

# Procesamiento de Imágenes

## UNIDAD 6 - 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.

# Análisis Morfológico

La morfología matemática se basa en operaciones de teoría de conjuntos.

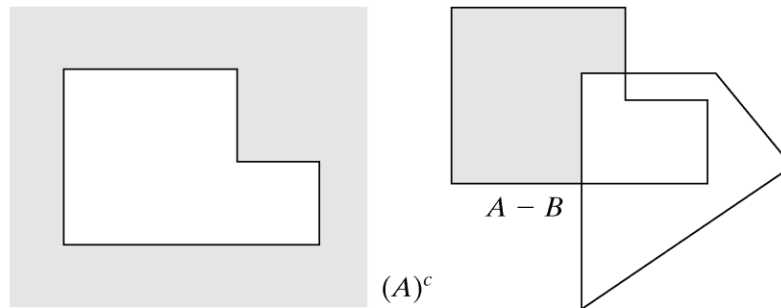
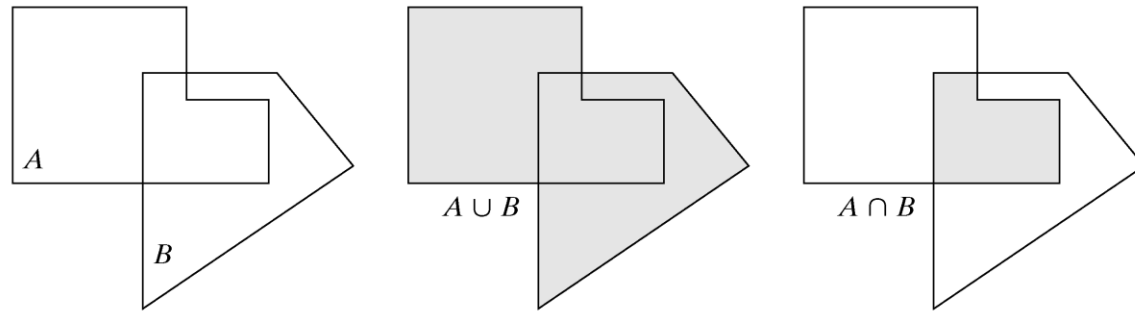
En el caso de imágenes binarias, los conjuntos tratados son subconjuntos de  $Z^2$  y en el de las imágenes en escala de grises, se trata de conjuntos de puntos con coordenadas en  $Z^3$ .

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:

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



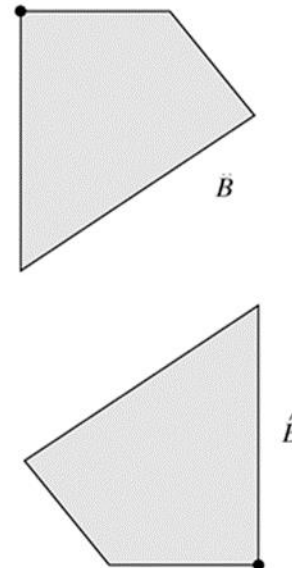
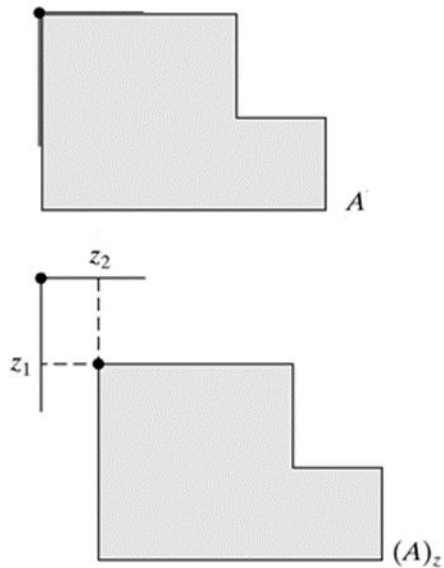
$$A - B = \{w / w \in A \wedge w \notin B\}$$

# Operaciones básicas de conjuntos

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

$$A_z = \{c / c = a + z \quad \forall a \in A\}$$

$$\hat{B} = \{w / w = -b \quad \forall b \in B\}$$



# Operaciones básicas de conjuntos

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

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



# Dilatación

Esta operación “**augmenta**” 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.

