Procesamiento de Imágenes

Ingeniería Biomédica

Unidad 6: Procesamiento Morfológico

- Conceptos básicos de teoría de conjuntos
- Procesamiento morfológico de imágenes binarias
- Erosión y dilatación
- Apertura (erosión + dilatación) y Cierre (dilatación+erosión)
- Transformación “Hit or Miss”
- Aplicaciones:
  - Extracción de bordes (frontera)
  - Relleno de regiones (filling hole)
  - Extracción de componentes conectados
  - Adelgazamiento (thinning) y engrosamiento (thickening)
  - Esqueletonización (skeleton)
  - Poda (pruning)
  - Etiquetado (label)
- Procesamiento morfológico de imágenes en escala de gris

- Morfología matemática: se basa en operaciones enmarcadas en la teoría de conjuntos.
- Operaciones primarias: erosión y dilatación.
- Combinando operaciones primarias obtenemos: apertura y cierre.

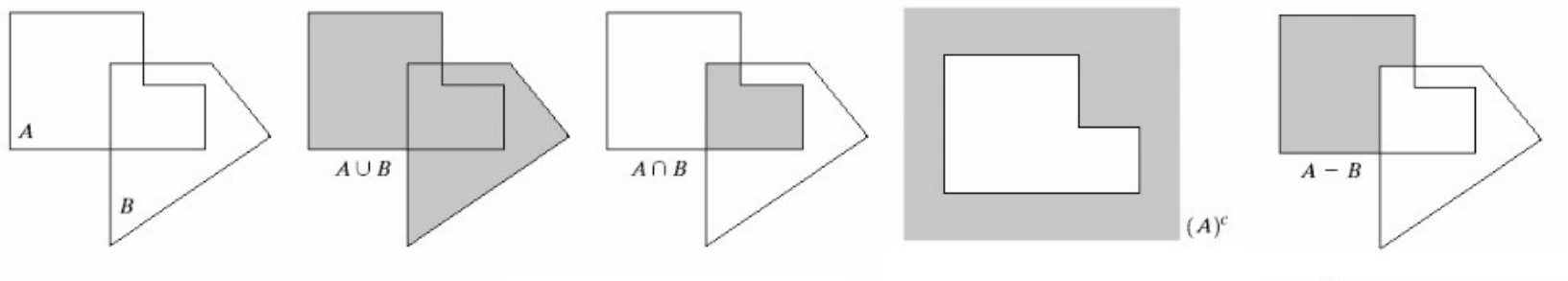
Aplicaciones: suprimir ruido, simplificar formas, extraer esqueleto, detectar objetos, determinar fronteras, adelgazar/engrosar, rellenar regiones, describir objetos (área, perímetro, longitud, etc.)

$$\textit{unión}: C = A \cup B = \{(x,y) | (x,y) \in A \textbf{ or } (x,y) \in B \textbf{ or } (x,y) \in (A \textbf{ and } B)\}$$

$$\textit{intersección}: C = A \cap B = \{(x,y) | (x,y) \in (A \textbf{ and } B)\}$$

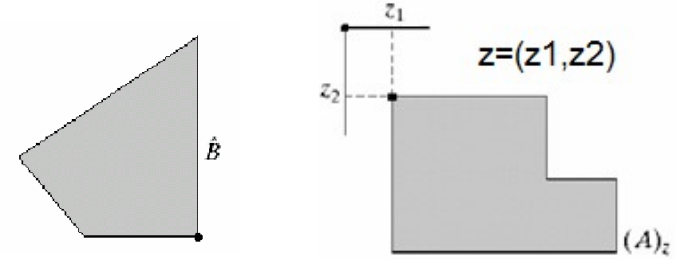
$$\textit{complemento}: (A)^c = \{(x,y) | (x,y) \notin A\}$$

$$\textit{diferencia}: C = A - B = \{(x,y) | (x,y) \in A \textit{ and } \notin B\}$$



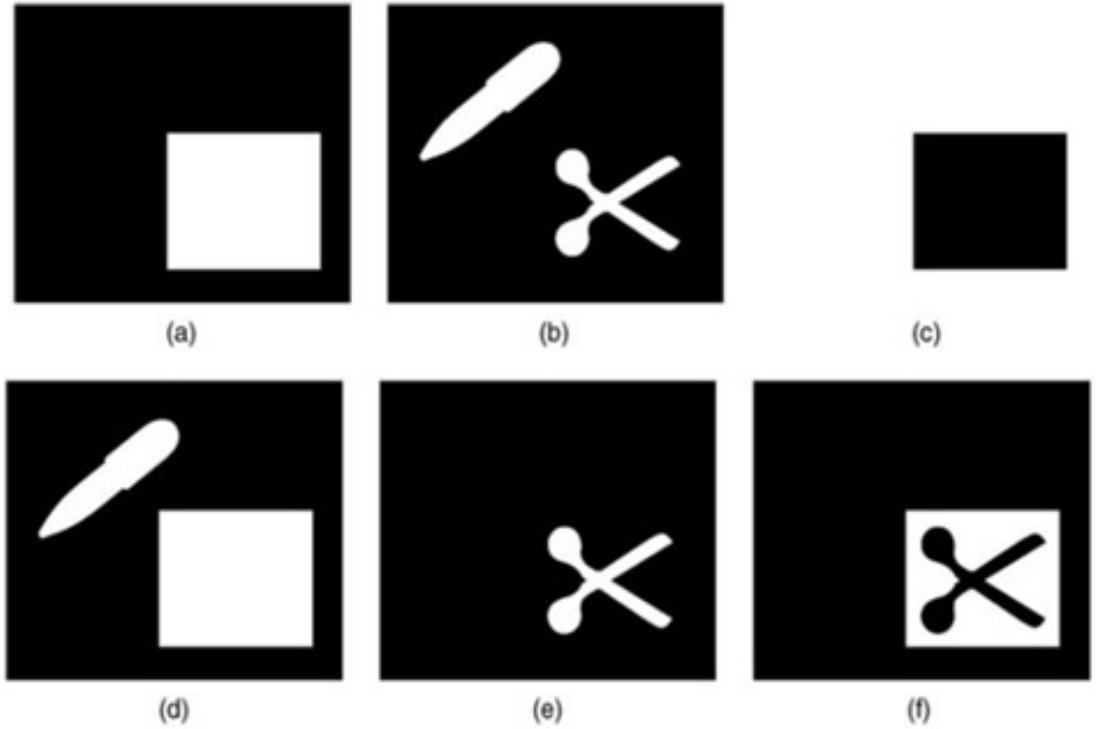
$$\textit{reflexión}: \hat{B} = \{(x,y) | (x,y) = -b, para\ b \in B$$

$$\textit{traslación}: (A)_z = \{c | c = a + z\ para\ a \in A$$



El punto negro denota el origen del conjunto (punto de referencia definido por el usuario)

Convención: blanco (valor 1) *foreground pixels (objeto)*, negro (valor 0) *background pixels (fondo)*.



(a) imagen binaria A
(b) imagen binaria B
(c) complemento de A (A^c)
(d) unión ($A \cup B$)
(e) intersección ($A \cap B$)
(f) diferencia (A-B)

Operaciones lógicas equivalentes a las operaciones con conjuntos:

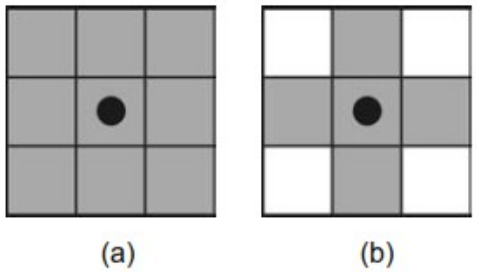
$$A \cap B \equiv A \,\&\, B$$
$$A \cup B \equiv A \mid B$$
$$A^c \equiv \sim A$$
$$A - B \equiv A \& \sim B$$

Elemento Estructural (SE: Structural Element)

Matriz cuyo tamaño y valores impacta sobre los resultados de aplicar un operador morfológico.

Usamos un punto negro para representar el origen del SE (punto de referencia), pixel grises para los valores True y blancos para los False.

En el ejemplo, los pixeles grises son miembros del SE, los blancos son usados como padding.



SE (a) cuadrado, (b) cruz.
(ver función *strel*)

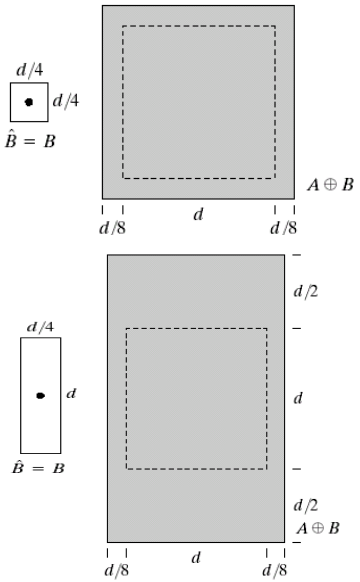
La función ***strel*** permite construir elementos estructurales (*SE*) con una variedad de formas y tamaños. Su sintaxis es:

SE = strel (forma, parámetros)

SE = strel ('diamond', R)	Forma de diamante. ***R*** es la distancia del origen a los puntos extremos del diamante.
SE = strel ('disk', R)	Forma de disco. ***R*** *es el radio.*
SE = strel ('line', len, deg)	Forma de línea. Longitud ***len,*** ángulo ***deg*** [grados] medido desde el eje x en sentido antihorario.
SE = strel ('rectangle',MN)	Forma de rectángulo. ***MN*** es el tamaño (vector de dos elementos enteros positivos).
SE = strel ('square',W)	Forma de cuadrado. ***W*** es el ancho en píxeles (debe ser entero positivo).
SE = strel ('arbitrary',NHOOD)	Forma arbitraria. **NHOOD** es una matriz de 0 y 1 que especifica la forma.

