Bloque 2: Operadores morfológicos

MATERIAL EXPERTOS	**Experto 2: Dilatación y erosión por desplazamiento de la señal.**

Aproximaciones a la realización de filtros morfológicos

La aplicación de filtros morfológicos es una aproximación no lineal, a diferencia de las vistas en las prácticas anteriores. Análogamente al caso de los operadores lineales, los operadores morfológicos están caracterizados por una máscara b, que aquí denominaremos *kernel*. En la mayoría de los casos, el *kernel* suele ser plano, es decir con un rango limitado a dos valores: $\{0,-\infty\}$. En la práctica estos valores se representan por la pareja $\{1,0\}$ respectivamente. La siguiente figura muestra ejemplos de *kernels* con diversas formas y simetrías (el origen de coordenadas del *kernel* se ha marcado con un punto negro). Observe que todos los elementos salvo el **6** son simétricos respecto del origen y que todos los elementos salvo el **5** son conexos.

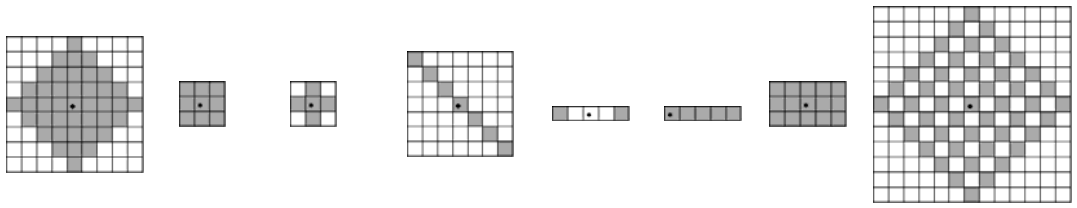

Figura 1: Ejemplos de elementos estructurantes (*kernels*) planos. Los valores oscuros representan ceros y los valores claros representan $-\infty$. El punto negro indica el origen de coordenadas.

Según las explicaciones de la parte teórica de la asignatura sabemos, por ejemplo, que la operación de dilatación con cualquiera de estos *kernels* aplicada sobre un píxel de una imagen consiste en situar el origen del *kernel* sobre el píxel considerado y hallar el máximo de los valores del entorno de dicho píxel que coinciden con ceros (valor oscuro o unidad en binario) del *kernel* utilizado.

Dilatación y erosión por desplazamiento de la señal

Según se ha visto en las explicaciones teóricas, si un elemento estructurante es sencillo expresarlo como una superposición de funciones básicas (algo que siempre ocurre con los elementos planos que habitualmente se usan), es posible operar con las expresiones de la dilatación y de la erosión para obtener una nueva expresión con una interpretación operativa de gran utilidad.

Efectivamente, para el caso de señales unidimensionales, si se tiene un elemento:

$$b[n] = \bigvee \left(\delta_L [n-2], \delta_L [n-1], \delta_L [n], \delta_L [n+1] \right)$$

Note que este elemento corresponde a la máscara [1 1 <u>1</u> 1], donde _ *indica la posición central.*

, la respuesta (dilatación o erosión) a una señal de entrada es posible expresarla según:

$$y[n] = x[n] \oplus b[n] = \bigvee \left(x[n+1], x[n+2], x[n], x[n-1] \right)$$
$$y[n] = x[n] \ominus b[n] = \bigwedge \left(x[n+1], x[n], x[n-1], x[n-2] \right)$$

, es decir, como el máximo (supremo) o mínimo (ínfimo) respectivamente de versiones de la señal desplazadas siguiendo el mismo patrón de los ceros del elemento estructurante.

En esta práctica aprovecharemos la observación anterior para realizar una implementación alternativa, y mucho más eficiente, de las funciones que está estudiando el experto 1 **`imfilter_dilate`** e **`imfilter_erode`**, a las que se denominará **`imfilter_dilateD`** e **`imfilter_erodeD`**.

Observe que, según lo explicado más arriba, la única diferencia entre estas dos últimas funciones está en el sentido del desplazamiento y en la operación (máximo o mínimo) que se realiza. En los ejercicios guía deberá implementar ambas funciones desde cero (es decir, no se inspire en **`imfilter_dilate`** e **`imfilter_erode`**) teniendo en cuenta para ello las siguientes consideraciones:

- Se supone que el punto de aplicación (origen) del operador es su centro.
- La función deberá ser genérica, permitiendo el uso de cualquier elemento estructurante, tanto en dimensión como en estructura interna (organización de ceros y unos).
- No es necesario crear simultáneamente tantas imágenes desplazadas como ‘1’s tenga el elemento estructurante (lo que exigiría elevados recursos de memoria). Basta con ir creando imágenes desplazadas e ir calculando el máximo (o mínimo) con el anterior resultado parcial.
    - Para generar imágenes desplazadas MatLab dispone de la función **`circshift`** que desplaza circularmente la matriz de entrada (**`ima`**) dependiendo de los valores del vector de desplazamiento (**`shift_vector`**). Este vector está compuesto de dos valores que indican el número de desplazamientos a realizar en las filas y columnas:

      ```
      >> ima_despl = circshift(ima, shift_vector);
      ```

- La función deberá retornar el valor correcto de la operación sólo en los píxeles que sea posible, es decir, la función deberá devolver una imagen procesada donde el marco de la imagen (los píxeles cerca de los extremos, definidos en función del tamaño del elemento estructurante) sean iguales a los de la imagen original.