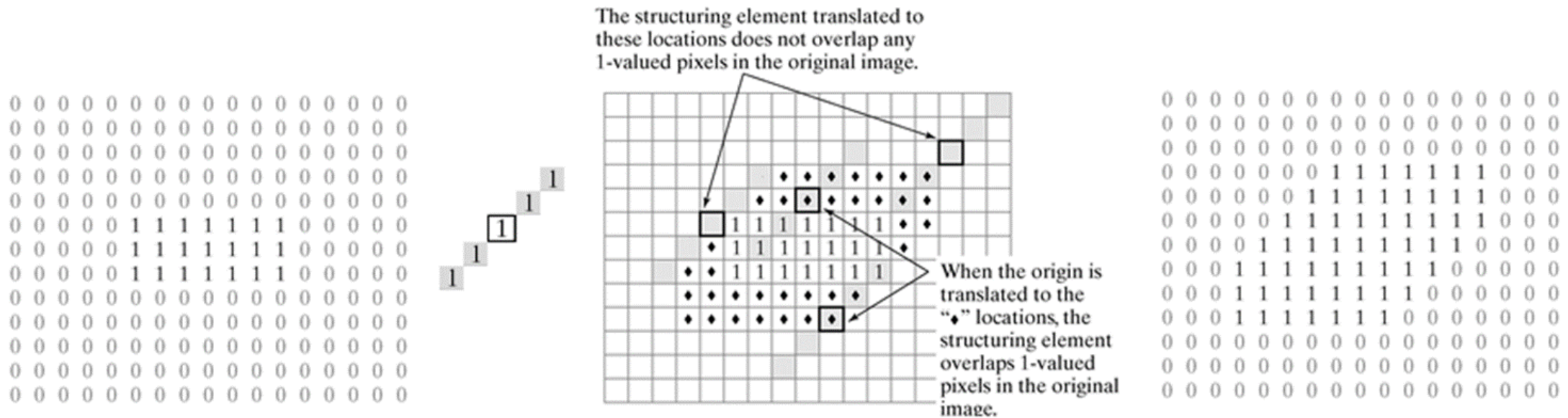
El elemento estructural es por lo general mucho más pequeño que la imagen.

Matemáticamente,

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

# Dilatación

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



```
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (L,L))  
Fd = cv2.dilate(F, kernel, iterations=1)
```



# Dilatación

```
F = cv2.imread('broken_text.tif')
kernel = np.array([[0,1,0],[1,1,1],[0,1,0]],np.uint8)
Fd = cv2.dilate(F, kernel, iterations=1)
imshow(Fd)
```

Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.

Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.

Elemento  
Estructural

0	1	0
1	<b>1</b>	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:

$$B = (B_1 \oplus B_2) \Rightarrow A \oplus B = A \oplus (B_1 \oplus B_2) = (A \oplus B_1) \oplus B_2$$

Es decir, la operación de dilatación  $A \oplus B$  se puede descomponer en dos dilataciones sucesivas con los elementos estructurales  $B_1$  y  $B_2$ .

En ciertos casos, computacionalmente esto último reduce sustancialmente la cantidad de cálculos a realizar.

# Elementos Estructurales

```
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (L,L))
```

Enumerator	
MORPH_RECT Python: cv.MORPH_RECT	a rectangular structuring element: $E_{ij} = 1$
MORPH_CROSS Python: cv.MORPH_CROSS	a cross-shaped structuring element: $E_{ij} = \begin{cases} 1 & \text{if } i = \text{anchor.y or } j = \text{anchor.x} \\ 0 & \text{otherwise} \end{cases}$
MORPH_ELLIPSE Python: cv.MORPH_ELLIPSE	an elliptic structuring element, that is, a filled ellipse inscribed into the rectangle Rect(0, 0, esize.width, 0.esize.height)

# Erosión

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

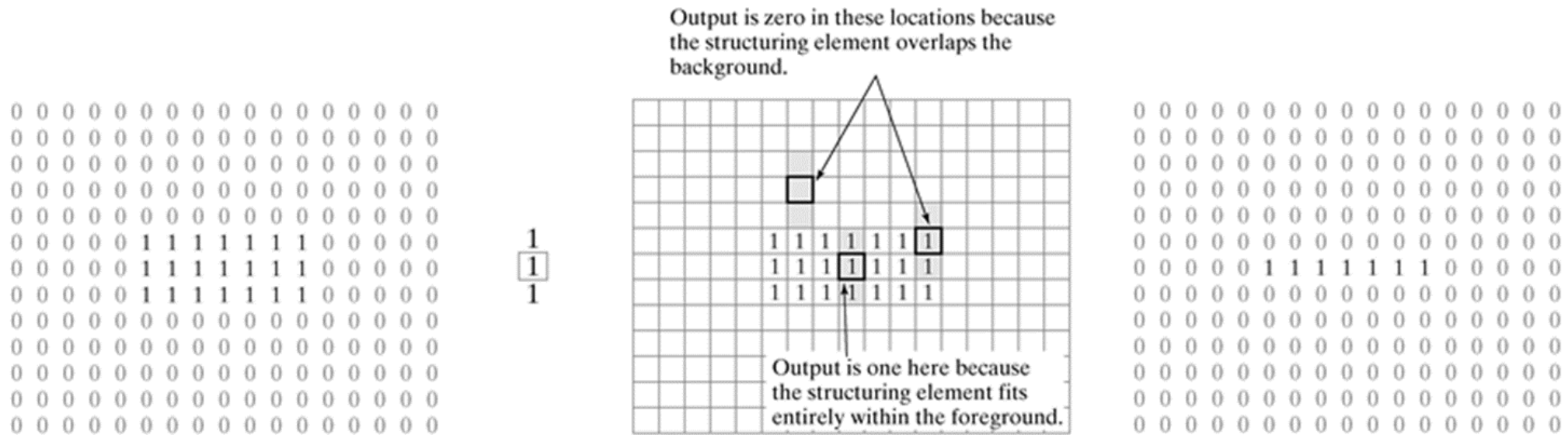
El elemento estructural tiene claramente identificado su origen. Así la erosión consiste en trasladar el elemento estructural a lo largo de toda la imagen y ver si el mismo queda contenido completamente en la zona de la imagen con valores 1. Si esto sucede el resultado de la erosión tendrá un valor 1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