Ejercicio: Generar diferentes SE empleando la función ***strel.***

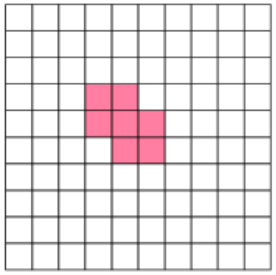
Dilatación

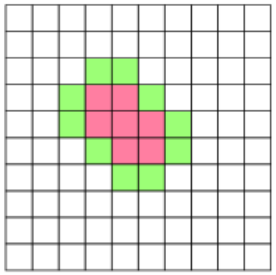


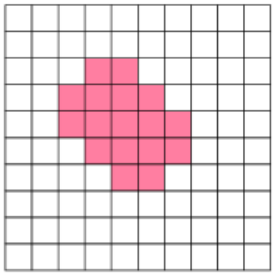
- Operación morfológica cuyo efecto es hacer crecer (growing) o espesar/engrosar (thickening) los objetos en una imagen.
- La extensión y dirección del crecimiento estará controlada por el tamaño y la forma del elemento estructural.
- Matemáticamente la dilatación se define:

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\}$$

"Al menos un elemento del SE (además del origen) está contenido en el objeto "



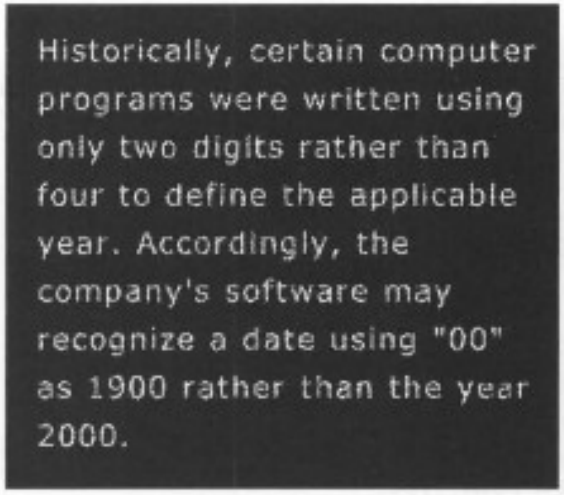


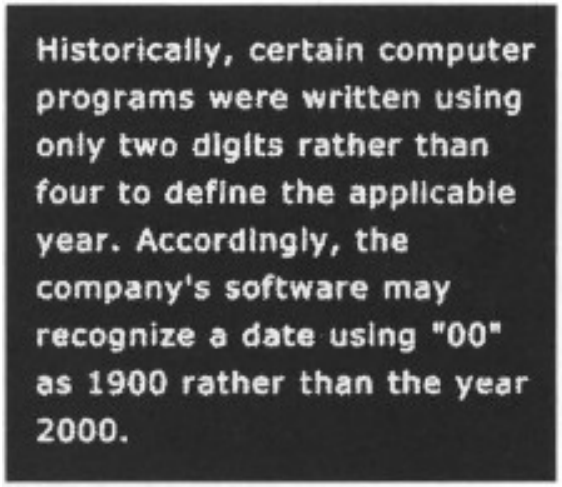


En Matlab usamos el comando imdilate, el cual recibe dos parámetros: la imagen (A) y un SE (B):

$$C = imdilate\,(A, B)$$

$$B = \begin{bmatrix} 0 & 1 & 0 \\ 1 & \boxed{1} & 1 \\ 0 & 1 & 0 \end{bmatrix}$$





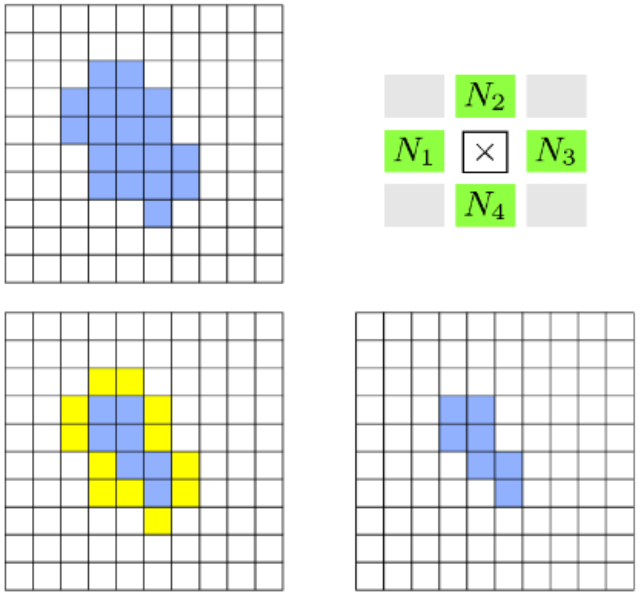
Ejercicio: Aplicar el operador dilatación a una imagen.

Erosión

- Su efecto es encoger (shrinking) o adelgazar (thinning) los objetos en una imagen.
- La dirección y extensión del adelgazamiento se controla con la forma y el tamaño del SE. Los objetos de menor tamaño que el SE desaparecen.
- Matemáticamente la erosión de A por B se define como:

$$A \ominus B = \{z \mid (\hat{B})_z \cap A^c = \emptyset\}$$

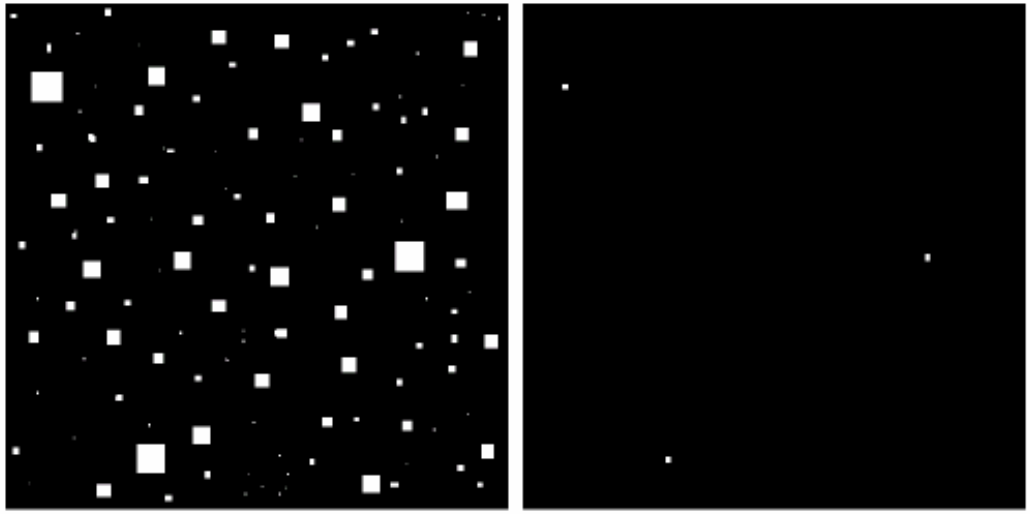
"Todo el SE debe estar contenido dentro del objeto"



Ejercicio: Aplicar el operador erosión a una imagen.

La erosión se realiza con el comando imerode.

$$C = imerode(A, B)$$



Izq) Imagen conformada por cuadrados de diferentes tamaños. Der) Resultado de aplicar la erosión con un SE cuadrado de < tamaño que el más grande de los cuadraditos y mayor que el resto.

Apertura

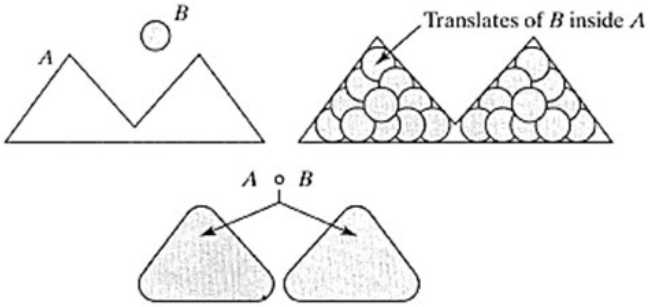
Erosión seguida de una dilatación: $A \circ B = (A \ominus B) \oplus B$

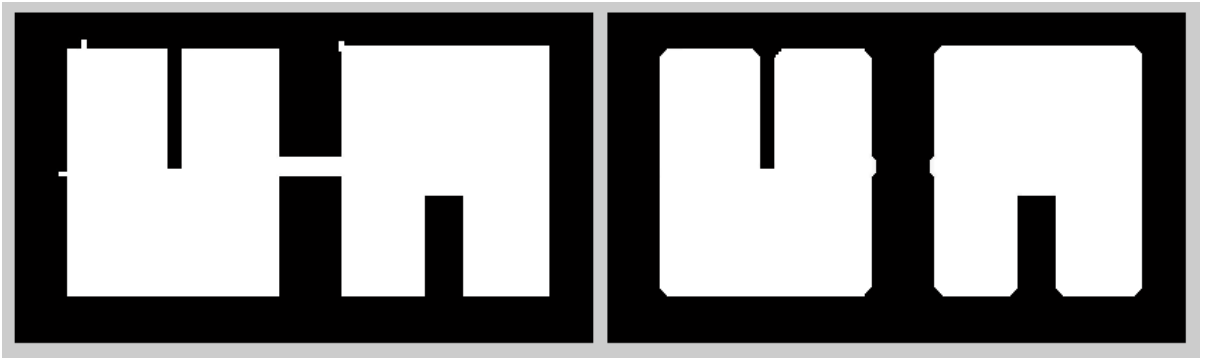
C = imopen (A, B)

- La apertura es ***idempotente,*** una vez que a una imagen se le aplicó la apertura con un SE, subsecuentes aplicaciones del mismo SE no causará efecto alguno.

- Es la unión de todas las traslaciones de B que ajustan enteramente dentro de A.

