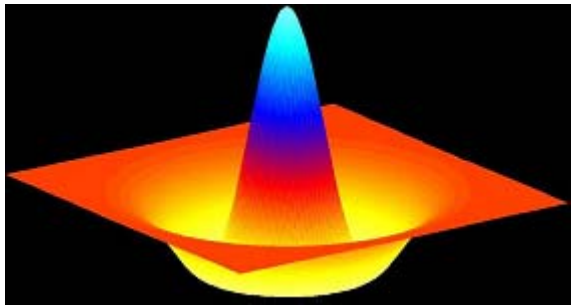


Procesamiento Digital de Imágenes

Análisis Morfológico

Análisis Morfológico

El procesamiento morfológico de imágenes es una herramienta para la extracción de componentes de la imagen que sean útiles en la representación y descripción de regiones como ser: envolvente convexa, esqueletos, etc.

En otras palabras, las operaciones morfológicas simplifican las imágenes y conservan las principales características de forma de los objetos.

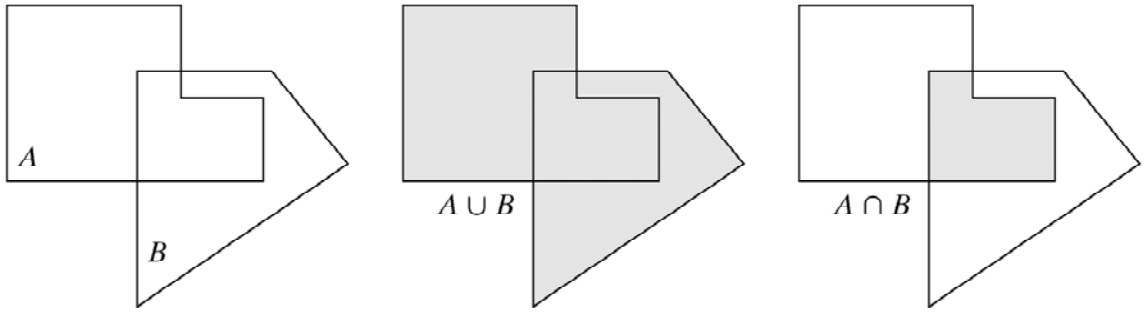
También se utilizan para tareas de pre y post procesamiento como ser el filtrado morfológico y el adelgazamiento y engrosamiento.

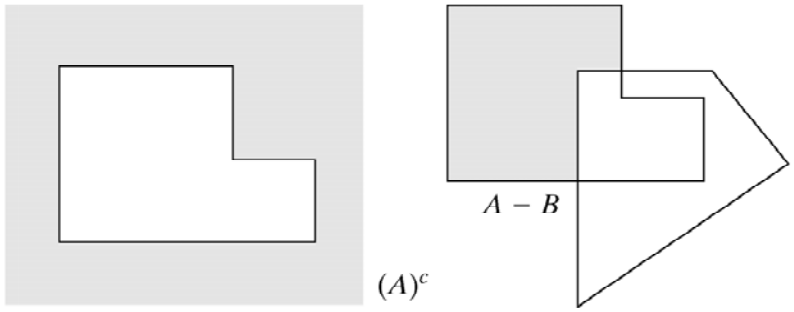
La morfología matemática se basa en operaciones de teoría de conjuntos. En el caso de imágenes binarias, los conjuntos tratados son subconjuntos de Z^2 y en el de las imágenes en escala de grises, se trata de conjuntos de puntos con coordenadas en Z^3. En definitiva, se mapea la grilla de coordenadas (x,y) a partir de una f(x,y) que puede tener valores: solo 0s y 1s (imagen binaria) o valores enteros en un rango [0,255] (imagen en escala de grises)

Operaciones básicas de conjuntos

Veamos algunas operaciones y sus definiciones:

$$A \cup B = \{w/\ w \in A \vee w \in B\} \quad A \cap B = \{w/\ w \in A \wedge w \in B\}$$



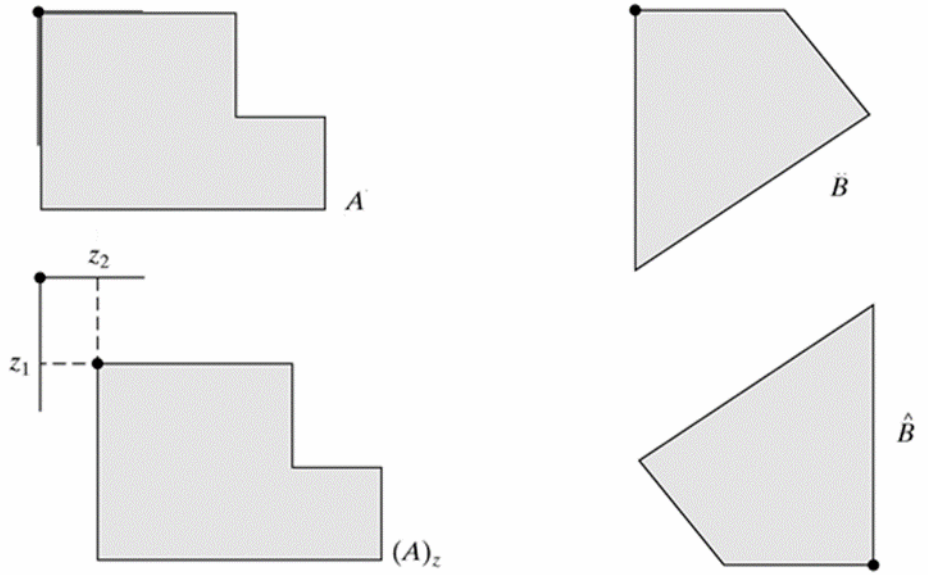


$$A - B = \{w/\ w \in A \wedge w \notin B\}$$

Operaciones de traslación y reflexión

Estas operaciones son específicas de conjuntos donde sus elementos son coordenadas de pixels.

$$A_z = \{c \,/\, c = a + z \quad \forall\, a \in A\} \qquad \hat{B} = \{w \,/\, w = -b \quad \forall\, b \in B\}$$



Operaciones básicas en MATLAB

Para imágenes binarias ya tenemos implementadas en MATLAB las siguientes funciones:

Set Operation	MATLAB Expression for Binary Images	Name
$A \cap B$	A & B	AND
$A \cup B$	A | B	OR
A^c	~A	NOT
$A - B$	A & ~B	DIFFERENCE

Dilatación

Esta operación “aumenta” o “engrosa” los objetos de una imagen binaria. La forma en que se engrosan los objetos depende de un elemento estructural.

El elemento estructural tiene claramente identificado su origen. Así la dilatación consiste en trasladar el origen del elemento estructural a lo largo de toda la imagen y ver si existe solapamiento con los pixels de valor 1.

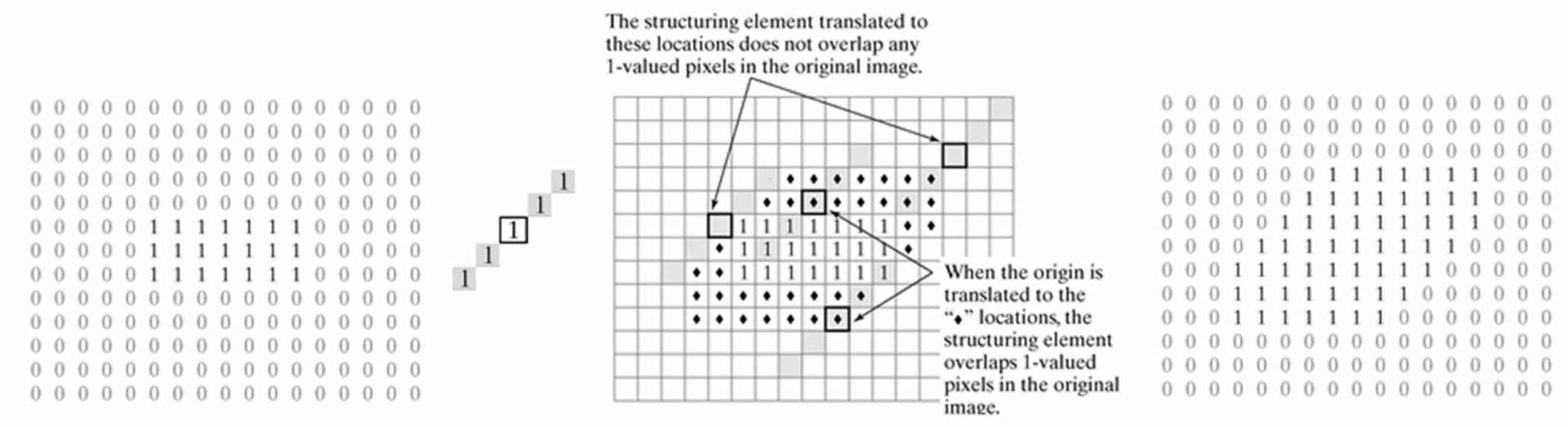
El elemento estructural es por lo general mucho más pequeño que la imagen.

Matemáticamente,

$$A \oplus B = \left\{ z \,/\, \hat{B}_z \cap A \neq \emptyset \right\}$$

Dilatación

En esta operación, la traslación del elemento estructural es similar a la convolución de 2 imágenes como vimos anteriormente.