Matemáticamente:

$$A \ominus B = \{z | (B)_z \subseteq A\}$$

# Erosión

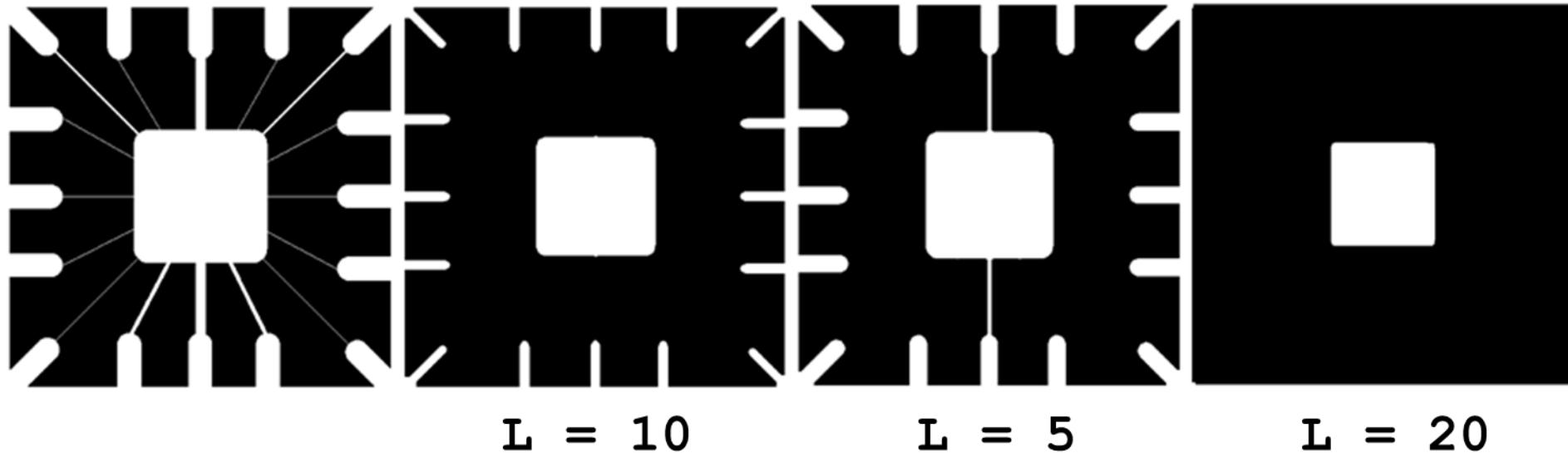
En otras palabras la erosión de A con B es el conjunto de las ubicaciones del origen del elemento estructural tal que el elemento se solape completamente con los pixels de valor 1 de la imagen original.



```
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (L,L))  
Fe = cv2.erode(F, kernel, iterations=1)
```

# Erosión

```
F = cv2.imread('wirebond_mask.tif')  
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (L,L))  
Fe = cv2.erode(A, kernel, iterations=1)  
imshow(Fe)
```