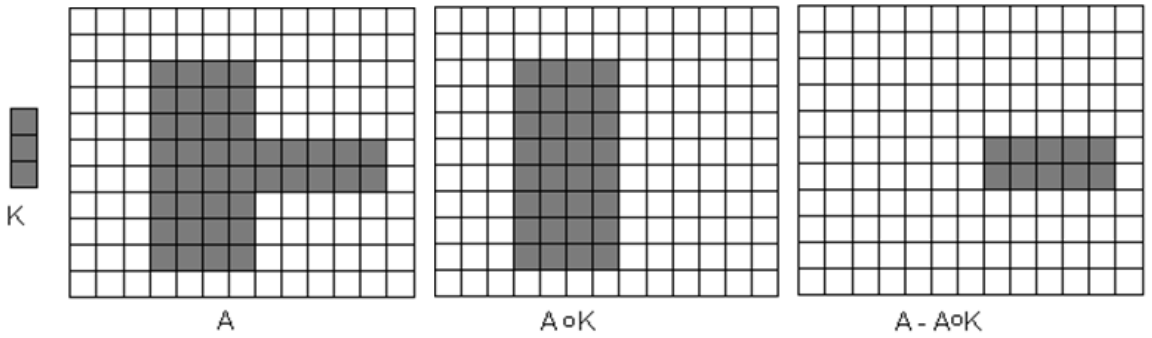
- Aplicación: suavizar los contornos de los objetos y eliminar pequeñas salientes (*remueve pequeñas regiones-1*). Rompe uniones angostas (zonas de un objeto que sean "más estrechas" que el elemento estructural).





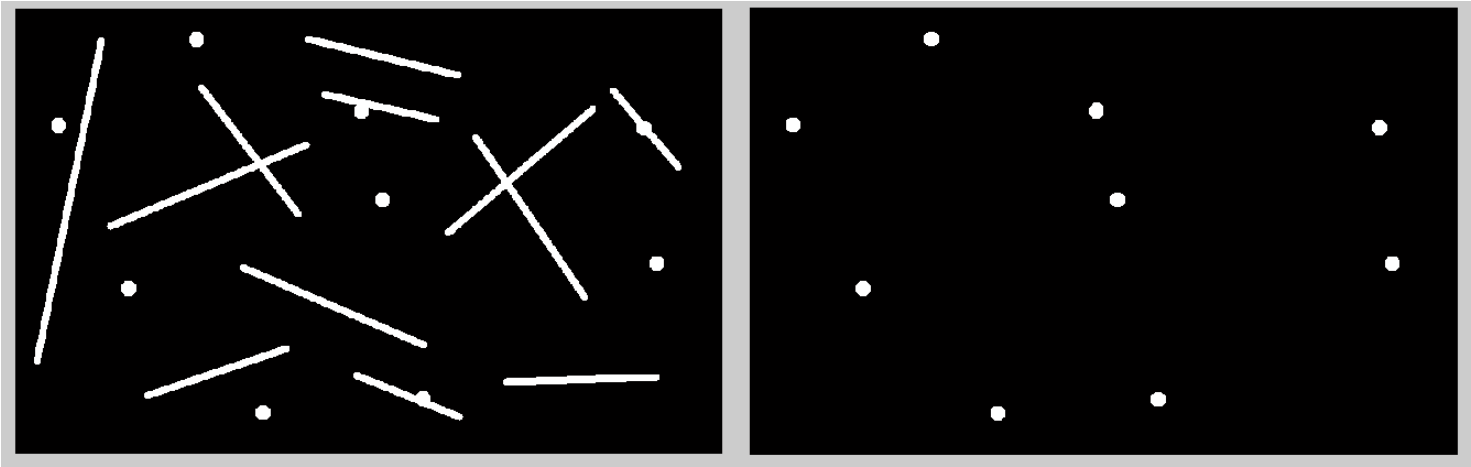
- Usando la apertura es posible descomponer objetos.

Ej. podemos separar ambas partes, realizando la apertura de A por K y luego restamos A y el resultado anterior.



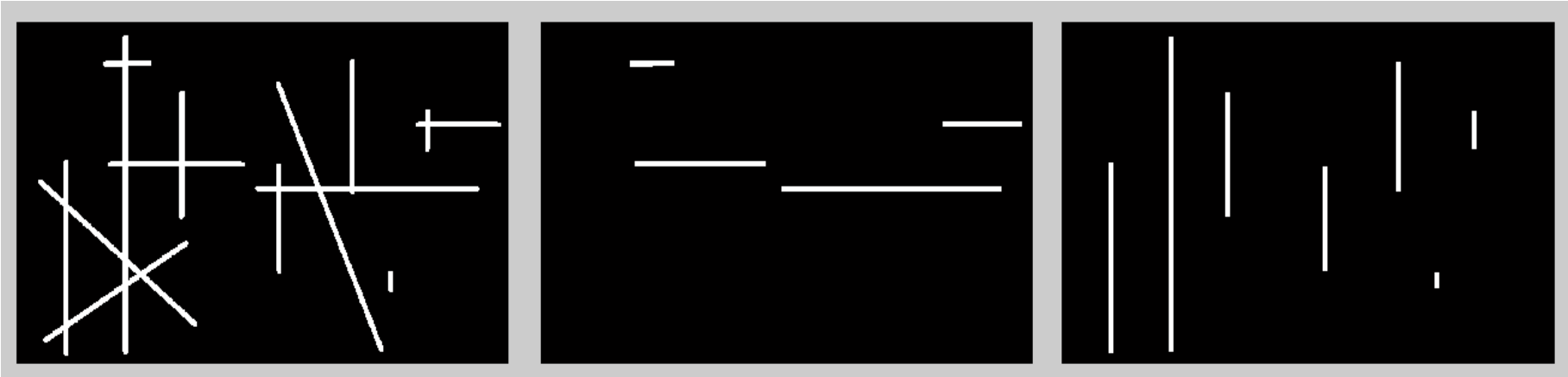
Aplicaciones de la apertura

Ejemplo Nº1:



- Objetivo: separar los círculos de las líneas, y contar cuantos círculos hay.
- Solución: apertura con un SE en forma de disco de diámetro apropiado

Ejemplo Nº2:



- Objetivo: separar las líneas verticales de las horizontales (eliminar las diagonales).
- Solución: apertura morfológica con un SE rectangular de dimensiones apropiadas

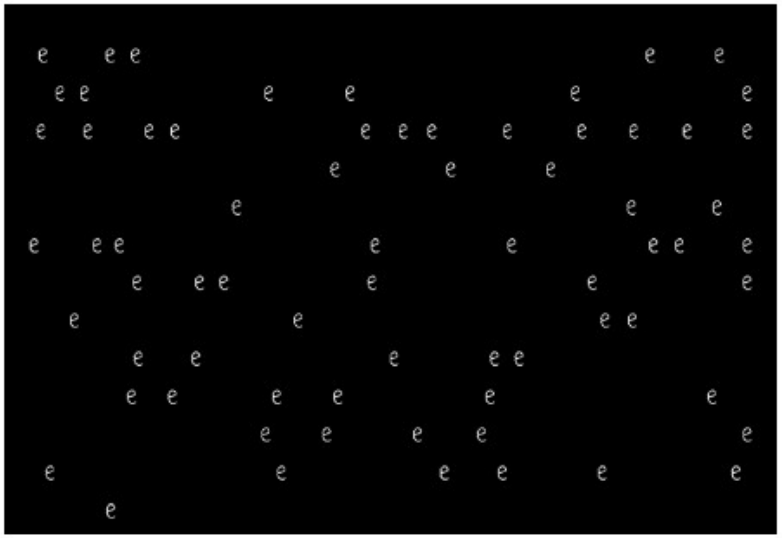
Ejemplo Nº3:

Objetivo: detectar el carácter “e” dentro del texto.
Solución: apertura del complemento de la imagen (quiero fondo negro) con un SE apropiado

Obs: si la imagen es binaria, el SE es plano, si la imagen es en escala de grises, el SE no es plano

INTEREST-POINT DETECTION

Feature extraction typically starts by finding the salient interest points in the image. For robust image matching, we desire interest points to be repeatable under perspective transformations (or, at least, scale changes, rotation, and translation) and real-world lighting variations. An example of feature extraction is illustrated in Figure 3. To achieve scale invariance, interest points are typically computed at multiple scales using an image pyramid [15]. To achieve rotation invariance, the patch around each interest point is canonically oriented in the direction of the dominant gradient. Illumination changes are compensated by normalizing the mean and standard deviation of the pixels of the gray values within each patch [16].

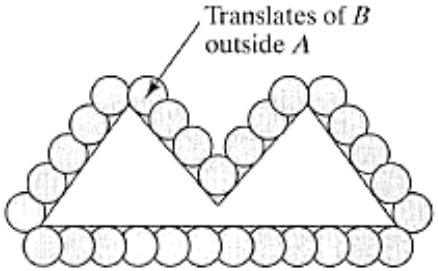


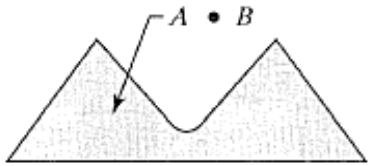
Cierre

Dilatación seguida de una erosión. $A \cdot B = (A \oplus B) \ominus B$

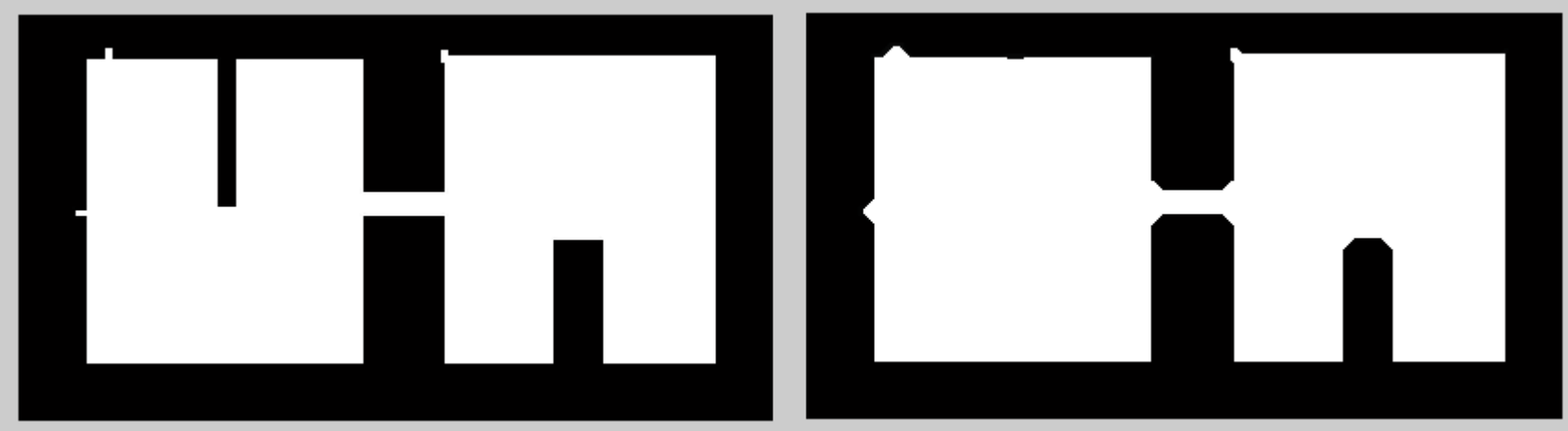
C = imclose (A, B)