La función MATLAB que implementa esta operación es:

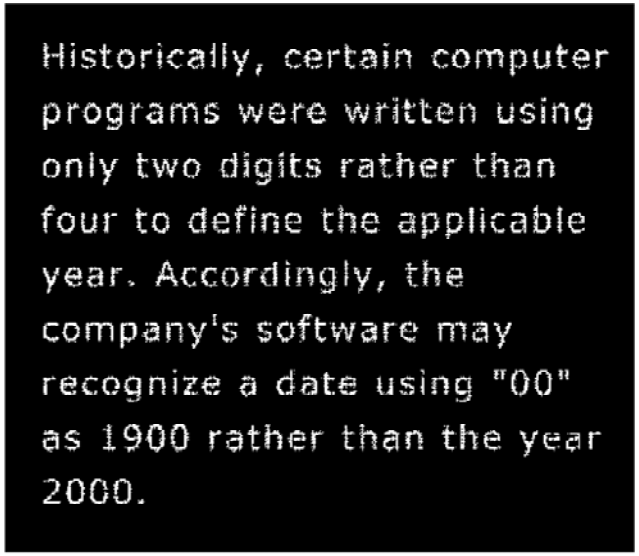
```
A2 = imdilate ( A , B )
```

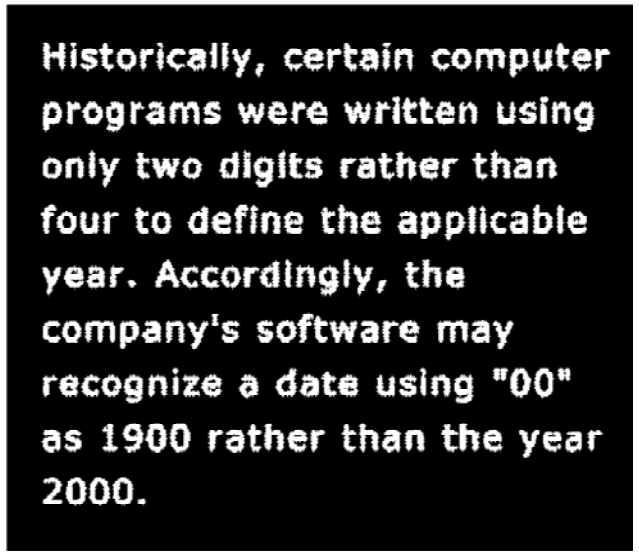
Dilatación

```
>> A = imread('broken_text.tif');
>> B = [0 1 0;1 1 1;0 1 0];
>> A2 = imdilate ( A , B );
>> imshow(A2)
```

Elemento Estructural

0	1	0
1	**1**	1
0	1	0





Dilatación

Dado que la dilatación es asociativa, es decir:

$$A \oplus (B \oplus C) = (A \oplus B) \oplus C$$

Y que, por lo general, un elemento estructural se puede pensar como el resultado de la dilatación de 2 elementos estructurales más chicos:

$$B = (B_1 \oplus B_2) \quad \Longrightarrow \quad A \oplus B = A \oplus (B_1 \oplus B_2) = (A \oplus B_1) \oplus B_2$$

Es decir, la operación de dilatación A ⊕ B se puede descomponer en dos dilataciones sucesivas con los elementos estructurales B1 y B2. Computacionalmente esto último reduce sustancialmente la cantidad de cálculos a realizar.
Para generar un elemento estructural y descomponerlo en elementos estructurales más pequeños con MATLAB utilizamos las funciones:

```
se = strel( 'diamond' , 5 );
decomp = getsequence(se);
```

Dilatación – Generando elementos estructurales

`strel`

Syntax Forms	Description
`se = strel('diamond', R)`	Creates a flat, diamond-shaped structuring element, where `R` specifies the distance from the structuring element origin to the extreme points of the diamond.
`se = strel('disk', R)`	Creates a flat, disk-shaped structuring element with radius `R`. (Additional parameters may be specified for the disk; see the `strel` help page for details.)
`se = strel('line', LEN, DEG)`	Creates a flat, linear structuring element, where `LEN` specifies the length, and `DEG` specifies the angle (in degrees) of the line, as measured in a counterclockwise direction from the horizontal axis.
`se = strel('octagon', R)`	Creates a flat, octagonal structuring element, where `R` specifies the distance from the structuring element origin to the sides of the octagon, as measured along the horizontal and vertical axes. `R` must be a nonnegative multiple of 3.
`se = strel('pair', OFFSET)`	Creates a flat structuring element containing two members. One member is located at the origin. The second member's location is specified by the vector `OFFSET`, which must be a two-element vector of integers.
`se = strel('periodicline', P, V)`	Creates a flat structuring element containing `2*P + 1` members. `V` is a two-element vector containing integer-valued row and column offsets. One structuring element member is located at the origin. The other members are located at `1*V`, `-1*V`, `2*V`, `-2*V`, `...`, `P*V`, and `-P*V`.
`se = strel('rectangle', MN)`	Creates a flat, rectangle-shaped structuring element, where `MN` specifies the size. `MN` must be a two-element vector of nonnegative integers. The first element of `MN` is the number rows in the structuring element; the second element is the number of columns.
`se = strel('square', W)`	Creates a square structuring element whose width is `W` pixels. `W` must be a nonnegative integer scalar.
`se = strel('arbitrary', NHOOD)` `se = strel(NHOOD)`	Creates a structuring element of arbitrary shape. `NHOOD` is a matrix of 0s and 1s that specifies the shape. The second, simpler syntax form shown performs the same operation.

Erosión

Esta operación “afina” los objetos de una imagen binaria. La forma en que se erosionan los objetos depende de un elemento estructural.

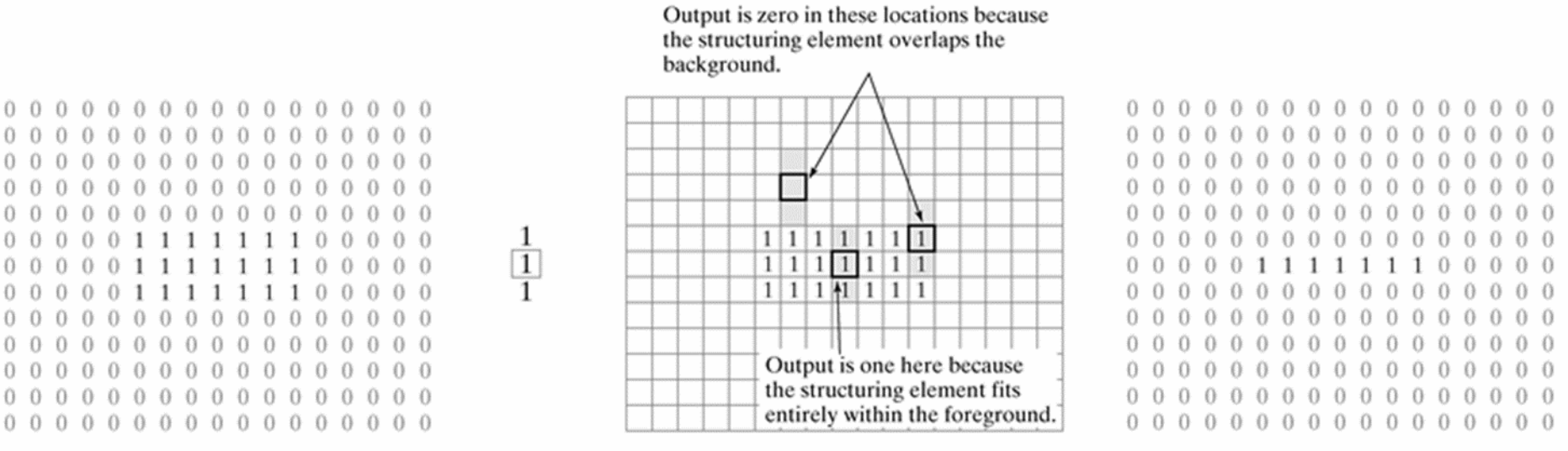
El elemento estructural tiene claramente identificado su origen. Así la erosión consiste en trasladar el elemento estructural a lo largo de toda la imagen y ver si el mismo queda contenido completamente en la zona de la imagen con valores 1. Si esto sucede el resultado de la erosión tendrá un valor 1 en el origen del elemento estructural.

Matemáticamente,

$$A \ominus B = \{z \,/\, B_z \cap A^C \neq \emptyset\}$$

Erosión

En otras palabras la erosión de A con B es el conjunto de las ubicaciones del origen del elemento estructural tal que el elemento se solape completamente con los pixels de valor 1 de la imagen original.

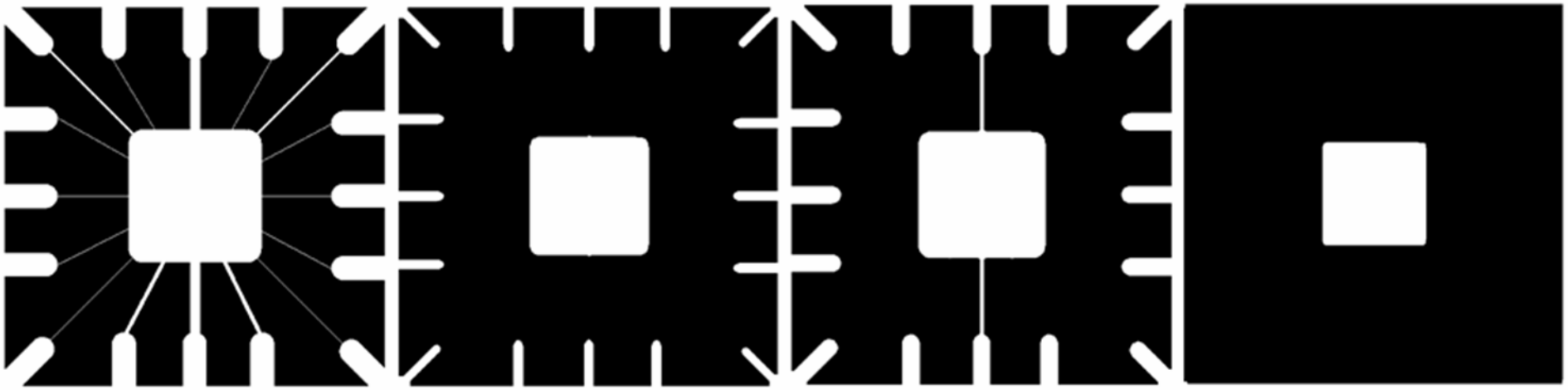


La función MATLAB que implementa esta operación es:

```
A2 = imerode ( A , B )
```

Erosión

```
>> A = imread('wirebond_mask.tif');
>> L = 10;
>> B = strel('disk', L );
>> A2 = imerode( A , B );
>> imshow(A2)
```