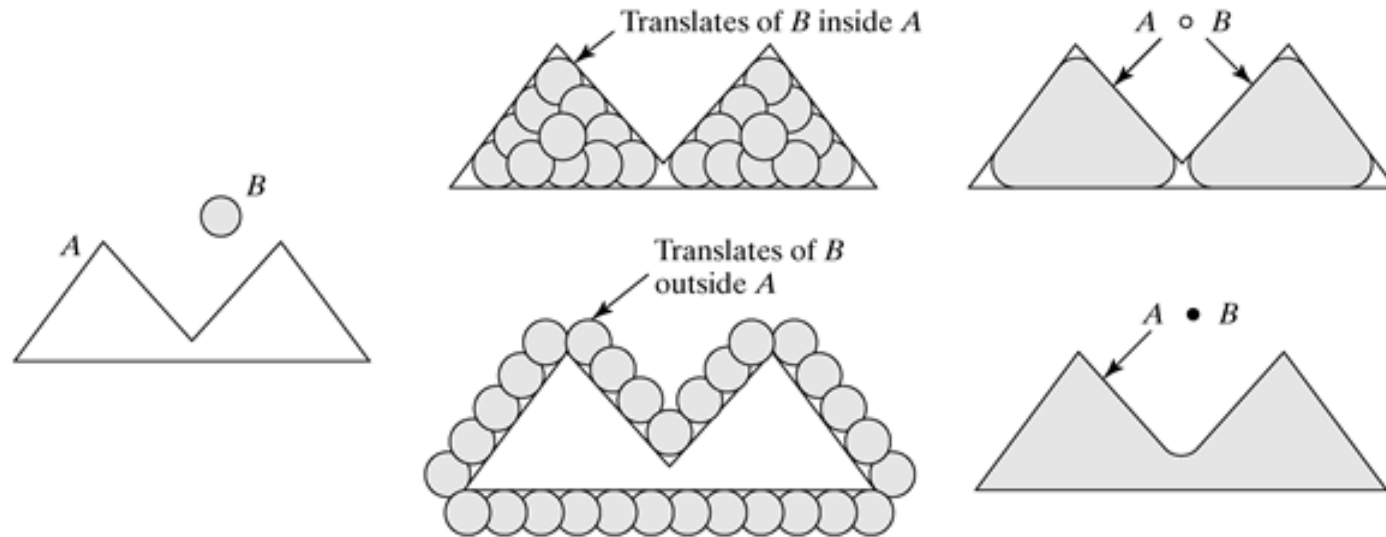


# Apertura y Clausura

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

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

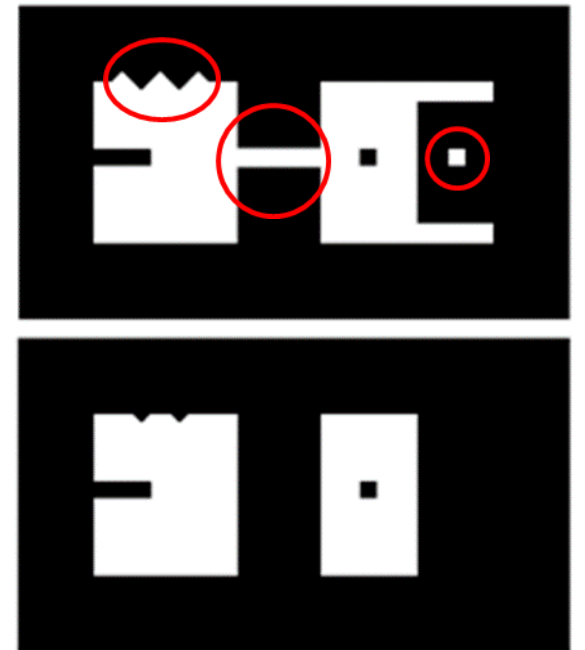
$$\text{Clausura} \rightarrow A \bullet B = (A \oplus B) \ominus B$$



# 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.

```
A = cv2.imread('shapes.tif')  
B = cv2.getStructuringElement(cv2.MORPH_RECT, (50,50))  
Aop = cv2.morphologyEx(A, cv2.MORPH_OPEN, B)  
imshow(Aop)
```

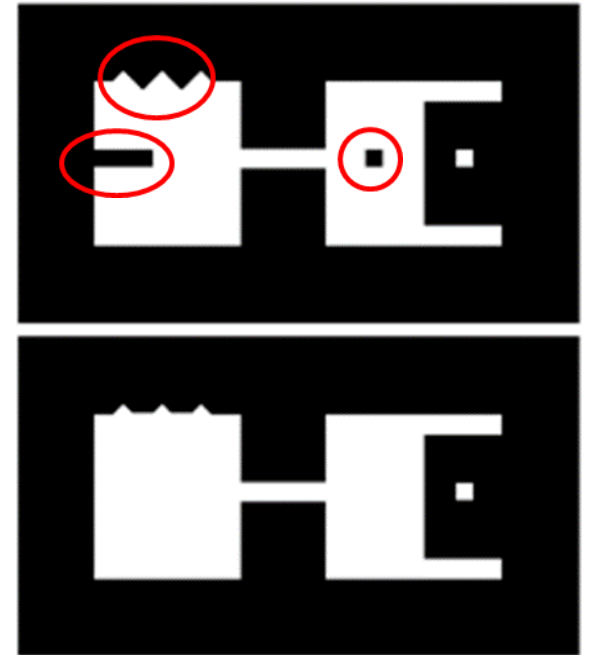




# Clausura

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.

```
A = cv2.imread('shapes.tif')  
B = cv2.getStructuringElement(cv2.MORPH_RECT, (50,50))  
Aclau = cv2.morphologyEx(A, cv2.MORPH_CLOSE, B)  
imshow(Aclau)
```



# Combinando Apertura y Clausura

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

```
f = cv2.imread('fingerprint.tif')
se = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))
fop = cv2.morphologyEx(f, cv2.MORPH_OPEN, se)
fop_cls = cv2.morphologyEx(fop, cv2.MORPH_CLOSE, se)
imshow(fop)
imshow(fop_cls)
```

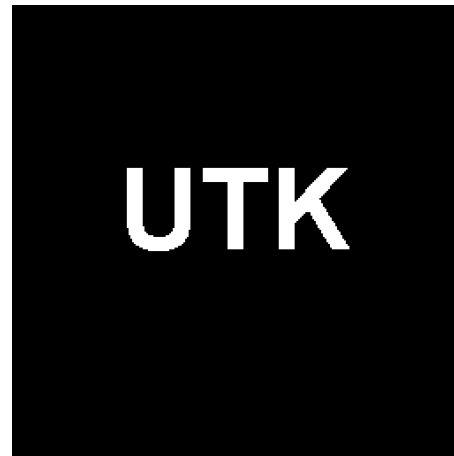


# Gradiente Morfológico

Para obtener el gradiente morfológico, primero se aplica una erosión y una dilatación individualmente sobre la imagen, y luego se calcula la diferencia entre la imagen dilatada y erosionada. La salida obtenida será un “**contorno**” de la imagen original, cuyo grosor dependerá del elemento estructural utilizado.

```
f = cv2.imread('UTK.tif', cv2.IMREAD_GRAYSCALE)
se = cv2.getStructuringElement(cv2.MORPH_RECT, (3,3))
f_mg = cv2.morphologyEx(f, cv2.MORPH_GRADIENT, se)
imshow(f)
imshow(f_mg)
```

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



# 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 blancos de B y  $B_2$  el conjunto formado por los pixels blancos de  $B^c$ .