- Equivale al complemento de la unión de todas las traslaciones del SE que no se superponen con los objetos de la imagen.
- Al igual que la apertura, es una operación *idempotente*

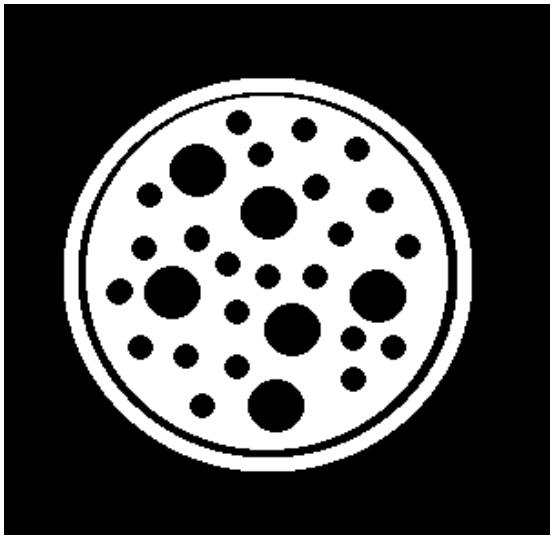




- Aplicación: suavizar contornos, rellenar agujeros pequeños, fusionar rajaduras estrechas (*remueve pequeñas regiones-0*) sin cambiar el tamaño de los objetos (como lo haría la dilatación).

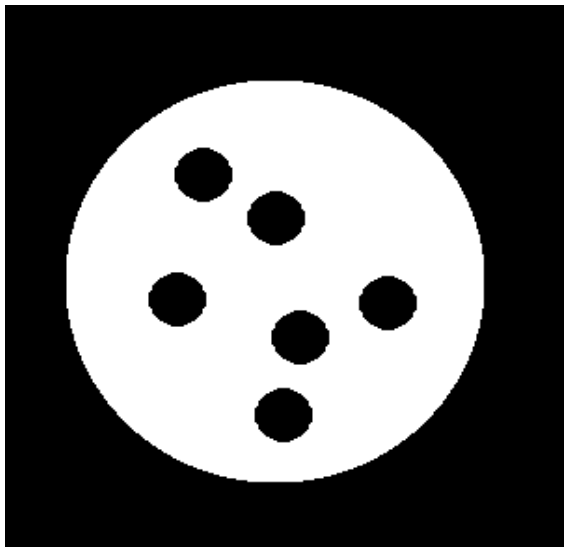


Ejemplo Nº4:



Objetivo: eliminar los agujeros pequeños.

Solución: cierre con un SE en forma de disco con un diámetro mayor que los agujeros más pequeños, pero más pequeño que los agujeros grandes



Ejercicio: Aplicar el operador apertura y cierre a una imagen, y observar el efecto que produce.

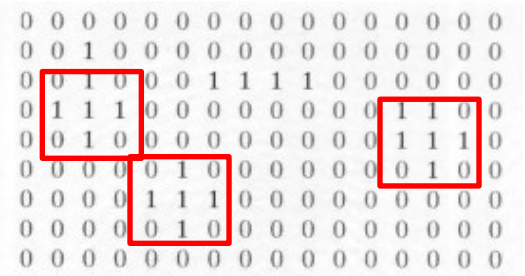
Transformación morfológica Hit-or-Miss

Herramienta básica para la detección de formas. Permite buscar una determinada configuración de píxeles B/N en la imagen. Se define en términos de dos SE como:

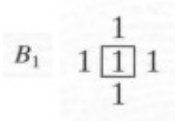
$$A \otimes B = (A \ominus B_1) \cap (A^c \ominus B_2)$$

C = bwhitmiss (A, B1, B2)

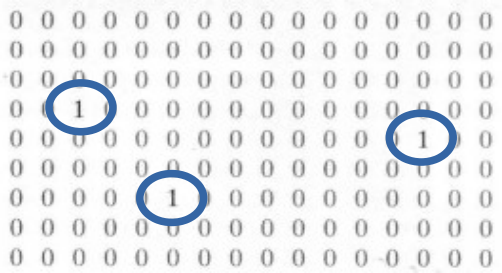
- La imagen resultante consistirá en todas las ubicaciones de A que coincidan (match) con los píxeles de B1 (hit) y que no coincidan (ni siquiera un pixel) con B2 (miss).

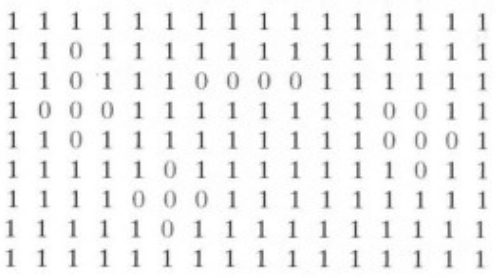


B_1

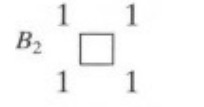


$(A \ominus B_1)$

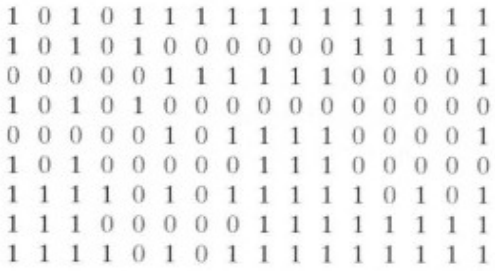


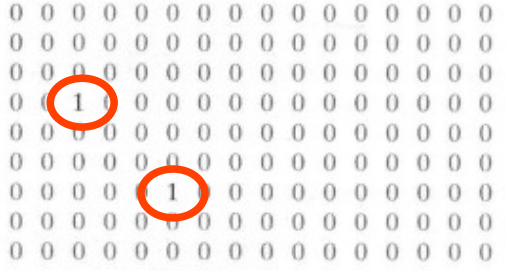


B_2



$(A^c \ominus B_2)$





Alternativa: C = bwhitmiss (A, B)

siendo B=matriz intervalo

- B = 1 donde B1 = 1
- B = -1 donde B2 = 1
- B = 0 en el resto

$$B = \begin{bmatrix} -1 & 1 & -1 \\ 1 & 1 & 1 \\ -1 & 1 & -1 \end{bmatrix}$$

Ejercicio: Aplique la transformación Hit-or-Miss a una imagen binaria, especificando un par de elementos estructurales (B1 y B2) de tal forma que permitan detectar un cierto patrón.

Transformación Hit-or-Miss usando Look-up Tables

- Cuando los SE son pequeños, una manera más rápida de computar esta transformación es usando una LUT (look-up table).

- Se calcula cada configuración posible de la vecindad y se almacenan los resultados en una tabla, que será utilizada posteriormente.

- Para una vecindad 3x3 existen 2^9=512 diferentes configuraciones posibles.

- Para utilizar la LUT debemos asignar un *único índice* a cada posible configuración. Esto lo podemos hacer multiplicando cada configuración 3x3 por la matriz:

$$\begin{bmatrix} 1 & 8 & 64 \\ 2 & 16 & 128 \\ 4 & 32 & 256 \end{bmatrix}$$

- Luego se suman todos los productos. Este procedimiento asigna un único valor en el rango [0, 511] a cada configuración de la vecindad 3x3.

Por ejemplo, para la vecindad:

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

el valor será: 1 (1) + 2 (1) + 4 (1) + 8 (1) + 16 (0) + 32 (0) + 64 (0) + 128 (1) + 246 (1) = 399

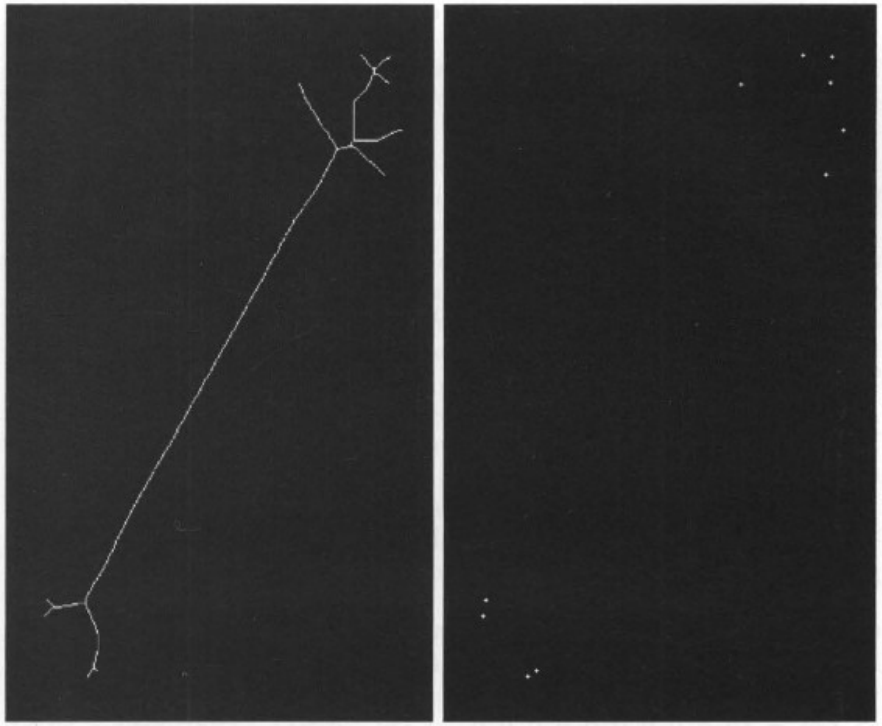
- El comando makelut construye la LUT basada en una función suministrada por el usuario y el comando applylut procesa la imagen binaria usando esta LUT.
- Para usar makelut es necesario escribir una función que acepte una matriz binaria 3x3 y devuelva un único valor (cero o uno).
- Makelut llamará 512 veces a la función provista por el usuario, pasando cada vez una configuración 3x3 diferente, y retornando todos los resultados en un vector de 512 elementos
- Definimos puntos extremos a aquellos píxeles cuya vecindad se ajusta (match) a cualquiera de las siguientes matrices intervalo hit-or-miss (y sus rotaciones 90º): (ver endpoint_fcn.m)

```
lut = makelut(@endpoint_fcn, 3);
g = applylut(f,lut);
```

```
function is_end_point = endpoint_fcn(nhood)
interval1=[0 1 0;-1 1 -1; -1 -1 -1];
interval2=[1  -1 -1; -1 1 -1;-1 -1 -1];

for k = 1:4
    % rot90(A, k) rota A 90° k veces
    C = bwhitmiss(nhood, rot90(interval1, k));
    D = bwhitmiss(nhood, rot90(interval2, k));

    if (C (2,2) == 1) || (D (2,2) == 1)
% La vecindad del pixel matchea
% una configuración de punto extremo
        is_end_point = true;
        return;
    end
end
is_end_point = false;
```



