Bloque 2: Operadores lineales

MATERIAL EXPERTOS	**Experto 1: Uso de la función imfilter y valor de ponderación**

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.

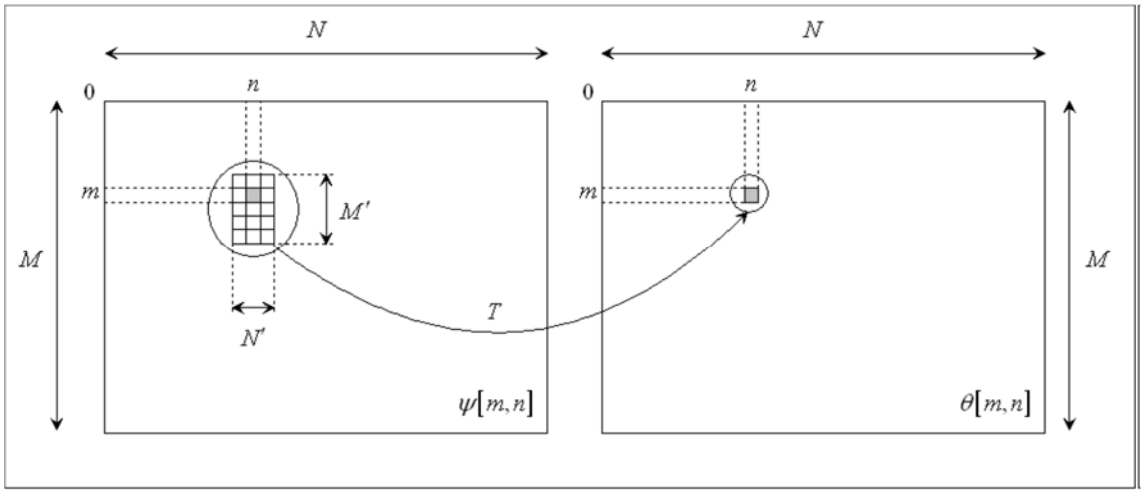

Figura 1: Esquema genérico de aplicación de un operador local

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.

Filtrado lineal

Si un operador local efectúa una transformación lineal, su comportamiento puede definirse completamente por su respuesta al impulso, $h[n,m]$. Según se ha visto en las clases de teoría, si la respuesta al impulso es rectangular y simétrica respecto de su origen (para lo cual M', N' han de ser impares) su aplicación sobre la imagen (es decir, la operación de convolución) puede efectuarse de manera sencilla aplicando la máscara $w[n,m] = h[-n,-m]$ sobre cada píxel de la imagen original, según muestra la Figura 2.