B

0	1	0
1	1	1
0	1	0

$B_1$

0	1	0
1	1	1
0	1	0

$B_2$

1	0	1
0	0	0
1	0	1

PYTHON: valores de B

- 1 ➡ donde debe haber HIT.
- -1 ➡ donde debe haber MISS.
- 0 ➡ da igual.

```
C = cv2.morphologyEx(F, cv2.MORPH_HITMISS, B)
```

# Transformación Hit-or-Miss

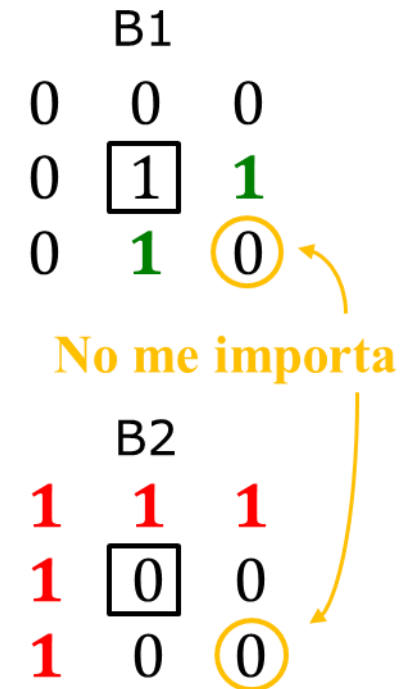
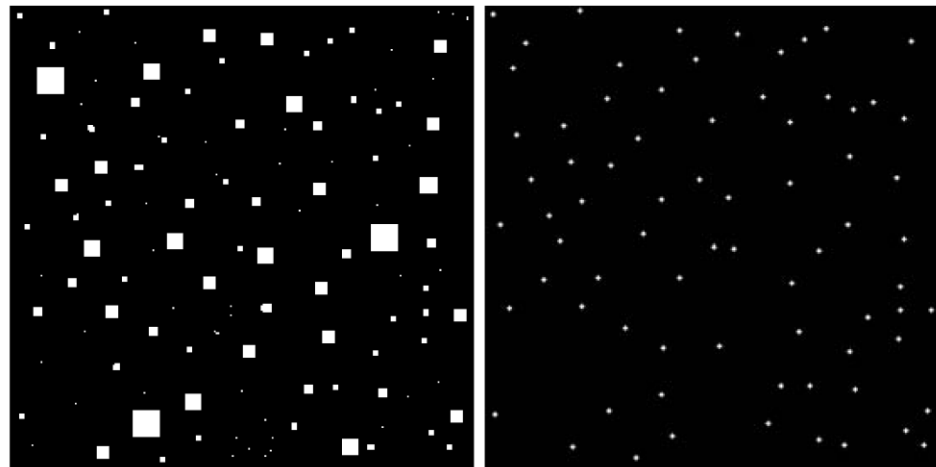
$$A \otimes B = (A \ominus B_1) \cap (A^c \ominus B_2)$$

Figure 1 illustrates the construction of a 2D array  $A$  from two 1D arrays  $A^C$  and  $A^D$ . The arrays are represented as grids of 0s and 1s. The construction is shown as  $A = A^C \oplus B_1 \oplus B_2$ , where  $B_1$  and  $B_2$  are 16x16 grids of 0s and 1s. The construction is shown as  $A = A^C \oplus B_1 \oplus B_2$ .