Ejercicio: Calcular los puntos extremos usando la transformación hit-or-miss con LUT, aplicada a una imagen binaria previamente esqueletonizada.

Operador bwmorph

Para efectuar operaciones morfológicas basadas en combinación de dilataciones, erosiones y LUT, podemos usar el siguiente comando:

O = bwmorph (I, operación, n)

I: imagen de entrada (binaria)

operación: especifica el tipo de operación deseada

n: número de veces que se repite (default n=1)

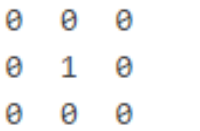
operaciones

'bothat'	'erode'	'shrink'
'bridge'	'fill'	'skel'
'clean'	'hbreak'	'spur'
'close'	'majority'	'thicken'
'diag'	'open'	'thin'
'dilate'	'remove'	'tophat'

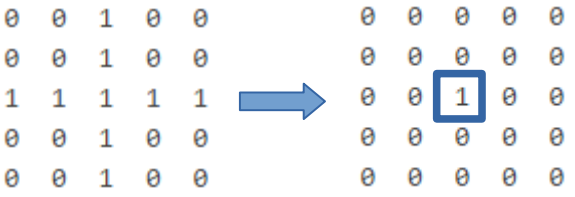
'bothat': sustrae la imagen de entrada de su cierre

'bridge' conecta píxeles separados por un gap (huevo, vacío) de un único pixel

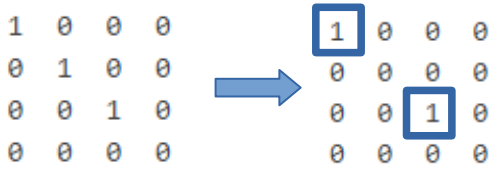
'clean': remueve pixeles aislados (un 1 rodeado de 0s)



'branchpoints': encuentra los puntos de ramificación (branch) de un esqueleto



‘endpoints’: encuentra los puntos extremos de un esqueleto.



‘remove’: remueve pixeles interiores

‘shrink’: encoge, contrae, reduce objetos sin agujero a puntos y objetos con agujeros a anillos.

‘skel’: esqueletoniza una imagen

‘spur’: remueve puntos extremos (endpoint) de líneas sin remover completamente pequeños objetos

‘thin’: reducen los objetos sin agujeros a un trazo de un pixel (si N=inf).

‘majority’: setea un pixel a 1 si 5 o más pixeles en una vecindad 3x3 son 1s, en otro caso vale 0.

‘close’: cierre 3x3

‘open’: apertura 3x3

‘dilate’: dilatación 3x3

‘erode’: erosión 3x3

‘fill’: rellena agujeros de único pixel (pixel 0 rodeado de 1s)

‘bothat’: sustrae la imagen de entrada de su cierre

‘hbreak’: remueve píxeles conectados con forma de H

Huella digital | Adelgazo una vez | Adelgazo nuevamente

G = bwmorph (F, ‘skel’, inf)

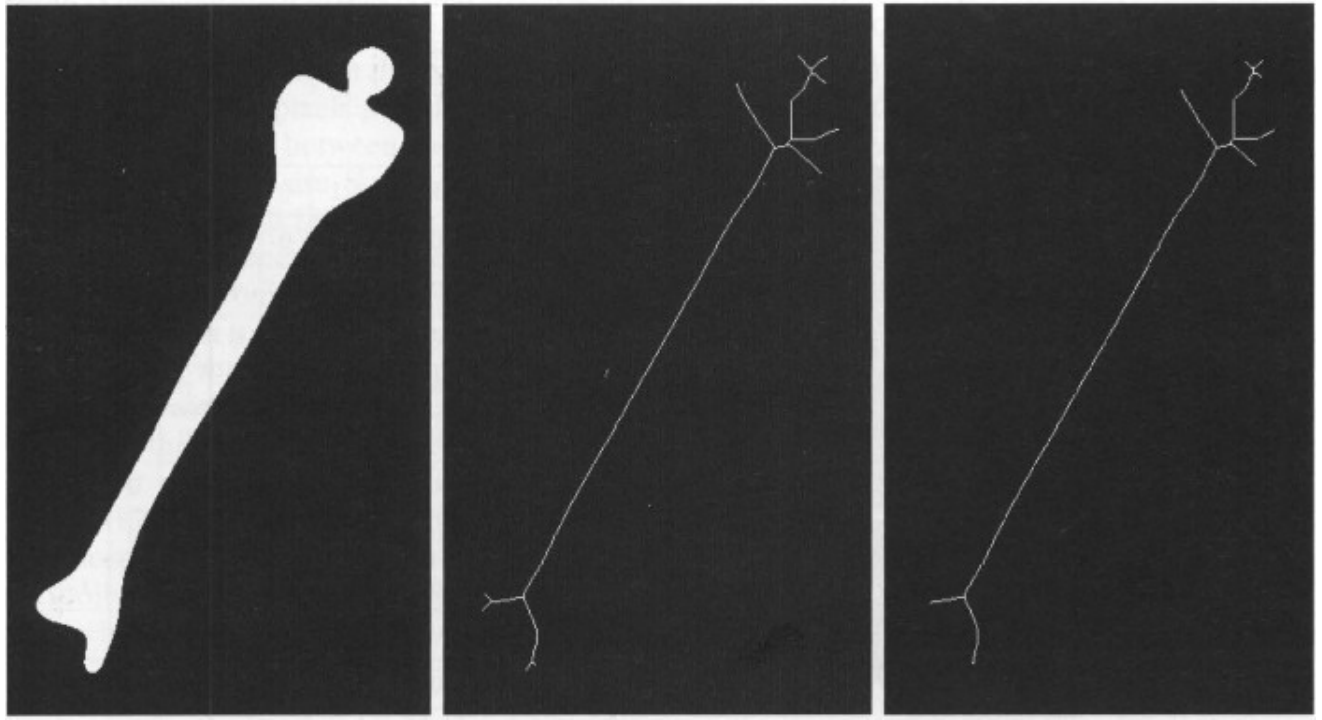


Imagen de un hueso | Esqueleto | Poda (pruning)

Ejercicio: Aplicar diferentes procesamientos morfológicos utilizando el comando bwmorph.

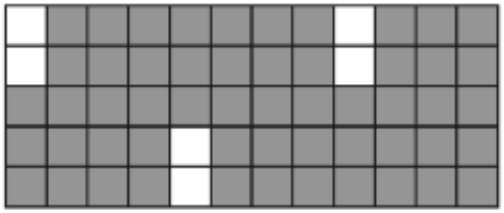
Aplicaciones de la dilatación y la erosión

❖ Extracción de la frontera

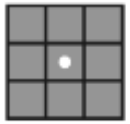
- La frontera de un conjunto A se puede obtener primero erosionando A por un SE apropiado (B) y realizando posteriormente la diferencia entre A y dicha erosión:

$$frontera\ de\ A = A - (A \ominus B)$$

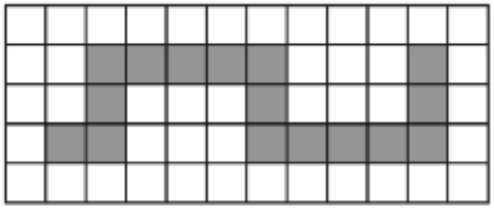
- El SE más usado es el cuadrado 3x3. Al incrementar la dimensión del SE (ej. 5x5) aumenta el grosor de la frontera.



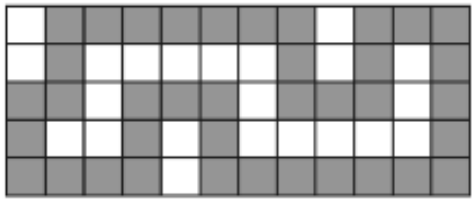
A



B



$A \ominus B$



$A - (A \ominus B)$

(explorar el uso de la función bwperim)

Ejercicio: Determinar la frontera de los objetos en una imagen, usando la formula y el comando bwperim.