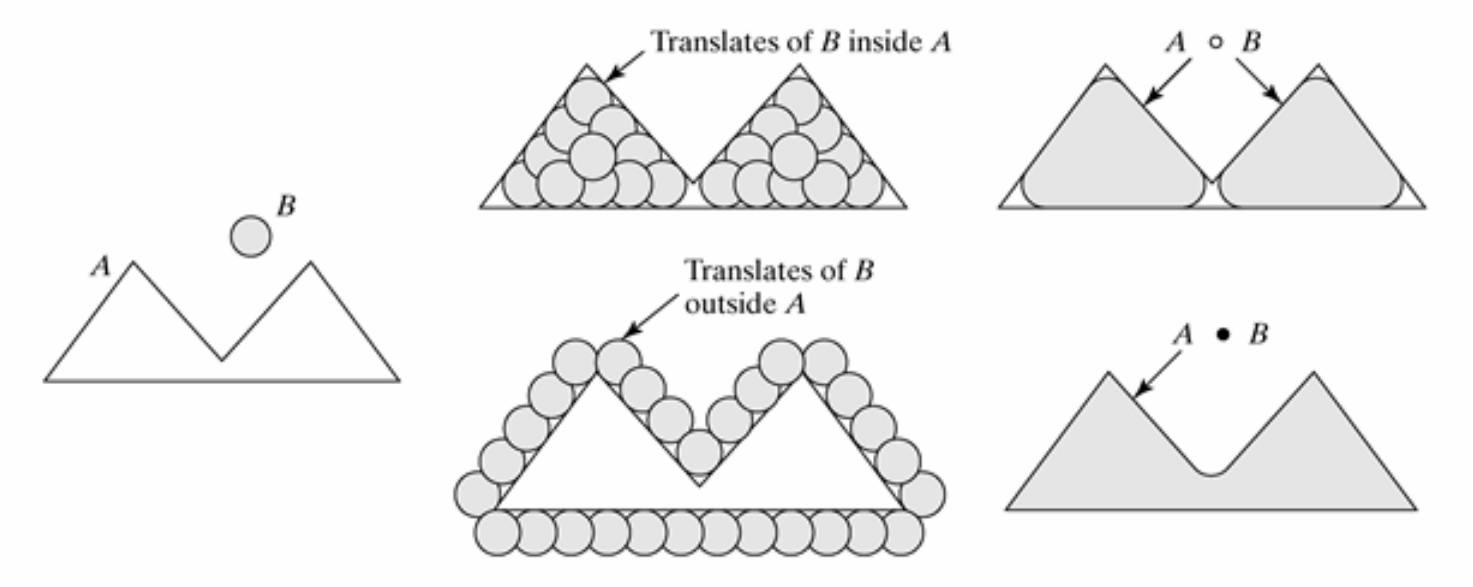


Apertura y Clausura

Combinando las 2 operaciones anteriores se pueden definir 2 nuevas operaciones: la apertura y la clausura. Matemáticamente,

$$\text{Apertura} \rightarrow A \circ B = (A \ominus B) \oplus B$$

$$\text{Clausura} \rightarrow A \bullet B = (A \oplus B) \ominus B$$



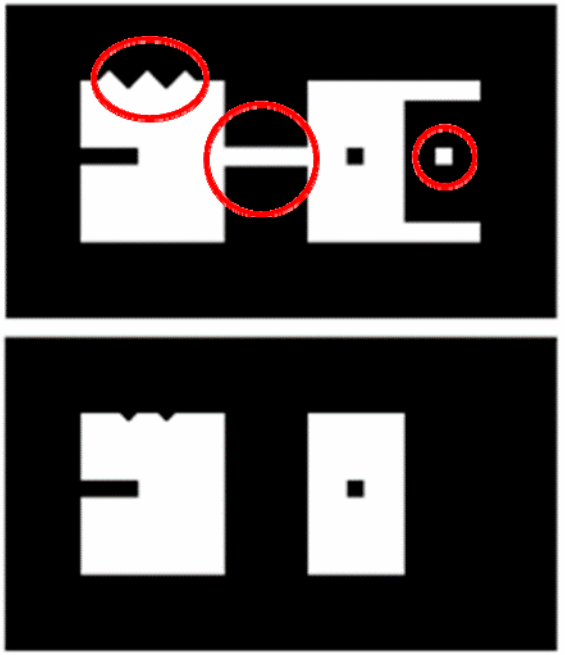
Apertura

Esta operación es la unión de todas las traslaciones de B que entran completamente en A. La apertura remueve las regiones de un objeto que no pueden contener al elemento estructural, suaviza el contorno y corta enlaces finos.

La función MATLAB que implementa esta operación es:

```
C = imopen( A , B )
```

```
>> A = imread('shapes.tif');
>> B = strel('square', 20);
>> Aop = imopen( A , B );
>> imshow(Aop)
```



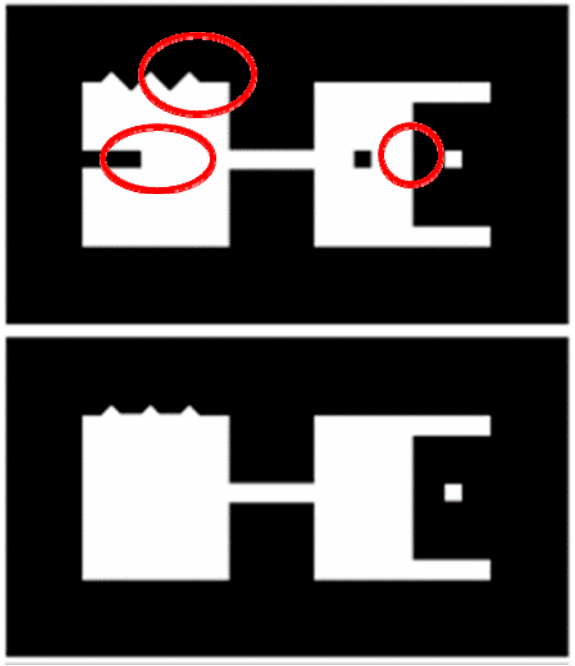
Clausura

Esta operación es el complemento de la unión de todas las traslaciones de B que no se solapan con A. Esta operación también tiende a suavizar contornos de objetos pero engrosa enlaces finos, rellena “golfos” y agujeros más pequeños que el elemento estructural.

La función MATLAB que implementa esta operación es:

```
C = imclose( A , B )
```

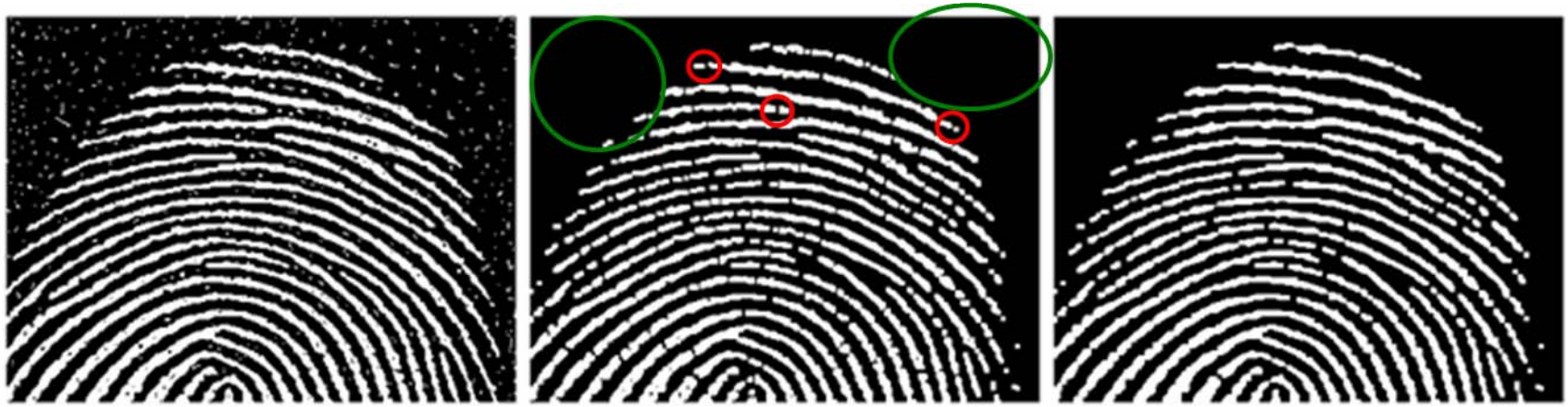
```
>> A = imread(‘shapes.tif’);
>> B = strel(‘square’, 20);
>> Acls = imclose( A , B );
>> imshow(Acls)
```



Combinando Apertura y Clausura

En algunos casos, la combinación de las dos operaciones puede ser efectiva para remover ruído como se muestra en el siguiente ejemplo.

```
>> f = imread('fingerprint.tif');
>> se = strel('square', 3);
>> fop = imopen( f , se );
>> imshow(fop)
>> fop_cls = imclose( fop , se );
>> imshow(fop_cls)
```



Transformación Hit-or-Miss

Esta operación permite identificar configuraciones específicas de pixels o pixels que se encuentran al final de un segmento de línea. Se define como:

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

donde B es un par de elementos estructurales, tal que $B=(B_1,B_2)$. Por ejemplo, sea B_1 el conjunto formado por los pixels negros de B y B_2 el conjunto formado por los pixels negros de B^C.

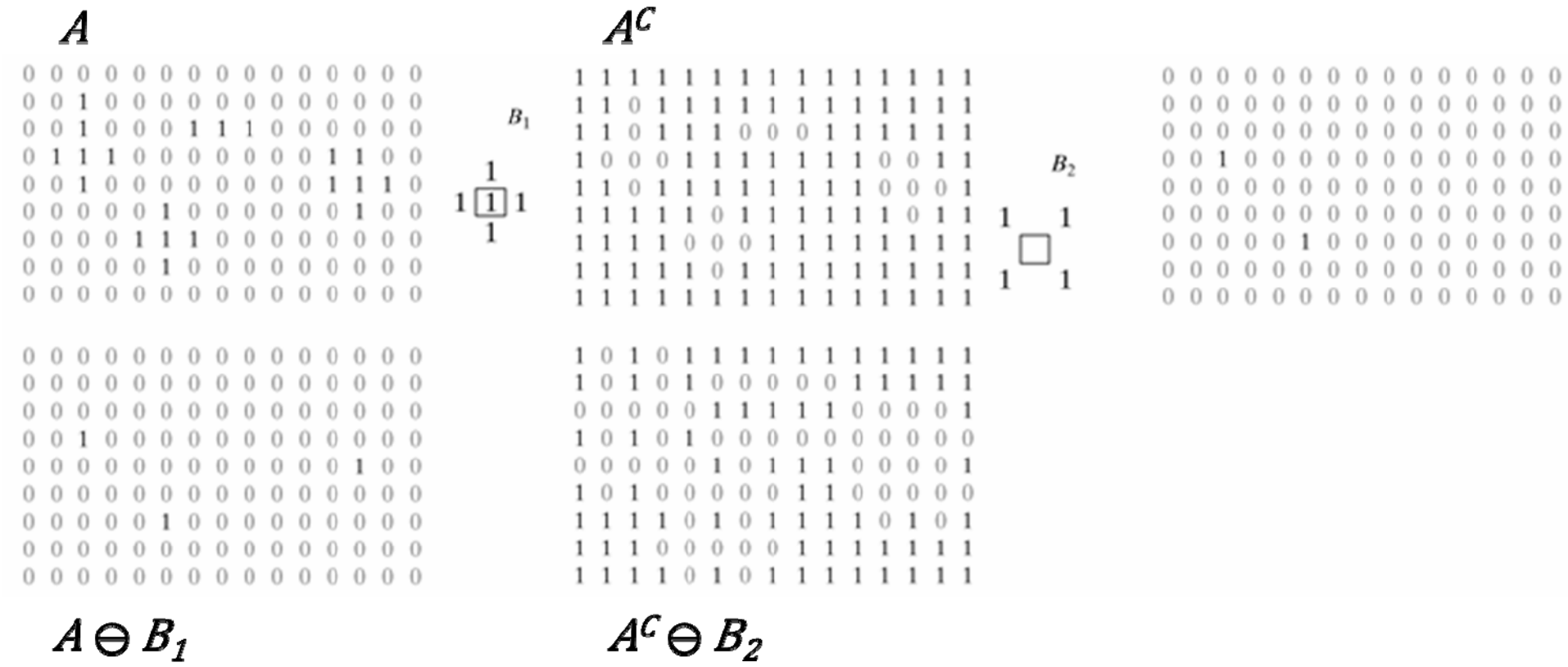
	0	1	0			0	1	0			1	0	1
B	1	1	1		B_1	1	**1**	1		B_2	0	**0**	0
	0	1	0			0	1	0			1	0	1