# 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 = cv2.imread('squares.tif', cv2.IMREAD_GRAYSCALE)
B = np.array([[ -1, -1, -1], [-1, 1, 1], [-1, 1, 0]])
g = cv2.morphologyEx(f.copy(), cv2.MORPH_HITMISS, B)
```



# Skeleton

En muchas aplicaciones de visión por ordenador a menudo tenemos que tratar con enormes cantidades de datos: el procesamiento puede ser, por tanto, lento y requerir mucha memoria.

Para conseguir un procesamiento más rápido y un menor consumo de memoria, a veces utilizamos una representación más compacta llamada **esqueleto**. Otras veces, utilizamos el esqueleto para analizar la “forma” intrínseca de un objeto, es decir, su **estructura interna**.

Un esqueleto debe conservar la estructura de la forma, pero deben eliminarse todos los píxeles redundantes.





# Skeleton

Imagen Original



Iteración 1



Iteración N  
no hay mas cambios





# 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.

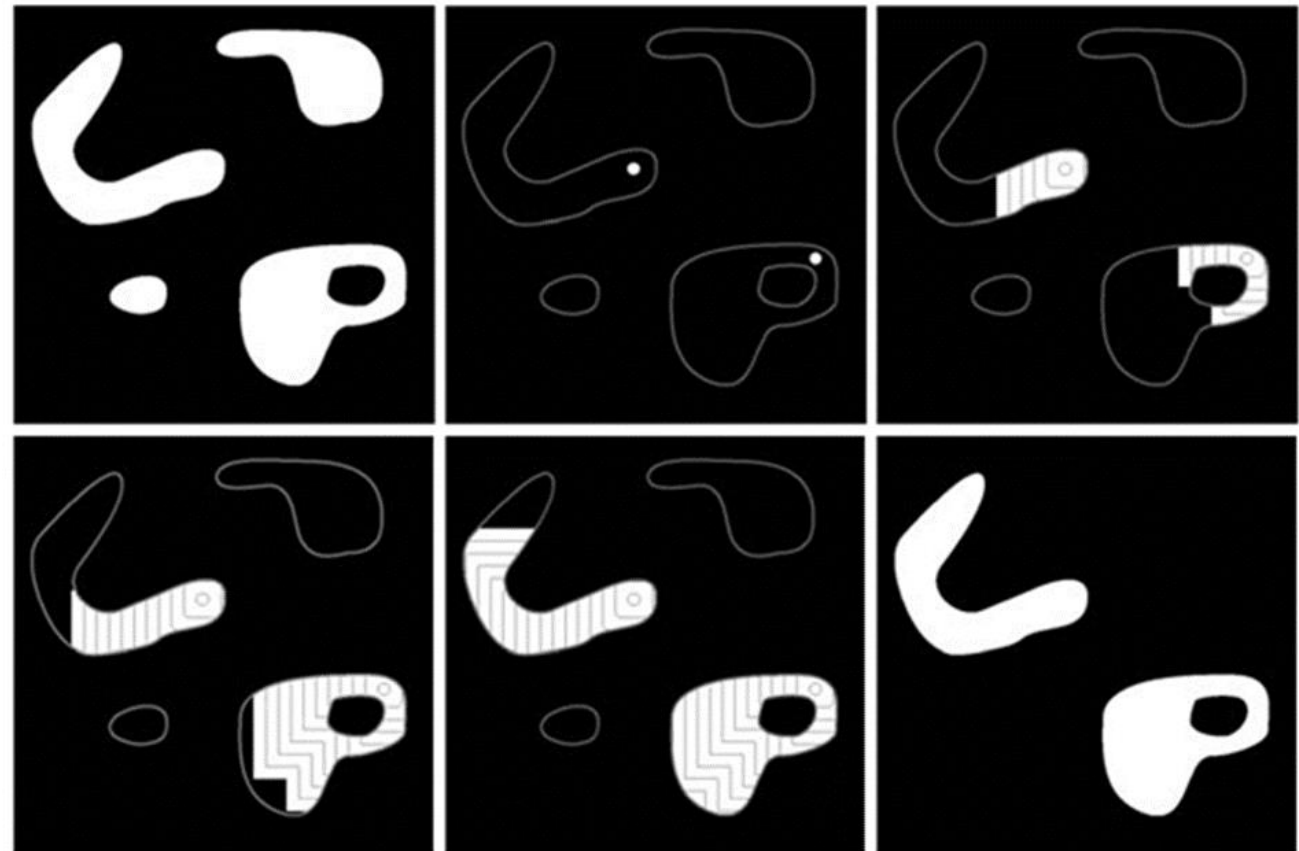
La reconstrucción de  $g$  a partir de  $f$ ,  $R_g(f)$ , se define a partir de un proceso iterativo.

# Reconstrucción Morfológica

1. Inicializamos  $h_1$  como la imagen marcador  $f$ , que debe ser un subconjunto de  $g$  ( $f \subseteq g$ ).
2. Creamos un elemento estructural. Por ejemplo,  $B = \text{np.ones}((3,3))$
3. Realizamos la siguiente operación iterativamente:

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

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



# Apertura por Reconstrucción

En la 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.

El método de Apertura por Reconstrucción, restaura exactamente las formas de los objetos que quedan luego de la erosión. La Apertura por Reconstrucción de  $f$  utilizando el objeto estructural  $B$  se define como  $R_f(f \ominus B)$ .

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