❖ Rellenado de regiones (agujeros):

La función *imfill* realiza el relleno de agujeros, y su sintaxis es:

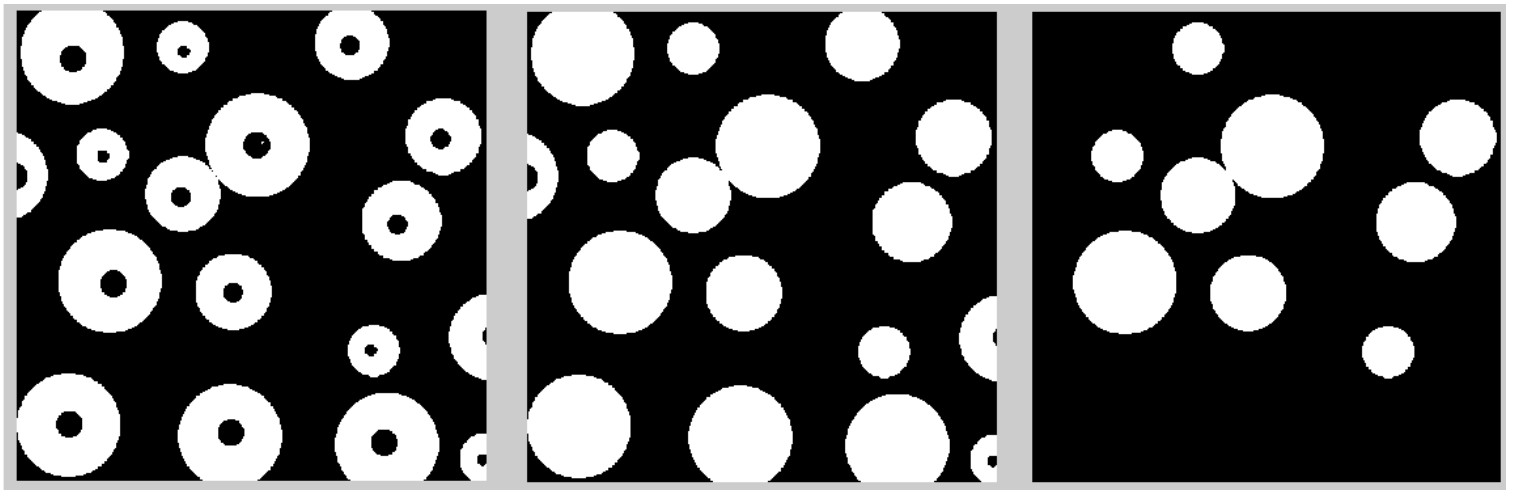
$$O = imfill\ (I,\ 'holes')$$

❖ Eliminación de objetos que tocan los bordes de la imagen:

Podemos remover objetos que tocan los bordes de la imagen usando el comando ***imclearborder***, cuya sintaxis es:

$$O = imclearborder\ (I,\ conn)$$

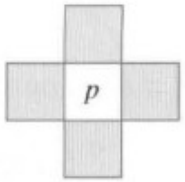
El parámetro *conn* puede ser 4 u 8 (default).



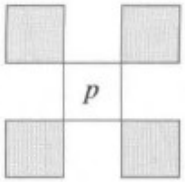
Ejercicio: Seleccione una imagen binaria que tenga objetos con agujeros, y aplíquele el comando **imfill** para rellenarlos. Pruebe también el comando **imclearborder** para eliminar objetos tocando los bordes de la imagen.

Etiquetado de componentes conectados

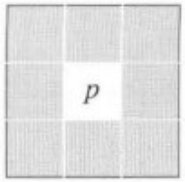
- Para localizar y operar con objetos (componentes conectados) debemos primero definir algunos términos:
  - Un pixel p de coordenadas (x,y) puede ser parte de una vecindad $N_4(p)$, $N_D(p)$ o $N_8(p)$.
  - Dos píxeles p y q son "4-adyacentes" si q pertenece $N_4(p)$. Similarmente, p y q son "8-adyacentes" si q pertenece a la vecindad $N_8(p)$
  - Un camino (path) entre p_1 y p_n es una secuencia de píxeles $p_1, p_2, p_3, \ldots, p_n$ tal que p_k es adyacente a p_{k+1}, con $1 <= k < n$. El camino puede ser 4-conectados u 8-conectados, dependiendo del tipo de adyacencia.



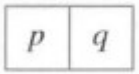
$N_4(p)$



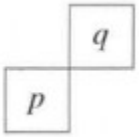
$N_D(p)$



$N_8(p)$



p y q son 4-adyacentes y 8-adyacentes



p y q son 8-adyacentes pero no 4-adyacentes

0	0	0	0	0
0	0	1	1	1
0	0	1	0	0
1	1	1	0	0
0	0	0	0	0

Los pixeles sombreados están 4 y 8-conectados

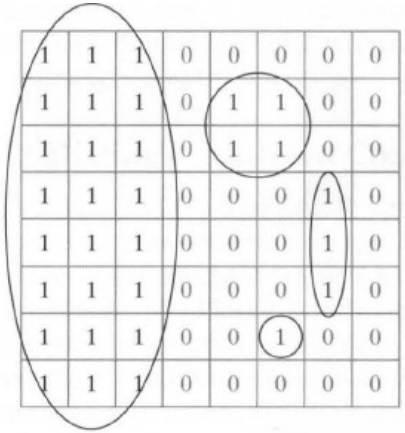
0	0	0	0	0
0	0	1	1	1
0	0	1	0	0
1	1	0	0	0
0	0	0	0	0

Pixeles sombreados están 8-conectados pero no 4-conectados

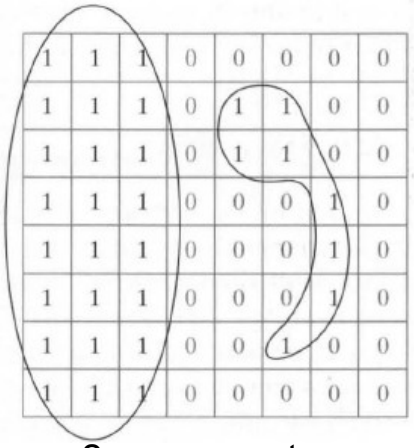
La función bwlabel computa todos los componentes conectados en la imagen binaria. Su sintaxis es:

[L, num] = bwlabel (BW, conn)

Donde *BW* es la imagen binaria de entrada, *conn* especifica la conectividad deseada (4 u 8), *L* es la matriz de etiquetas (label) y *num* (opcional) el número total de componentes conectados. El parámetro *conn* por default = 8.



4 componentes
4-conectados



2 componentes
8-conectados

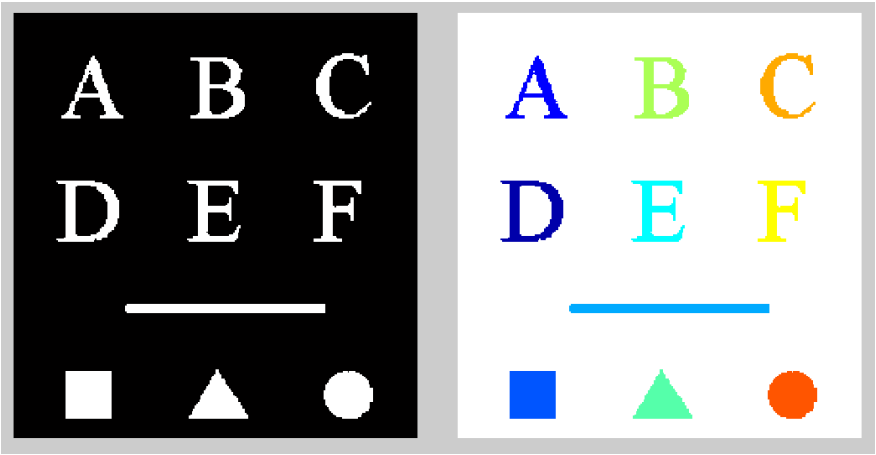
Label=0 (background)

1	1	1	0	0	0	0	0
1	1	1	0	2	2	0	0
1	1	1	0	2	2	0	0
1	1	1	0	0	0	4	0
1	1	1	0	0	0	4	0
1	1	1	0	0	0	4	0
1	1	1	0	0	3	0	0
1	1	1	0	0	0	0	0

Matriz label con
conectividad-4

1	1	1	0	0	0	0	0
1	1	1	0	2	2	0	0
1	1	1	0	2	2	0	0
1	1	1	0	0	0	2	0
1	1	1	0	0	0	2	0
1	1	1	0	0	0	2	0
1	1	1	0	0	2	0	0
1	1	1	0	0	0	0	0

Matriz label
conectividad-8



```
BW=imread('letras_objetos.bmp');
[L, n]=bwlabel(BW);
A=label2rgb(L);
```

Ejercicio: Aplique el comando **bwlabel** para etiquetar los objetos presentes en una imagen binaria.

Morfología en escala de grises

- Muchas operaciones morfológicas originalmente desarrolladas para imágenes binarias pueden ser extendidas a imágenes grayscale.
- Se emplea una imagen de entrada f(x,y) y un elemento estructural b(x,y)
- Los elementos estructurales en morfología de escala de grises son de dos categorías: planos y no planos (flat and not flat).
- SE no planos: pueden crearse usando **strel**, como lo hacíamos con los SE planos.
- En el caso de los SE no planos, tenemos que pasar como parámetro una segunda matriz (que contiene los valores de altura).