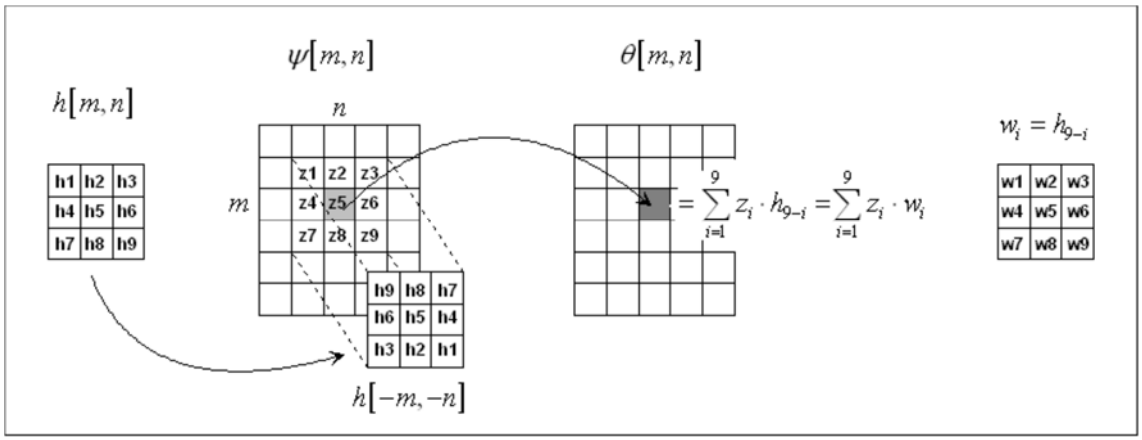

Figura 2: Esquema de aplicación de un operador local lineal

MatLab ofrece la función **`imfilter`** para llevar a cabo esta operación. Su modo de operación por defecto toma como parámetro la imagen original, **`ima`** y la máscara **`mask`** (una matriz de $M'x\ N'$) y arroja la imagen resultante, **`ima_res`** :

```
>> ima_res=imfilter(ima,mask);
```

La imagen **`ima_res`** resultante es del mismo tipo que la imagen original (**`double`**, **`uint8`**, **`uint16`**, etc.). Para evitar problemas de truncamiento, si la aplicación de la máscara hace que los píxeles de la imagen procesada puedan tener un rango de valores mayor que los de la imagen original, es conveniente convertir la imagen a **`double`** antes de operar.

Para comprobar su funcionamiento, cargue y visualice la imagen en escala de grises edificio_bw_512.bmp1.

Defina a continuación la siguiente máscara:

$$\mathbf{W} = \frac{1}{C}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & 6 & 4 & 1 & 4 & 6 & 7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

A partir del modo de aplicación mostrado en la Figura 2, indique el máximo valor, C, que puede arrojar este operador para una imagen con valores normalizados (valores en [0,1]): C = ¿?

Su inverso (1/C) es el valor que ha de ponderar al operador para garantizar que no se modifica el rango dinámico de la imagen.

Aplique esta máscara y la máscara rotada 90º (es decir $\mathbf{W}^T$) sobre la imagen dada y visualice simultáneamente las dos imágenes procesadas. Indique de forma cualitativa el efecto que produce cada una de estas dos máscaras.

Observe las bandas oscuras que aparecen en algunos extremos de las imágenes procesadas. Se deben al modo en que la función **`imfilter`** calcula el resultado cuando la máscara se aplica cerca de los extremos de la imagen: el comportamiento por defecto consiste en considerar que los píxeles que faltan tienen valor nulo.

Observe e intente explicar, la presencia y ausencia de las bandas oscuras en cada caso y su mayor o menor oscuridad.

Pruebe los otros tres modos predefinidos de operar en los extremos que tiene **`imfilter`**, observe los resultados e indique cuál le parece el más adecuado para esta máscara y porqué.

Filtros habituales

Suavizado

Una de las aplicaciones más comunes de los operadores LSI – y de los operadores locales en general- es el suavizado de imágenes. Como han visto en teoría, una máscara de promediado de orden 3 se define simplemente como:

$$mask = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Extracción de contornos

Durante este bloque se compararán tres de los métodos clásicos para la extracción de los contornos de una imagen: Roberts, Prewitt y Sobel. Los tres métodos pueden invocarse en MatLab como opciones de la función **`edge`** sin embargo, generalmente esta función retorna el resultado umbralizado, por lo que en este bloque partiremos de la definición de los filtros para obtener directamente la información de *intensidad*, *energía* o *probabilidad* de borde, mediante la combinación de dos operaciones de filtrado espacial por medio de la función **`imfilter`**.

Sea `I` la imagen a filtrar y `s` y `s2` los filtros de contornos, la *intensidad* de borde se calcula como:

```
FX = imfilter(I,s);
FY = imfilter(I,s2);
F = sqrt(FX.^2 + FY.^2);
```

Roberts.

Los filtros para la extracción de bordes por este método se definen como:

$$s_d = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \text{ y } s_{d'} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

Prewitt.

Los filtros para la extracción de bordes por este método se definen para el caso 3x3:

$$s_x = \frac{1}{6}\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix} \text{ y } s_y = \frac{1}{6}\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$$

Sobel

Los filtros para la extracción de bordes por este método se definen para el caso 3x3:

$$s_x = \frac{1}{8}\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \text{ y } s_y = \frac{1}{8}\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$