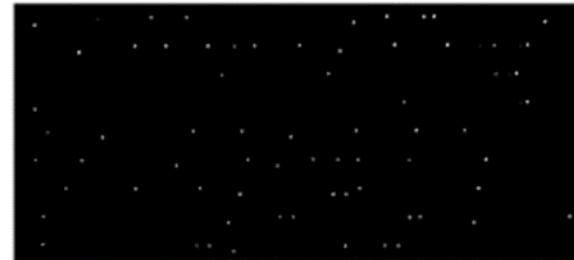
# Apertura por Reconstrucción

## Búsqueda de caracteres

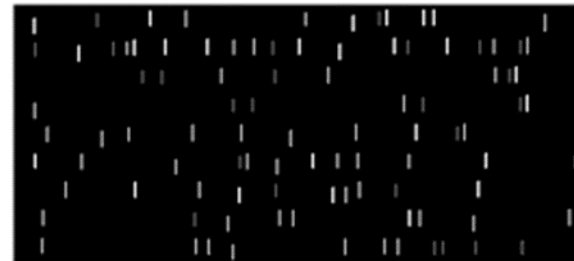
Comparemos la apertura y la apertura por reconstrucción para una imagen donde queremos detectar los caracteres que tienen una línea vertical larga (h, l, n, p, etc.).

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

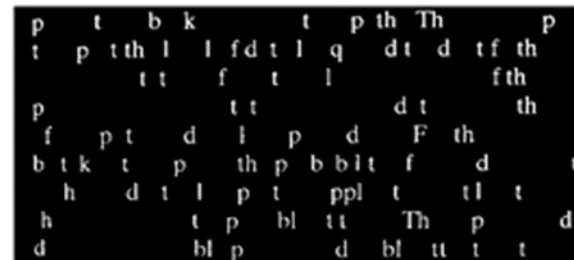
ponents or broken connection paths. There is no position past the level of detail required to identify those  
Segmentation of nontrivial images is one of the most difficult problems in image processing. Segmentation accuracy determines the effectiveness of computerized analysis procedures. For this reason, considerable effort must be taken to improve the probability of rugged segmentation. In applications such as industrial inspection applications, at least some degree of segmentation is possible at times. The experienced designer invariably pays considerable attention to such



Erosión



Apertura



Apertura por Reconstrucción

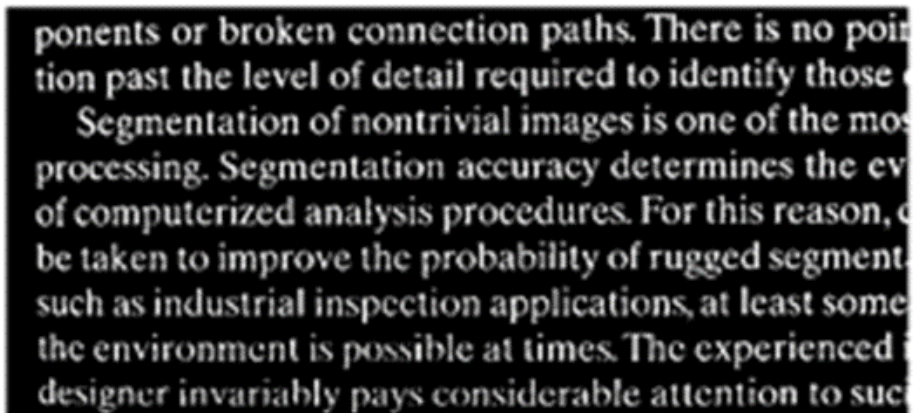
# Reconstrucción mediante Apertura

## Rellenado de Huecos

Si elegimos que la imagen marcador  $f_m$  sea 0 excepto en los bordes de la imagen donde se fuerza a que valga  $1 - f(x,y)$  (o  $255 - f(x,y)$ ), entonces:

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

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



# 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:

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

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.

ponents or broken connection paths. There is no position past the level of detail required to identify those elements.