❖ Dilatación y erosión:

Dilatación de una imagen grayscale f(x,y) por un SE plano b:

$$[f \oplus b](x, y) = \max_{(s,t) \in b} \{f(x - s, y - t)\}$$

Para SE no planos b_N tenemos que:

$$[f \oplus b_N](x, y) = \max_{(s,t) \in b_N} \{f(x - s, y - t) + b_N(s, t)\}$$

- Erosión de una imagen grayscale *f(x,y)* por un SE plano *b*:

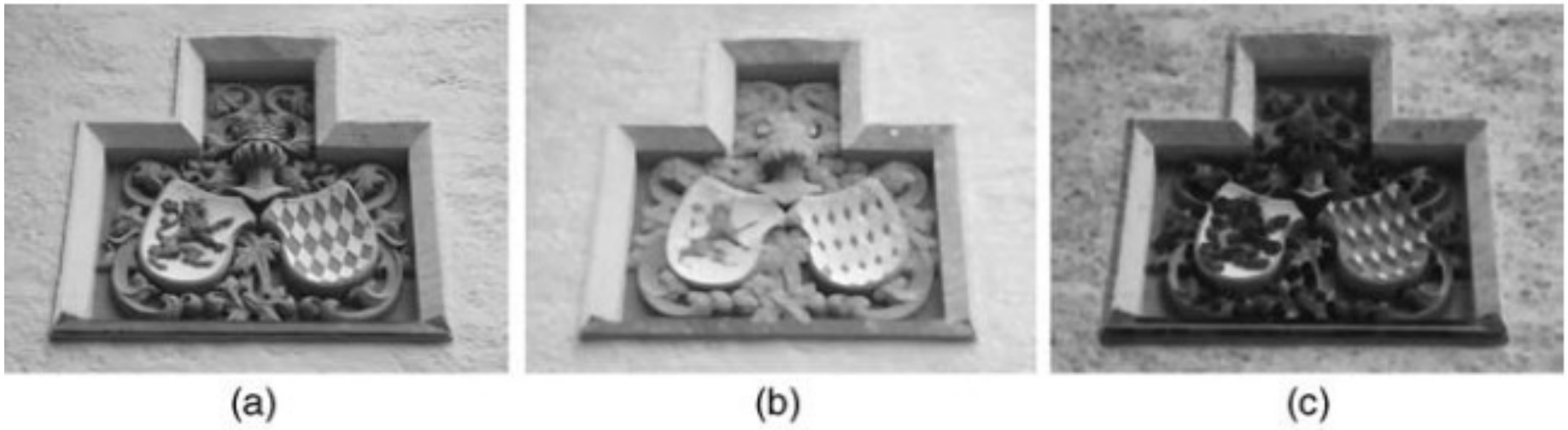
$$[f \ominus b](x, y) = \min_{(s,t) \in b} \{f(x + s, y + t)\}$$

Para SE no planos b_N:

$$[f \ominus b_N](x, y) = \min_{(s,t) \in b_N} \{f(x + s, y + t) - b_N(s, t)\}$$

- Ejemplo: erosión y dilatación de una imagen grayscale con un elemento estructural no plano con forma de bola (ball-shaped) de radio 5, generado con la función strel:

se = strel('ball',5,5);

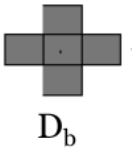


(a) Imagen de entrada, (b) dilatación, (c) erosión

Erosion

$$[f \ominus b](x, y) = \min_{(s,t)\in b}\{f(x+s, y+t)\}$$

b:



D_b

SE plano

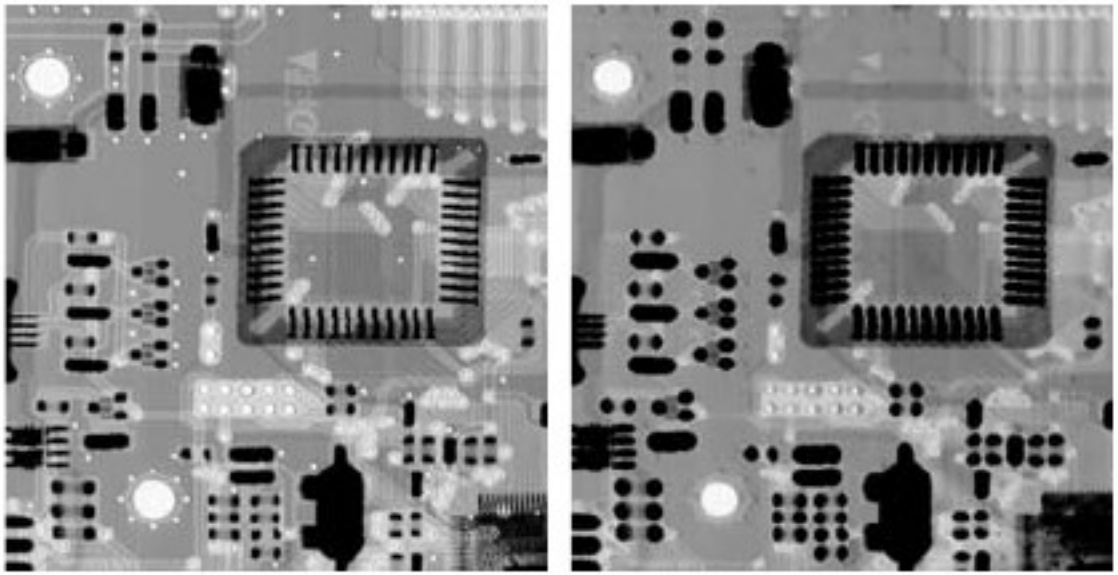
D_b *dominio* de b

1	1	1	1	1
1	6	1	7	1
6	6	1	1	1
6	6	6	1	1
6	7	7	7	6

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
6	6	1	1	1
6	6	6	6	1

- Se posiciona el origen del SE en todos los píxeles de la imagen.
- Se busca el mínimo entre los píxeles que quedan comprendidos por el SE.
- En general las imágenes grayscale erosionadas tienden a ser más oscuras que la original.
- El tamaño de los objetos claros disminuye y de los oscuros aumenta.

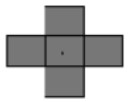
SE plano,
disco r=2



Dilatación

$$[f \oplus b](x, y) = \max_{(s,t)\in b} \{f(x - s, y - t)\}$$

b:



D_b

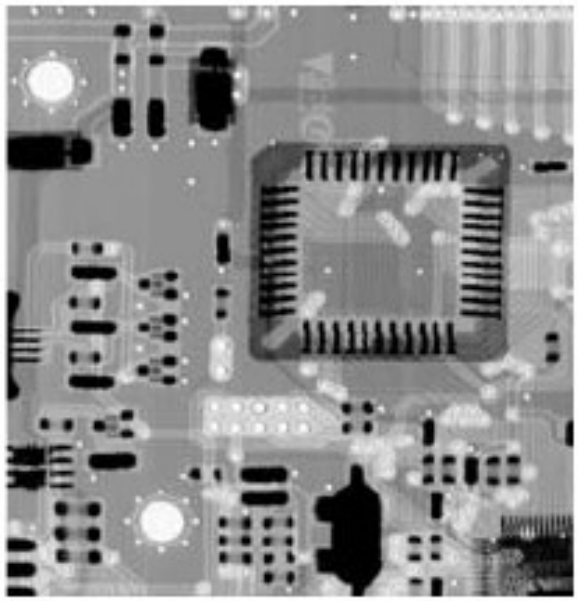
SE plano

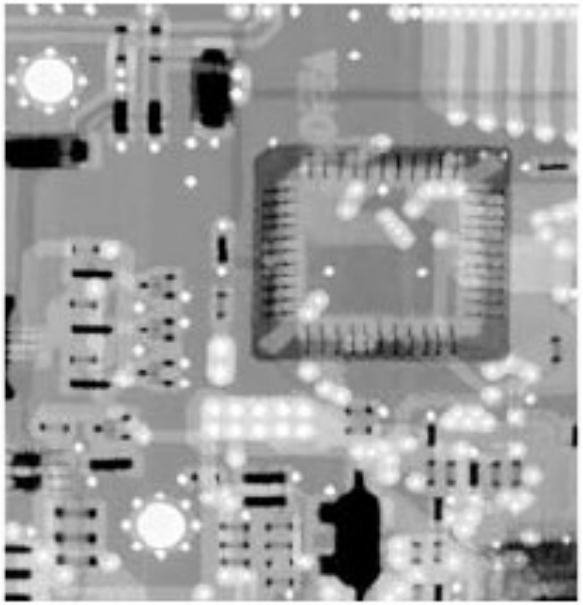
0	0	0	0	0
0	5	5	0	0
0	5	6	5	0
0	5	0	6	5
0	5	6	5	0

0	5	5	0	0
5	5	6	5	0
5	6	6	6	5
5	5	6	6	6
5	6	6	6	5

- Se busca el máximo entre los píxeles que quedan comprendidos por el SE.
- En general las imágenes en escala de grises dilatadas tienden a ser más claras que la original.
- Efecto opuesto a la erosión: objetos claros aumentan de tamaño y los oscuros disminuyen.

SE plano, disco r=2





0	0	0	0	0
0	5	5	0	0
0	5	6	5	0
0	5	0	6	5
0	5	6	5	0

Imagen original

1	6	6	1	1
6	6	7	6	1
6	7	7	7	6
6	6	7	7	7
6	7	7	7	6

Dilatación con b(x,y)=1

0	5	5	0	0
5	5	6	5	0
5	6	6	6	5
5	5	6	6	6
5	6	6	6	5

Dilatación con b(x,y)=0 (elemento plano)

