(j) = mean(dif(:));
end
[~,j]=find(mse==min(mse),1);
filter_mask = round(filter_mask.*j);
```

IV. *Ajuste de la respuesta de la máscara a una señal constante.*

La obtención del factor de ponderación: *C* será determinante en el uso de la máscara diseñada. El resultado de la operación de filtrado habrá de ponderarse por este factor. Consulte con el experto 1 el método para calcular *C* una vez que `filter_mask` ha sido obtenida.