En MATLAB está implementada la función:

```
C = bwhitmiss( A , B1, B2 )
```

Transformación Hit-or-Miss

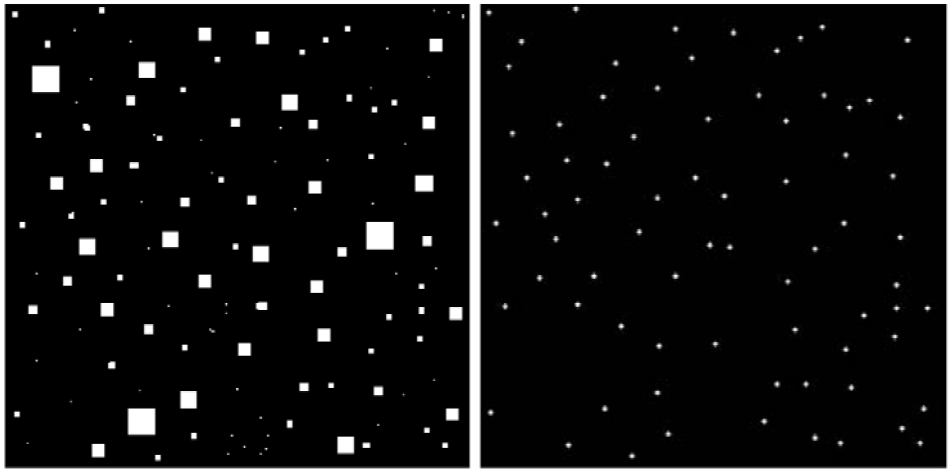
$$A \otimes B = (A \ominus B_1) \cap (A^C \ominus B_2)$$

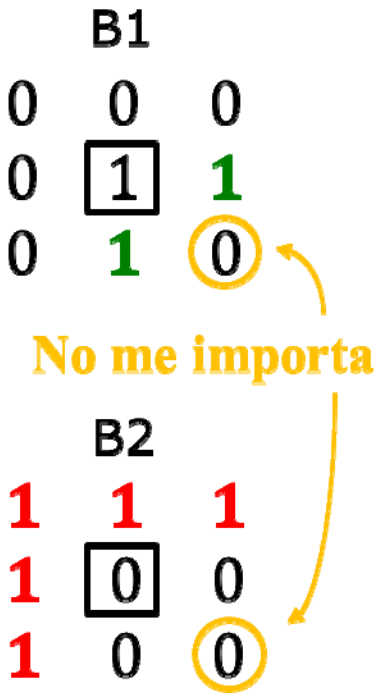


Transformación Hit-or-Miss

Para buscar las esquinas superiores de los cuadrados de la imagen, uso B_1 pixels con vecinos 1s en el sur y el este (hit) y B_2 pixels que no tengan vecinos 1s en el norte, noroeste, noreste, oeste ni suroeste (miss).

```
>> f = imread('squares.tif');
>> B1 = strel([0 0 0;0 1 1;0 1 0]);
>> B2 = strel([1 1 1;1 0 0;1 0 0]);
>> g = bwhitmiss( f , B1 , B2 );
>> imshow(g)
```





Transformación Hit-or-Miss – Lookup Tables

Una forma rápida de hacer la transformación hit-or-miss cuando B_1 y B_2 son pequeños es con las tablas lookup (LUT). La técnica consiste en precalcular el valor del pixel de salida para toda configuración de vecinos posible y guardar los resultados en una tabla para su uso posterior. Por ejemplo, para un vecindario de 3x3 hay $2^9 = 512$ configuraciones distintas en imágenes binarias. Entonces usamos la siguiente matriz:

$$\begin{matrix} 1 & 8 & 64 \\ 2 & 16 & 128 \\ 4 & 32 & 256 \end{matrix} \qquad \begin{matrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{matrix} \longrightarrow \mathbf{399}$$

para multiplicar elemento a elemento con la configuración deseada y guardamos el resultado. En MATLAB se usan las funciones:

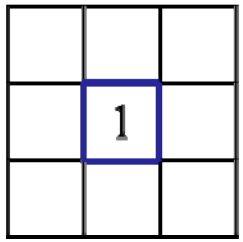
`makelut` → Construye una LUT a partir de una función definida por el usuario. Requiere escribir una función cuya entrada sea una matriz 3x3 y la salida 0 o 1. Luego **`makelut`** llama a la función 512 veces y genera una vector con los 512 pares.

`applylut` → Procesa la imagen a partir de la LUT.

Transformación Hit-or-Miss – Lookup Tables

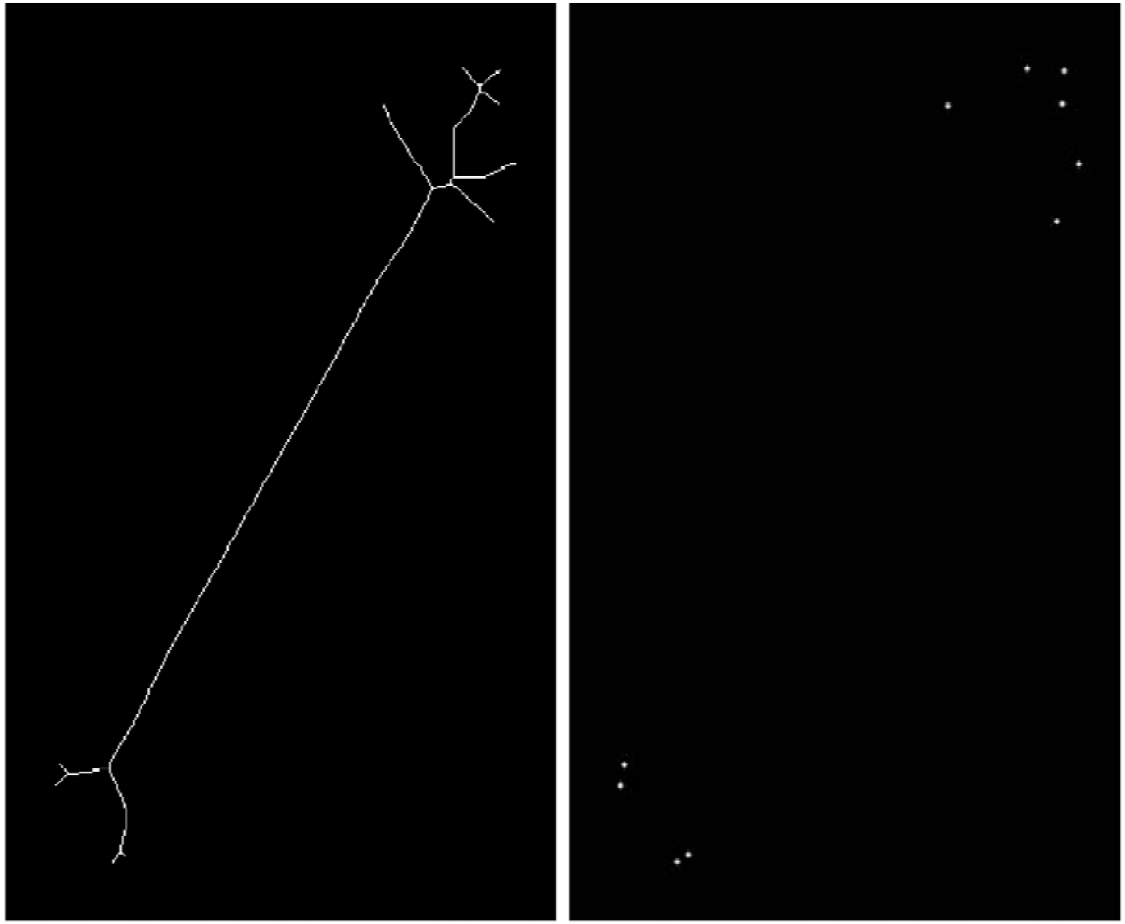
```
function g = endpoints(f)
% Busca los puntos terminales de una imagen binaria
persistent lut % es un “hold” de la lut para que la reinicie
if isempty(lut)
                lut = makelut(@endpoint_fcn,3);
end
g = applylut(f,lut);
```

Un solo 1 en alguno de los 8 vecinos y ceros en los restantes




```
function is_end_point = endpoint_fcn(nhood)
% Devuelve un 1 si el elemento central de nhood es un punto
% terminal
is_end_point = nhood(2,2) & (sum(nhood(:)) == 2);
```

Transformación Hit-or-Miss – Lookup Tables



Operaciones Morfológicas implementadas

La función **bwmorph** implementa una variedad de operaciones útiles a partir de combinar dilataciones, erosiones y tablas lookup.

```
g = bwmorph(f,operation,n)
```

Operation	Description
bothat	"Bottom-hat" operation using a 3 × 3 structuring element; use imbothat (see Section 9.6.2) for other structuring elements.
bridge	Connect pixels separated by single-pixel gaps.
clean	Remove isolated foreground pixels.
close	Closing using a 3 × 3 structuring element; use imclose for other structuring elements.
diag	Fill in around diagonally connected foreground pixels.
dilate	Dilation using a 3 × 3 structuring element; use imdilate for other structuring elements.
erode	Erosion using a 3 × 3 structuring element; use imerode for other structuring elements.
fill	Fill in single-pixel "holes" (background pixels surrounded by foreground pixels); use imfill (see Section 11.1.2) to fill in larger holes.
hbreak	Remove H-connected foreground pixels.
majority	Make pixel p a foreground pixel if at least five pixels in $N_8(p)$ (see Section 9.4) are foreground pixels; otherwise make p a background pixel.
open	Opening using a 3 × 3 structuring element; use function imopen for other structuring elements.
remove	Remove "interior" pixels (foreground pixels that have no background neighbors).
shrink	Shrink objects with no holes to points; shrink objects with holes to rings.
skel	Skeletonize an image.
spur	Remove spur pixels.
thicken	Thicken objects without joining disconnected 1s.
thin	Thin objects without holes to minimally connected strokes; thin objects with holes to rings.
tophat	"Top-hat" operation using a 3 × 3 structuring element; use imtophat (see Section 9.6.2) for other structuring elements.