1	1	1	1	1
1	6	1	7	1
6	6	1	1	1
6	6	6	1	1
6	7	7	7	6

Imagen original

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
5	5	0	0	0
5	5	5	0	0

Erosión con b(x,y)=1

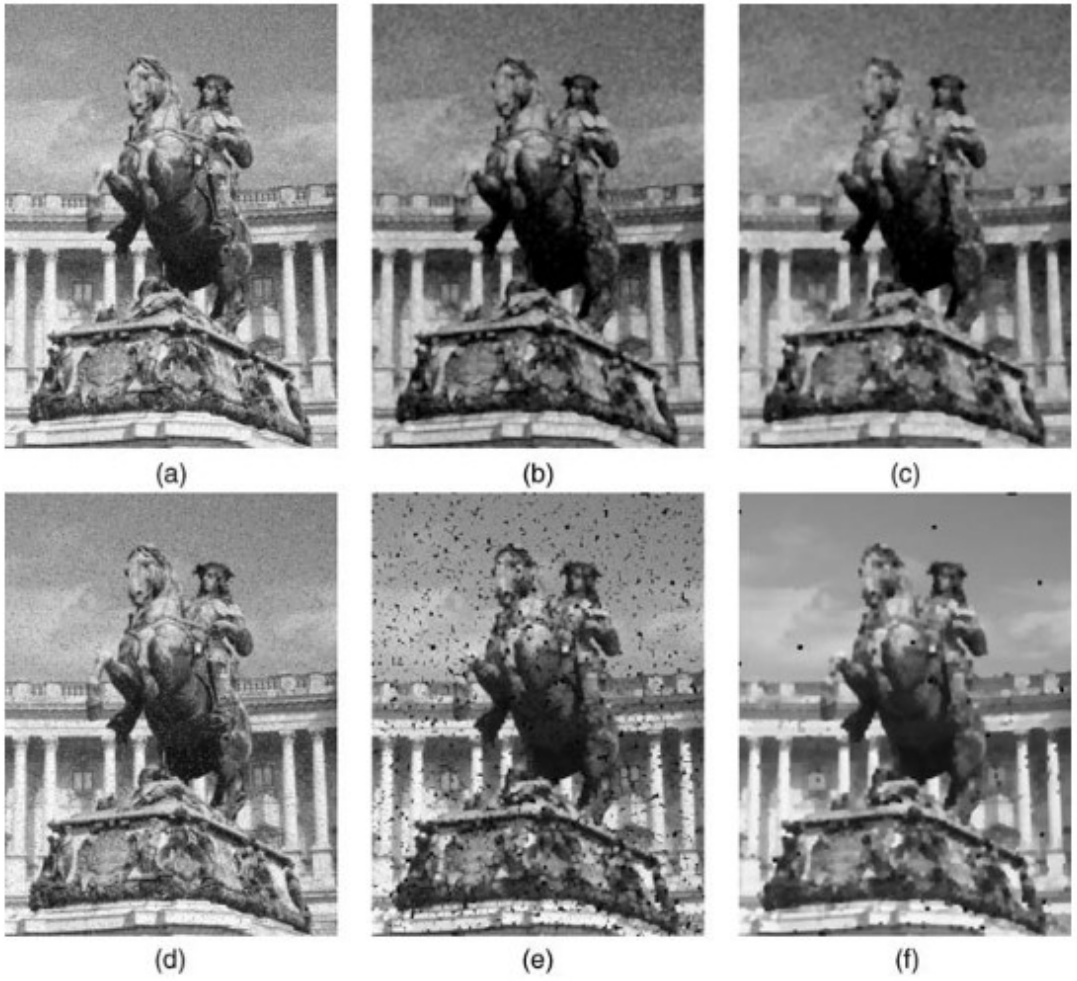
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
6	6	1	1	1
6	6	6	6	1

Erosión con b(x,y)=0 (elemento plano)

❖ **Apertura y cierre:** se definen exactamente como en morfología binaria. Son usadas en combinación para suavizar una imagen o reducir el ruido.

Suavizado morfológico:

Para reducir el ruido de una imagen en escala de grises podemos aplicarle las operaciones de apertura y cierre empleando un SE plano con forma de disco r=3, obteniendo:



(a) Imagen con **ruido Gaussiano** (media=0 y varianza=0.01)

(a) Apertura de a)

(a) Cierre de b)

(d) Imagen con **ruido sal y pimienta**

(e) Apertura de d)

(f) Cierre de e)

Apertura: eliminará formas claras de tamaño menor que el SE

Cierre: objetos oscuros serán atenuados

Transformaciones Top-Hat y Bottom-Hat

(operaciones que combinan apertura y cierre con sustracción de imágenes)

Top-Hat: usada en el proceso de *shading correction* (corrección de sombreado), el cual consiste en compensar la iluminación no uniforme de la escena.

Puede combinarse con la operación Bottom-Hat para mejorar el contraste de la imagen.

En Matlab las funciones imtophat y imbothat implementan ambas transformaciones.

Top-Hat es usada para objetos claros sobre un fondo oscuro, y Bottom-Hat para el caso contrario.

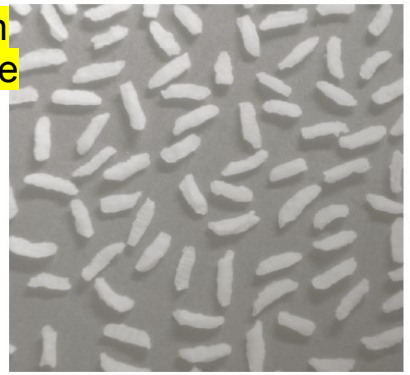
$$Top-hat(f) = f - (f \circ b)$$

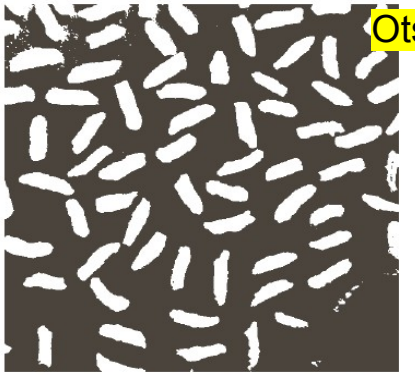
$$Bottom-hat(f) = (f \cdot b) - f$$

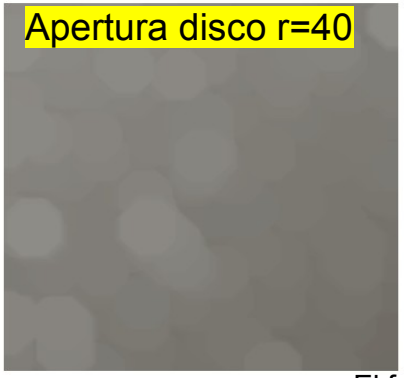
Aplicaciones: eliminar objetos usando un SE que no se ajuste a los objetos que se eliminarán.

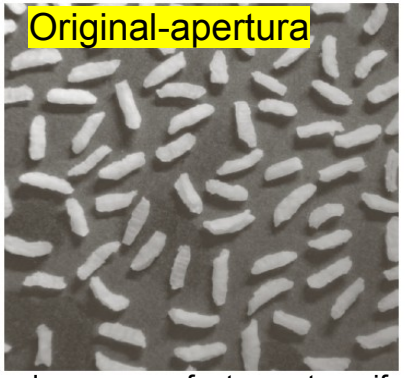
La diferencia entre la imagen original y la apertura o cierre, produce una imagen en la que sólo quedan los componentes eliminados.

Disco grande para no ajustarse a ningún objeto. Los objetos se eliminan.









El fondo no es perfectamente uniforme, pero las diferencias entre los objetos y el fondo son mayores, suficiente para segmentar

Mejora de contraste: se pueden aplicar las transformaciones Top-Hat y Bottom-Hat, como se observa en el ejemplo siguiente.

Aplicando imtophat e imbothat sobre una imagen grayscale se logra mejorar el contraste, usando un elemento estructural con forma de disco (radio 3).

La imagen de salida se obtiene haciendo:

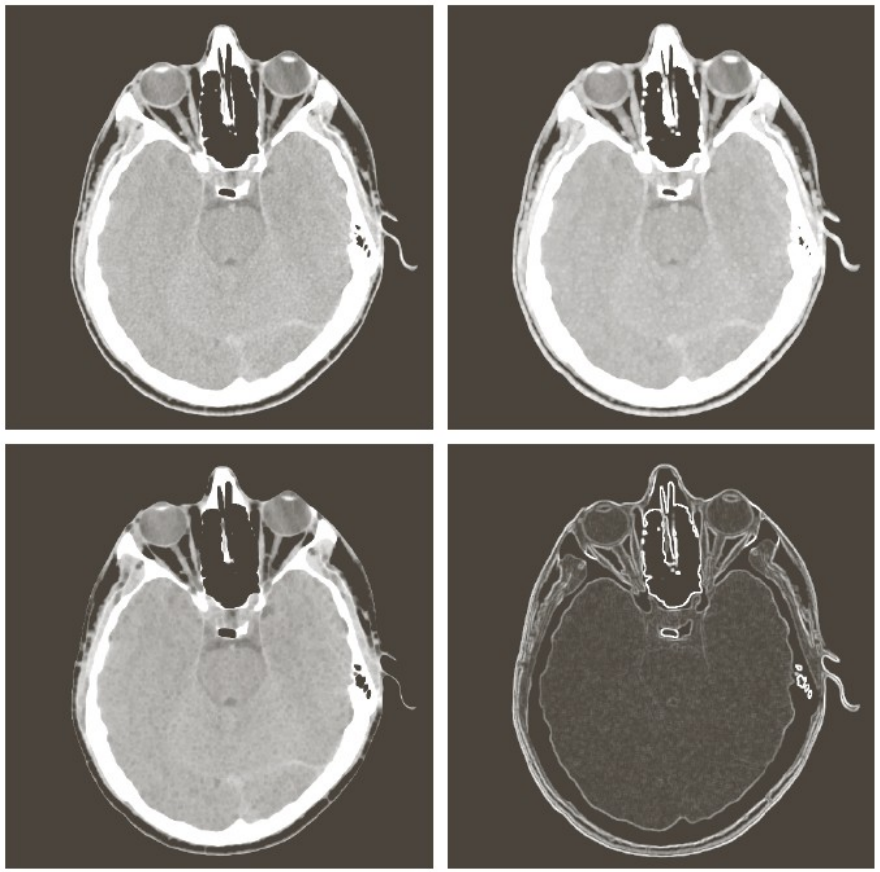
O= imadd (I, imtophat (I, se)),
R = imsubtract (O, imbothat (I,se));

Gradiente Morfológico

$$g = (f \oplus b) - (f \ominus b)$$

- La dilatación espesa regiones en una imagen y la erosión las encoge.
- La diferencia entre ambas operaciones enfatiza los límites entre las regiones.
- Áreas homogéneas no se ven afectadas (siempre que el SE sea relativamente pequeño), por lo que la operación de resta tiende a eliminarlos.
- El resultado neto es una imagen en la que los bordes son resaltados y la contribución de las áreas homogéneas se suprime, produciendo un efecto de "derivada" (gradiente).

SE cuadrado 3x3



a b
c d

FIGURE 9.39
(a) 512 × 512 image of a head CT scan. (b) Dilation. (c) Erosion. (d) Morphological gradient, computed as the difference between (b) and (c). (Original image courtesy of Dr. David R. Pickens, Vanderbilt University.)