Operaciones Morfológicas implementadas

```
>> f = imread('fingerprint_cleaned.tif');
>> g1 = bwmorph(f,'thin',1);
>> g2 = bwmorph(f,'thin',2);
>> imshow(g1),figure, imshow(g2)
>> ginf = bwmorph(f,'thin',inf);
>> figure, imshow(ginf)
```

g1

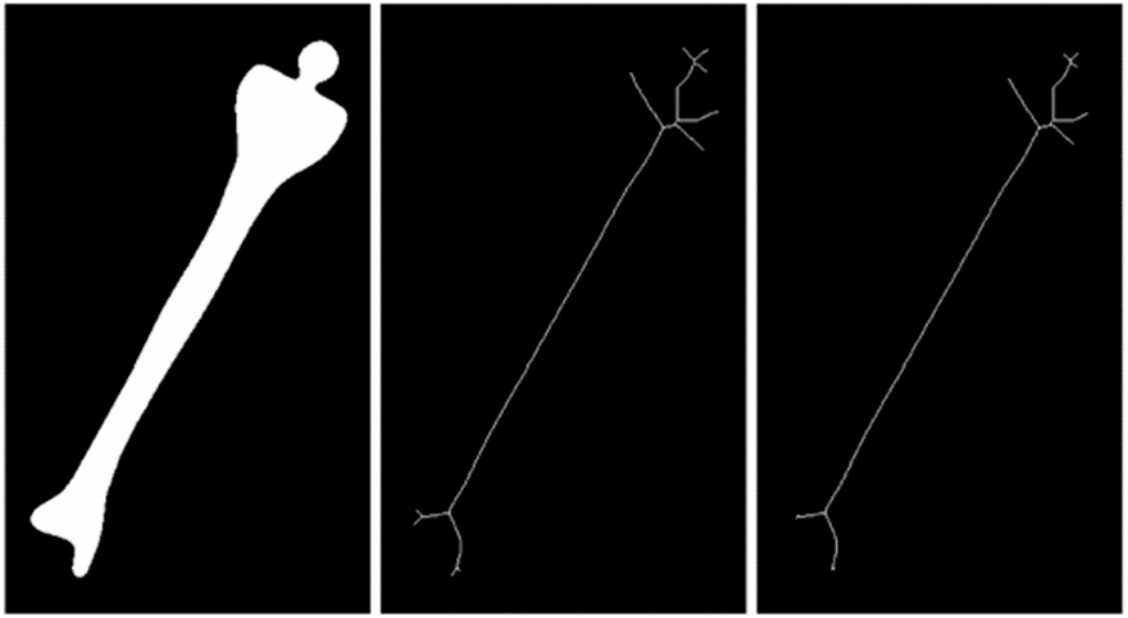
g2

ginf (repite la operación hasta que no hay cambios)

Operaciones Morfológicas implementadas

```
>> f = imread('bone.tif');
>> fs = bwmorph(f,'skel',inf);
>> imshow(f),figure, imshow(fs)
>> for k = 1:5
>>     fs = fs & ~endpoints(fs);
>> end; figure, imshow(fs)
```

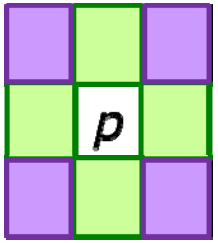
Realiza una "poda" de los puntos terminales



Etiquetado de Componentes Conectados

Para definir un objeto (formado por pixels conectados) de una imagen es necesario definir primero los diferentes tipos de vecinos. Sea un pixel *p* con coordendas *(x,y)* sus 4-vecinos, $N_4(p)$, serán los pixels:

(x+1,y) *(x-1,y)* *(x,y+1)* *(x,y-1)*



De manera similar sus vecinos diagonales, ND(p), serán:

(x+1,y+1) *(x+1,y-1)* *(x-1,y+1)* *(x-1,y-1)*

La unión de $N_4(p)$ y $N_D(p)$ forman los 8-vecinos de *p*, $N_8(p)$.

Luego, decimos que los pixels *p* y *q* son 4-adyacentes si $q \in N_4(p)$ y de manera similar los pixels *p* y *q* son 8-adyacentes si $q \in N_8(p)$.

Un camino desde p_1 a p_n podrá ser 4-conectado u 8-conectado si en la secuencia de pixels $p_1, p_2, \ldots, p_{n-1}, p_n$ cualquier par de pixels p_k y p_{k+1} son respectivamente 4-adyacentes u 8-adyacentes.

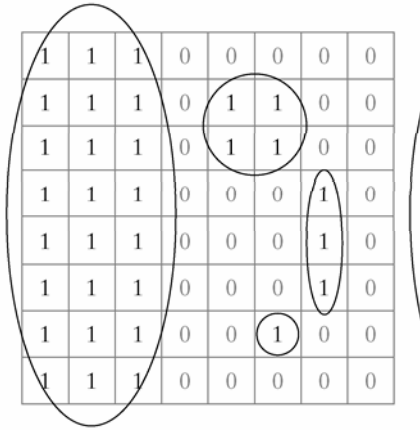
0	0	0	0	0
0	0	1	1	1
0	0	1	0	0
1	1	1	0	0
0	0	0	0	0

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

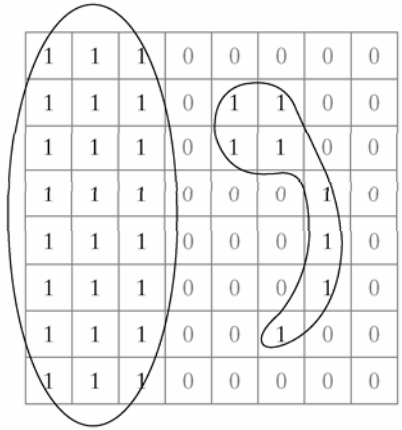
Etiquetado de Componentes Conectados

Dos pixels *p* y *q* estarán 4-conectados u 8-conectados si existe un camino 4-conectado u 8-conectado entre ellos respectivamente. El conjunto de todos los pixels conectados a *p* formarán un objeto conectado o simplemente objeto.

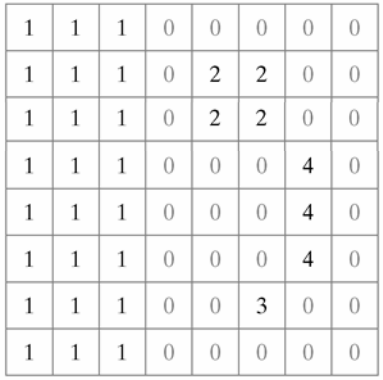
Objetos 4-conectados



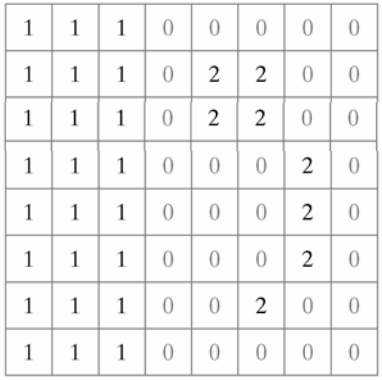
Objetos 8-conectados



4 Objetos



2 Objetos



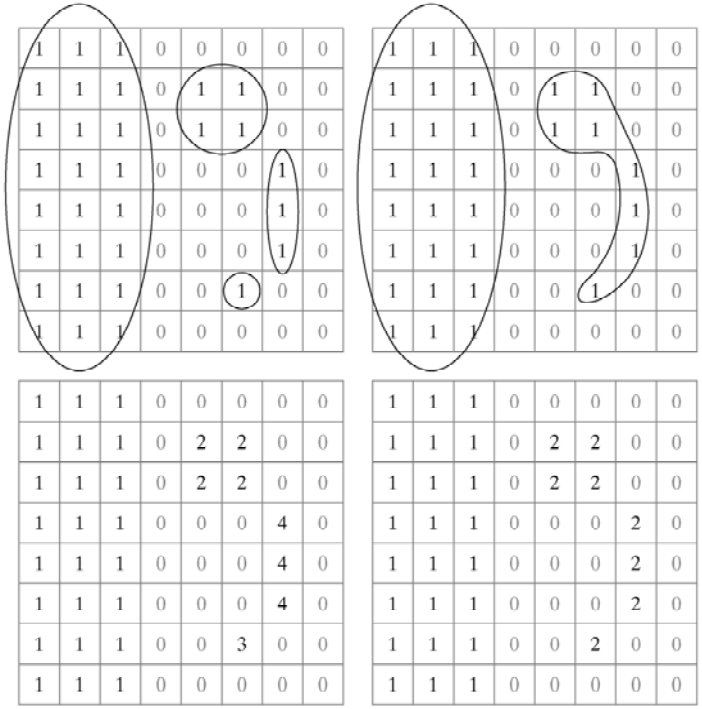
Etiquetado de Componentes Conectados

La función MATLAB que implementa esta operación es:

```
[ L , num ] = bwlabel( f , conn )
```

donde **conn** especifica el tipo de adyacencia de los pixels (4 u 8). La salida **L** es una matriz de etiquetas y **num** especifica la cantidad de objetos encontrados.

4-conectados

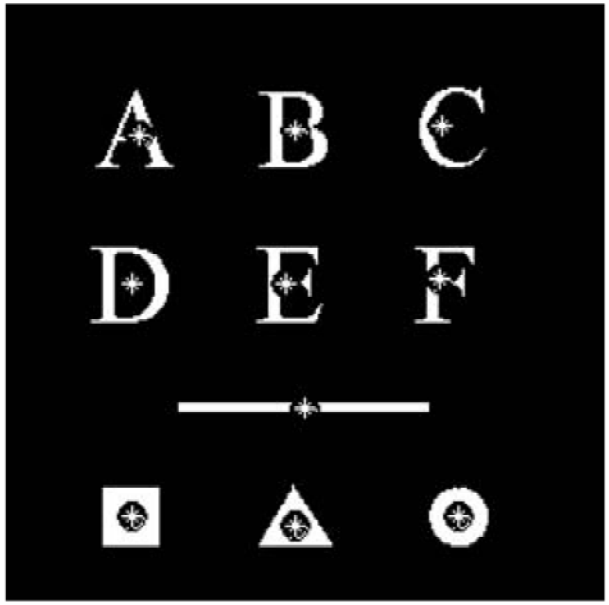


8-conectados

Etiquetado de Componentes Conectados

Puede ser de utilidad buscar el centro de masa de los objetos componentes de una imagen.

```
>> f = imread('objects.tif');
>> [ L , num ] = bwlabel( f ); % 8-conect
>> imshow(f); hold on
>> for k = 1:num
>>    [ r , c ] = find( L == k );
>>    rbar = mean( r );
>>    cbar = mean( c );
>>    plot(cbar,rbar,'Marker','c',...
>>    'MarkerEdgecolor','k',...
>>    'MarkerSize',10)
>>    plot(cbar,rbar,'Marker','*',...
>>    'MarkerEdgecolor','w')
>> end
```



Reconstrucción Morfológica

La Reconstrucción Morfológica es una operación que involucra 2 imágenes y un elemento estructural. Se definen: una imagen marcador (*marker*) *f*, que contiene el punto inicial de la transformación, una imagen máscara (*mask*) *g*, que delimita la transformación y un elemento estructural que define la conectividad.

La reconstrucción de *g* a partir de *f*, $R_g(f)$, se define a partir del siguiente proceso iterativo:

1.Inicializamos h_1 como la imagen marcador *f*, que debe ser un sub-conjunto de *g* ($f \subseteq g$)
2.Creamos un elemento estructural. Por ejemplo, **`B = ones(3)`**
3.Realizamos la siguiente operación iterativamente:

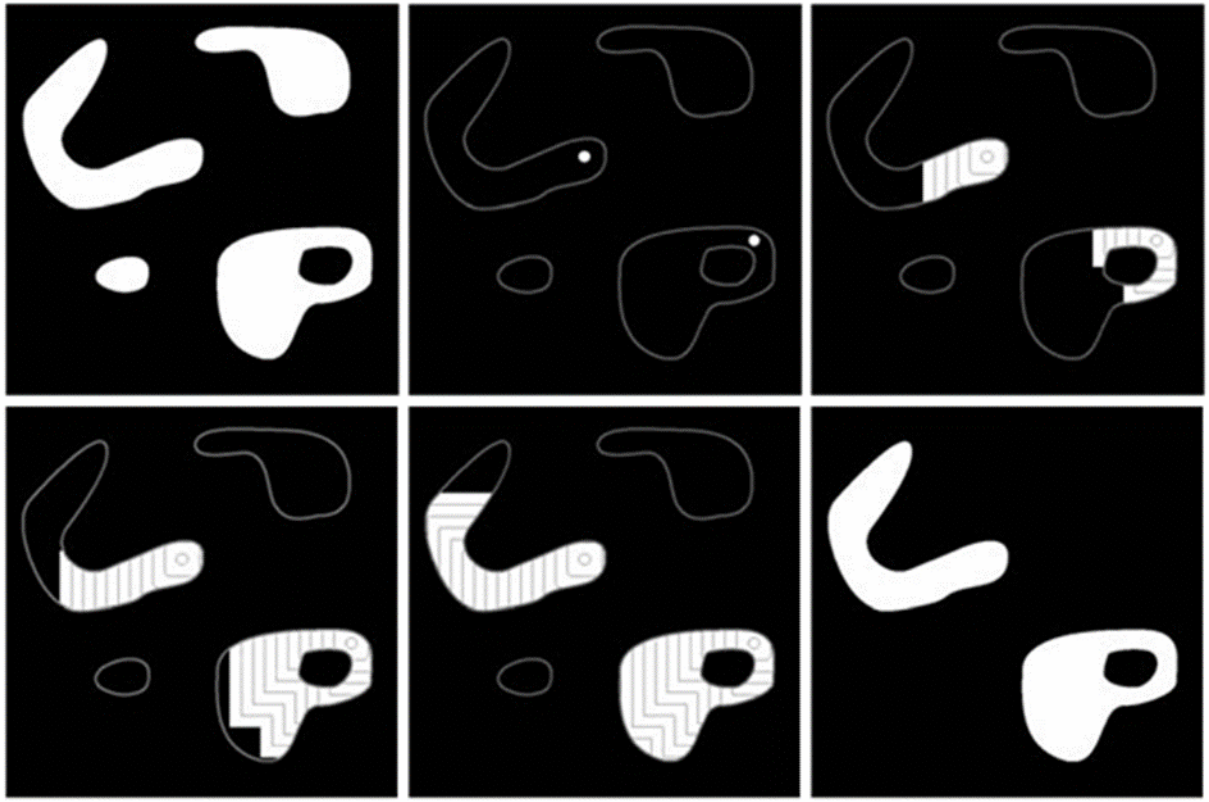
$$h_{k+1} = (h_k \oplus B) \cap g$$

hasta que $h_{k+1} = h_k$

Reconstrucción Morfológica

La función MATLAB que implementa esta operación es:

```
out = imreconstruct( marker , mask )
```



Reconstrucción mediante Apertura

En la Reconstrucción mediante Apertura, la erosión remueve los objetos pequeños y la subsecuente dilatación tiende a restaurar la forma de los objetos que no desaparecieron. Sin embargo, la exactitud de la restauración depende de la similitud entre los objetos y el elemento estructural.

La Reconstrucción mediante Apertura de f utilizando el objeto estructural B se define como $R_f (f \ominus B)$.

Las diferentes aplicaciones prácticas de la Reconstrucción mediante Apertura dependerán de la selección de la imagen marcador y la imagen máscara.

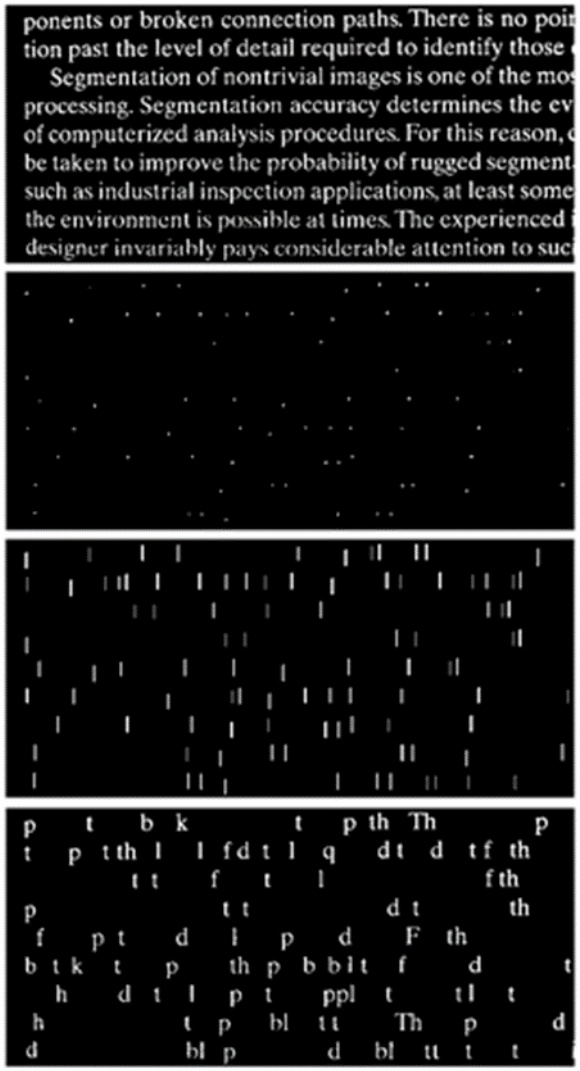
Reconstrucción mediante Apertura

Búsqueda de caracteres

Comparemos la apertura y la reconstrucción por apertura para una imagen donde queremos detectar los caracteres que tienen una línea vertical larga (h, l, n, p, etc.).

Como para la reconstrucción por apertura necesitamos una imagen marcador, la generamos esta a partir de una erosión de la imagen original.

```
>> f = imread('book_text_bw.tif');
>> fe = imerode( f , ones(51,1) );
>> fo = imopen( f , ones(51,1) );
>> fobr = imreconstruct( fe , f );
```



Reconstrucción mediante Apertura

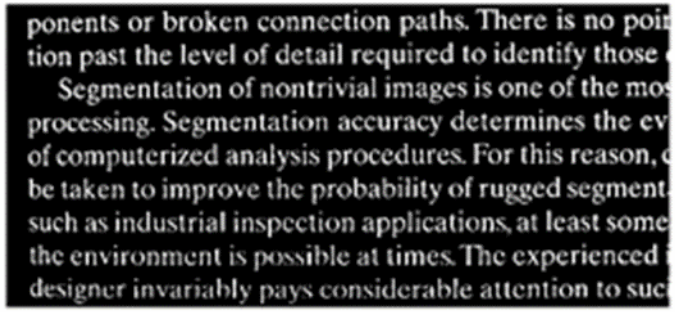
Rellenado de Huecos

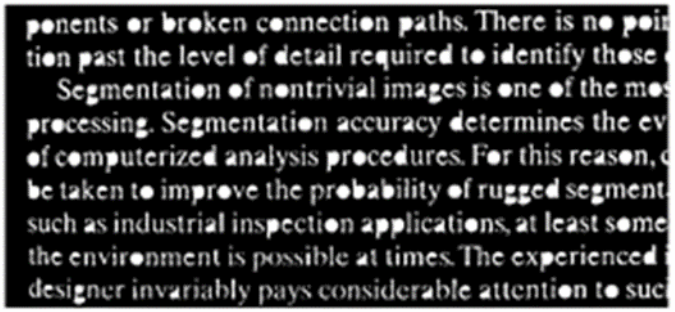
Si elegimos que la imagen marcador, f_m, sea 0 excepto en los bordes de la imagen donde se fuerza *1 – f*, entonces:

$$f_m(x,y) = \begin{cases} 1 - f(x,y) & si\ (x,y)\ pertenece\ al\ borde \\ 0 & c.o.c. \end{cases}$$

Entonces, $g = [R_f c(f_m)]^c$ tiene el efecto de rellenar los huecos de los caracteres que no tocan el borde.

```
>> f = imread('book_text_bw.tif');
>> g = imfill( f , 'holes' );
```





Reconstrucción mediante Apertura

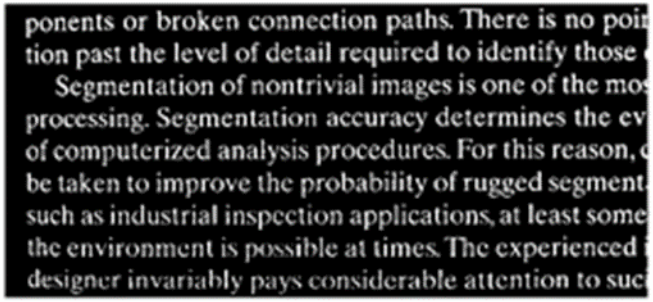
Eliminando objetos que tocan el borde

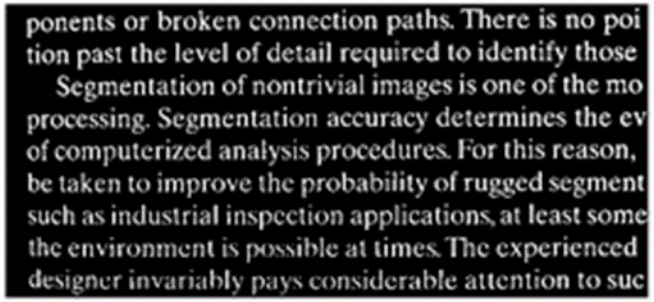
En este caso elegimos que la imagen marcador, f_m, coincida con la imagen f en los pixels 1 que se ubican en el borde, así:

$$f_m(x,y) = \begin{cases} f(x,y) & si\ (x,y)\ pertenece\ al\ borde \\ 0 & c.o.c. \end{cases}$$

Entonces, $g = R_f(f_m)$ contiene solo los caracteres que tocan el borde y la diferencia $f - R_f(f_m)$ contiene los elementos de la imagen original que no.

```
>> f = imread('book_text_bw.tif');
>> g = imclearborder( f , conn );
```





Análisis Morfológico en escala de grises

Dilatación en Escala de gris

La dilatación en escala de gris de f con el elemento estructural b, que denotamos $f \oplus b$, se define como

$$(f \oplus b)(x,y) = \max\{ f(x-x', y-y') + b(x',y') \mid (x',y') \in D_b \}$$

donde D_b es el dominio de b, y $f(x,y)$ se asume $-\infty$ fuera del dominio de f. Conceptualmente se rota y se traslada el elemento estructural a todos los pixeles de la imagen, y en cada ubicación los valores de los pixeles del elemento estructural se suman al valor de los pixeles de la imagen y se computa el máximo. La suma se realiza sólo para los pixeles $(x',y') \in D_b$ donde D_b tiene un valor 1.

Usualmente se utiliza un elemento estructural plano, por lo que $b(x',y') = 0$, para $(x',y') \in D_b$ y la ecuación de dilatación en escala de gris se simplifica a

$$(f \oplus b)(x,y) = \max\{ f(x-x', y-y') \mid (x',y') \in D_b \}$$

Erosión en Escala de gris

De manera similar se define erosión en escala de gris

$$(f \ominus b)(x,y) = \min\{ f(x-x', y-y') - b(x',y') \mid (x',y') \in D_b \}$$

que para el caso de un elemento estructural plano resulta

$$(f \ominus b)(x,y) = \min\{ f(x-x', y-y') \mid (x',y') \in D_b \}$$

Análisis Morfológico en escala de grises

```
>>se=strel('square',3);
>>gd=imdilate(f,se);
>>ge=imerode(f,se);
>>mgrad=imsubtract(gd,ge);
>>figure, imshow(f,[])
>>figure, imshow(gd,[])
>>figure, imshow(ge,[])
>>figure, imshow(mgrad,[])
```

a b
c d

FIGURE 9.23 Dilation and erosion. (a) Original image. (b) Dilated image. (c) Eroded image. (d) Morphological gradient. (Original image courtesy of NASA.)

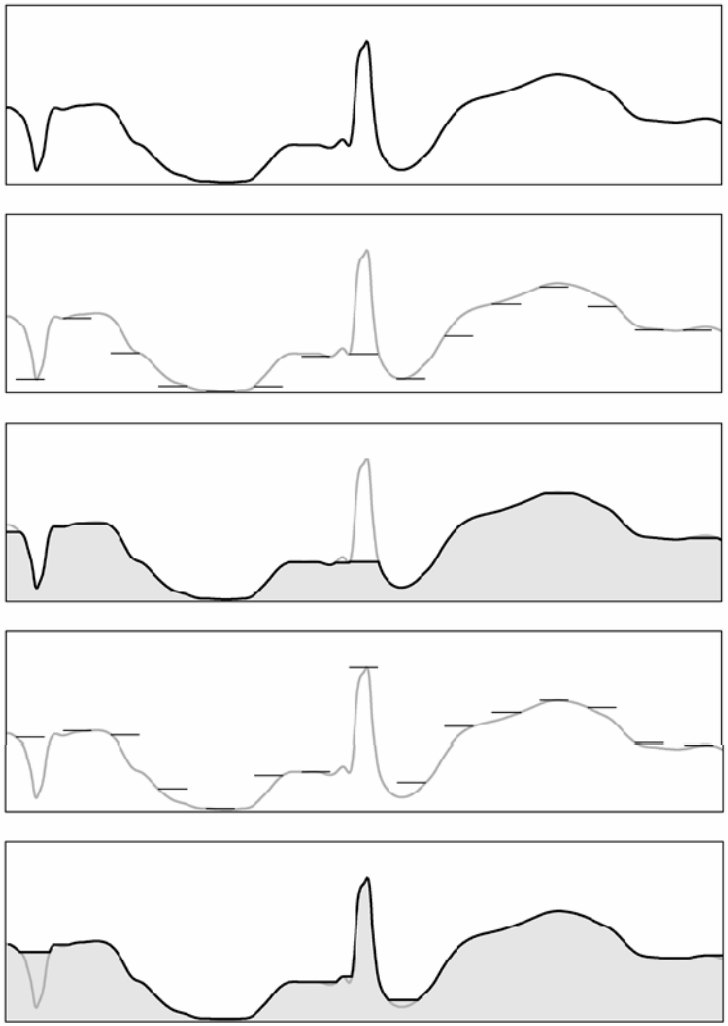
Análisis Morfológico en escala de grises

Apertura en Escala de gris

$$f \circ b = \left(f \ominus b \right) \oplus b$$

Clausura en Escala de gris

$$f \bullet b = \left(f \oplus b \right) \ominus b$$

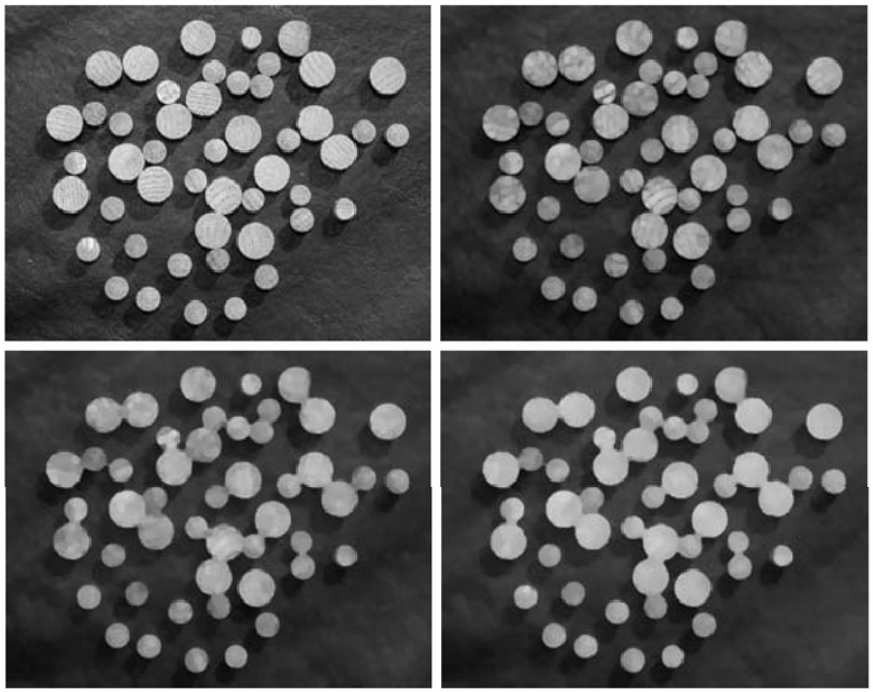


a
b
c
d
e

FIGURE 9.24 Opening and closing in one dimension. (a) Original 1-D signal. (b) Flat structuring element pushed up underneath the signal. (c) Opening. (d) Flat structuring element pushed down along the top of the signal. (e) Closing.

Análisis Morfológico en escala de grises

```
>>f=imread('plugs.jp');
>>se=strel('disk',5);
>>fo=imopen(f,se);
>>foc=imclose(fo,se);
```



a b
c d

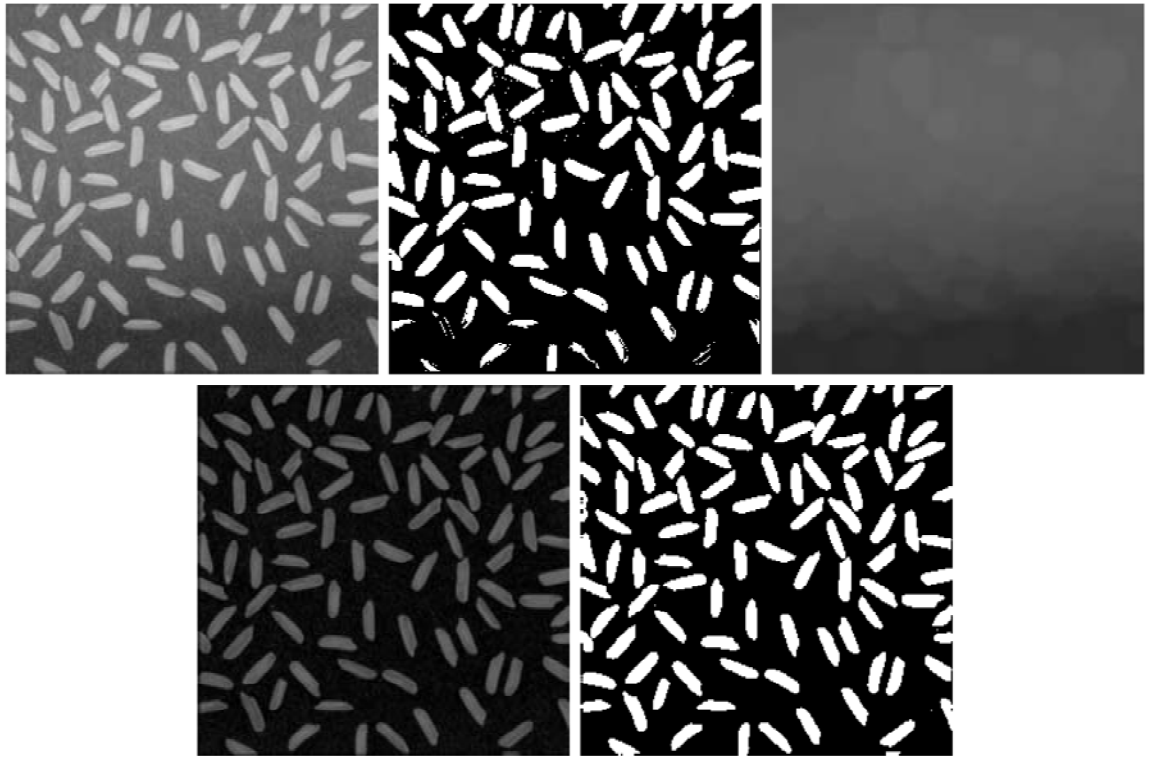
FIGURE 9.25
Smoothing using openings and closings.
(a) Original image of wood dowel plugs. (b) Image opened using a disk of radius 5.
(c) Closing of the opening.
(d) Alternating sequential filter result.

Filtro secuencial alternante

```
>>fasf=f;
>>for k=2:5
    se=strel(`disk',k);
    fasf=imclose(imopen(fasf,se),se);
  end
```

Análisis Morfológico en escala de grises

Puede obtenerse una imagen con un **background uniforme** substrayendo la apertura de la imagen de la imagen original. Esta operación se denomina **Transformación top-hat**



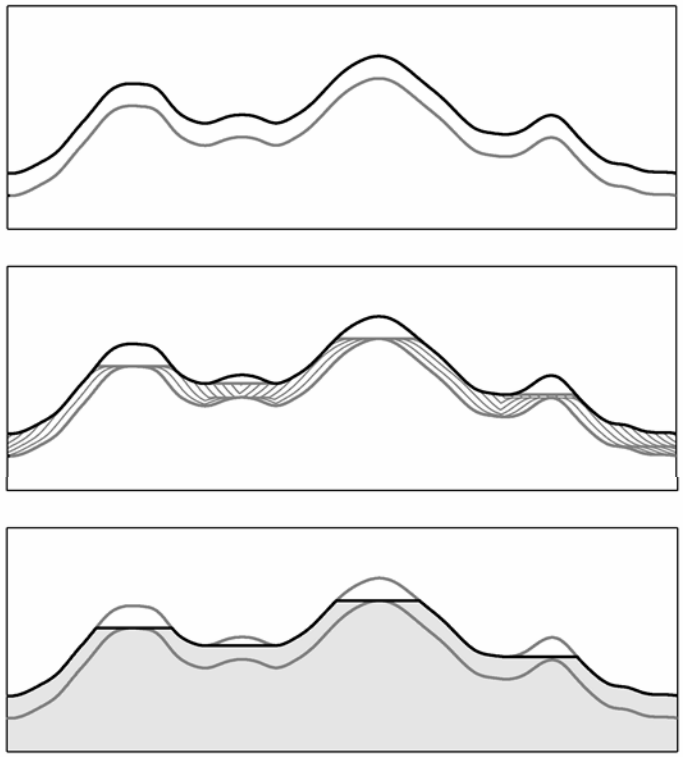
a b c
d e

FIGURE 9.26 Top-hat transformation. (a) Original image. (b) Thresholded image. (c) Opened image. (d) Top-hat transformation. (e) Thresholded top-hat image. (Original image courtesy of The MathWorks, Inc.)

Análisis Morfológico en escala de grises

Reconstrucción en Escala de gris

Mismo procedimiento que para imágenes binarias.



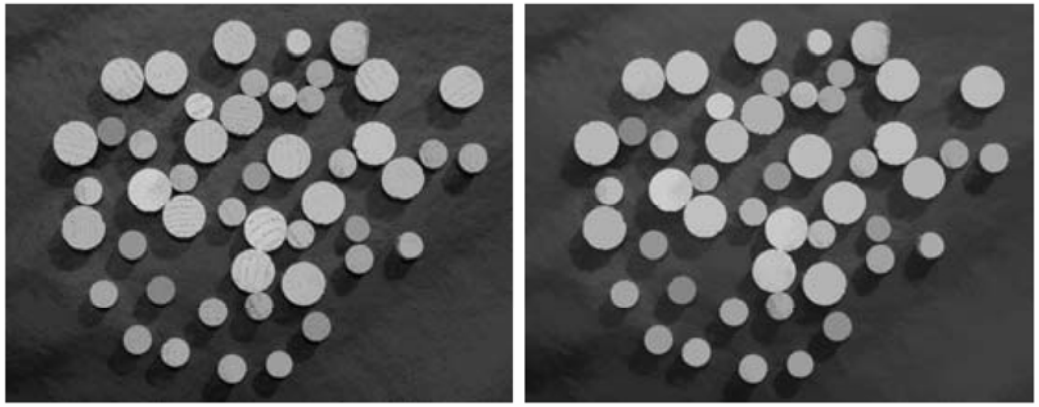
a
b
c

FIGURE 9.28 Gray-scale morphological reconstruction in one dimension. (a) Mask (top) and marker curves. (b) Iterative computation of the reconstruction. (c) Reconstruction result (black curve).

Análisis Morfológico en escala de grises

Apertura por Reconstrucción

```
>>f=imread('plugs.jp');
>>se=strel('disk',5);
>>fe=imerode(f,se);
>>fobr=imreconstruct(fe,f);
```



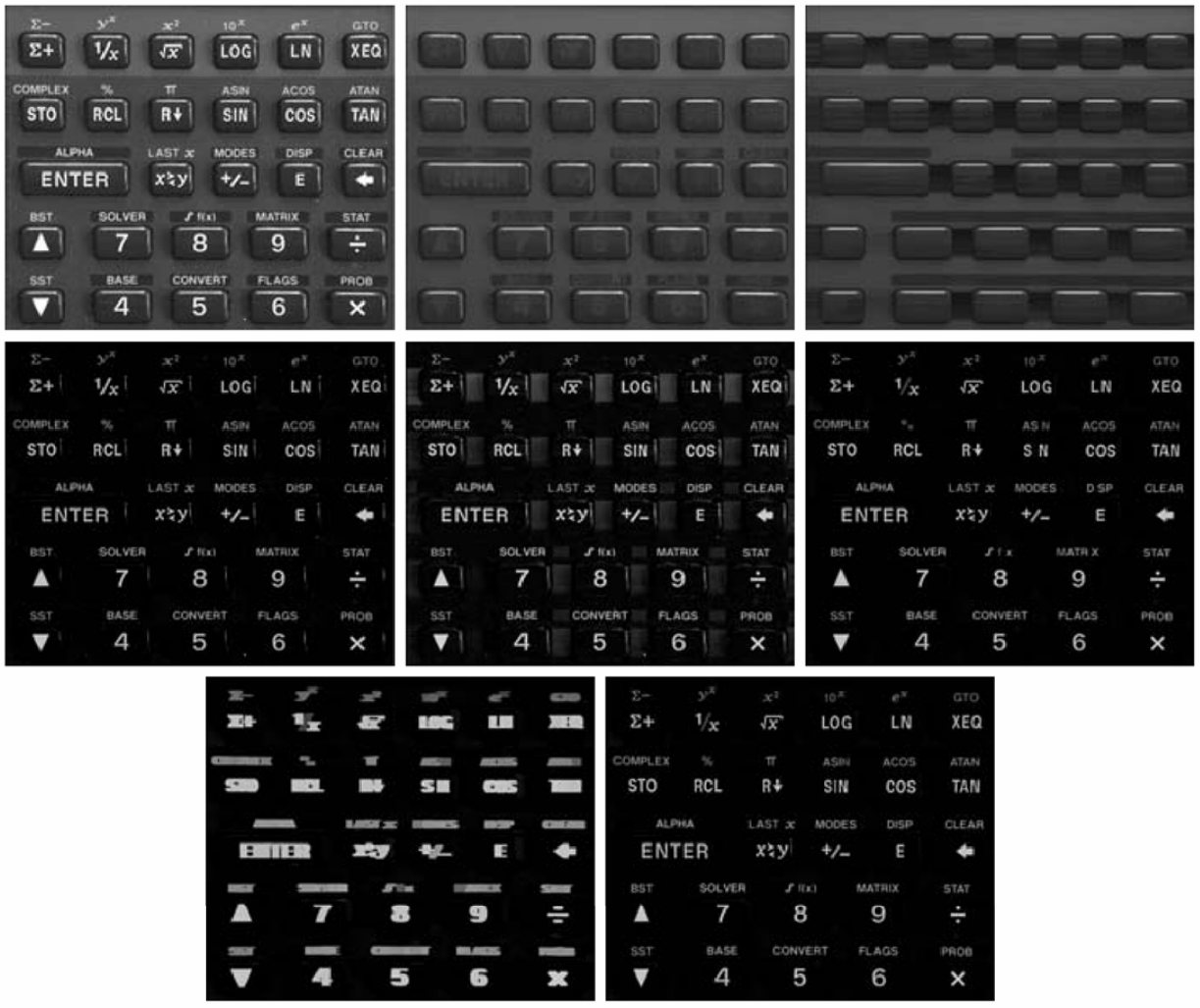
a b

FIGURE 9.29
(a) Opening-by-reconstruction. (b) Opening-by-reconstruction followed by closing-by-reconstruction.

Clausura por Reconstrucción

```
>>fobrc=imcomplement(fobr);
>>fobrce=imerode(fobrc,se);
>>fobrcbe=imcomplement(imreconstruct(fobrce,fobrc));
```

Análisis Morfológico en escala de grises



a b c
d e f
g h

FIGURE 9.30 An application of gray-scale reconstruction. (a) Original image. (b) Opening-by-reconstruction. (c) Opening. (d) Tophat-by-reconstruction. (e) Tophat. (f) Opening-by-reconstruction of (d) using a horizontal line. (g) Dilation of (f) using a horizontal line. (h) Final reconstruction result.