Segmentation of nontrivial images is one of the most difficult tasks in image processing. Segmentation accuracy determines the effectiveness of computerized analysis procedures. For this reason, considerable effort must be taken to improve the probability of rugged segmentation, such as industrial inspection applications, at least some of the time. The environment is possible at times. The experienced designer invariably pays considerable attention to such

ponents or broken connection paths. There is no position past the level of detail required to identify those elements.

Segmentation of nontrivial images is one of the most difficult tasks in image processing. Segmentation accuracy determines the effectiveness of computerized analysis procedures. For this reason, considerable effort must be taken to improve the probability of rugged segmentation, such as industrial inspection applications, at least some of the time. The environment is possible at times. The experienced designer invariably pays considerable attention to such



# Análisis Morfológico en 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 píxeles de la imagen, y en cada ubicación los valores de los píxeles del elemento estructural se suman al valor de los píxeles de la imagen y se computa el máximo. La suma se realiza sólo para los píxeles  $(x', y') \in D_b$  donde  $D_b$  tiene un valor 1.

# Análisis Morfológico en grises

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 escalas de grises 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 grises



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.)

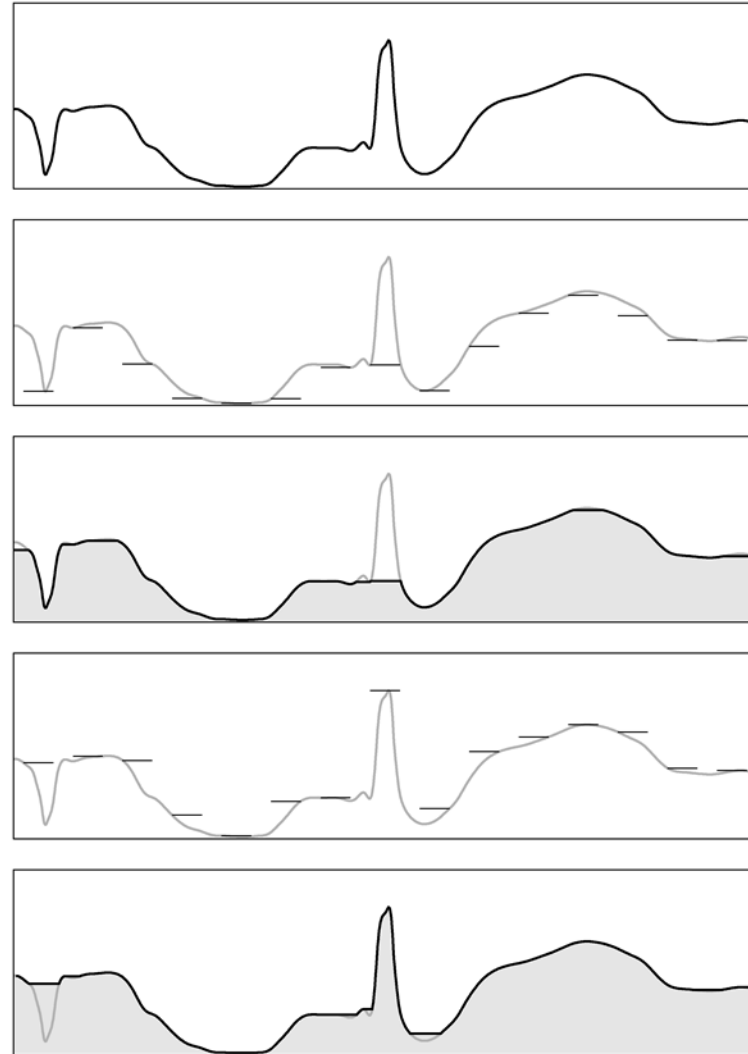
# Análisis Morfológico en grises

Apertura en escala de gris

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

Clausura en escala de gris

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

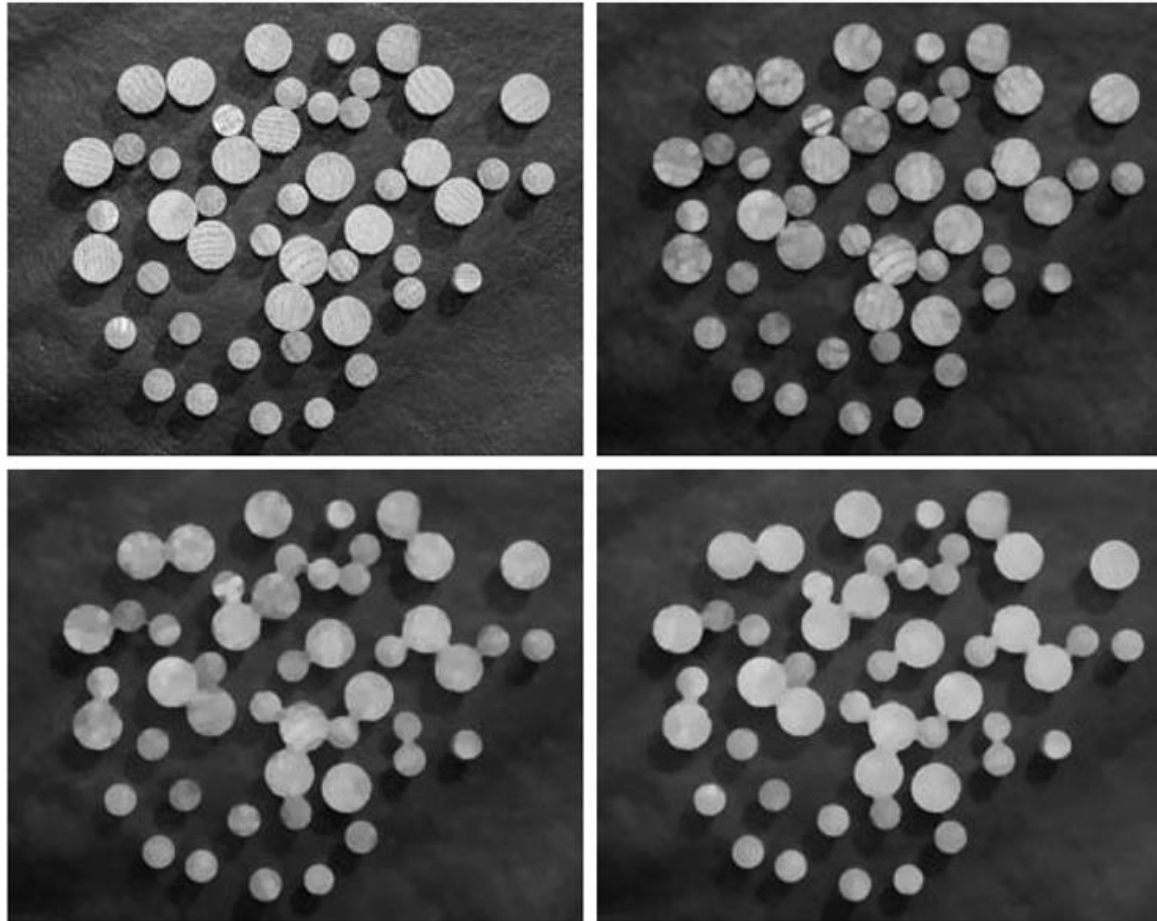


a  
b  
c  
d  
e

**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 grises

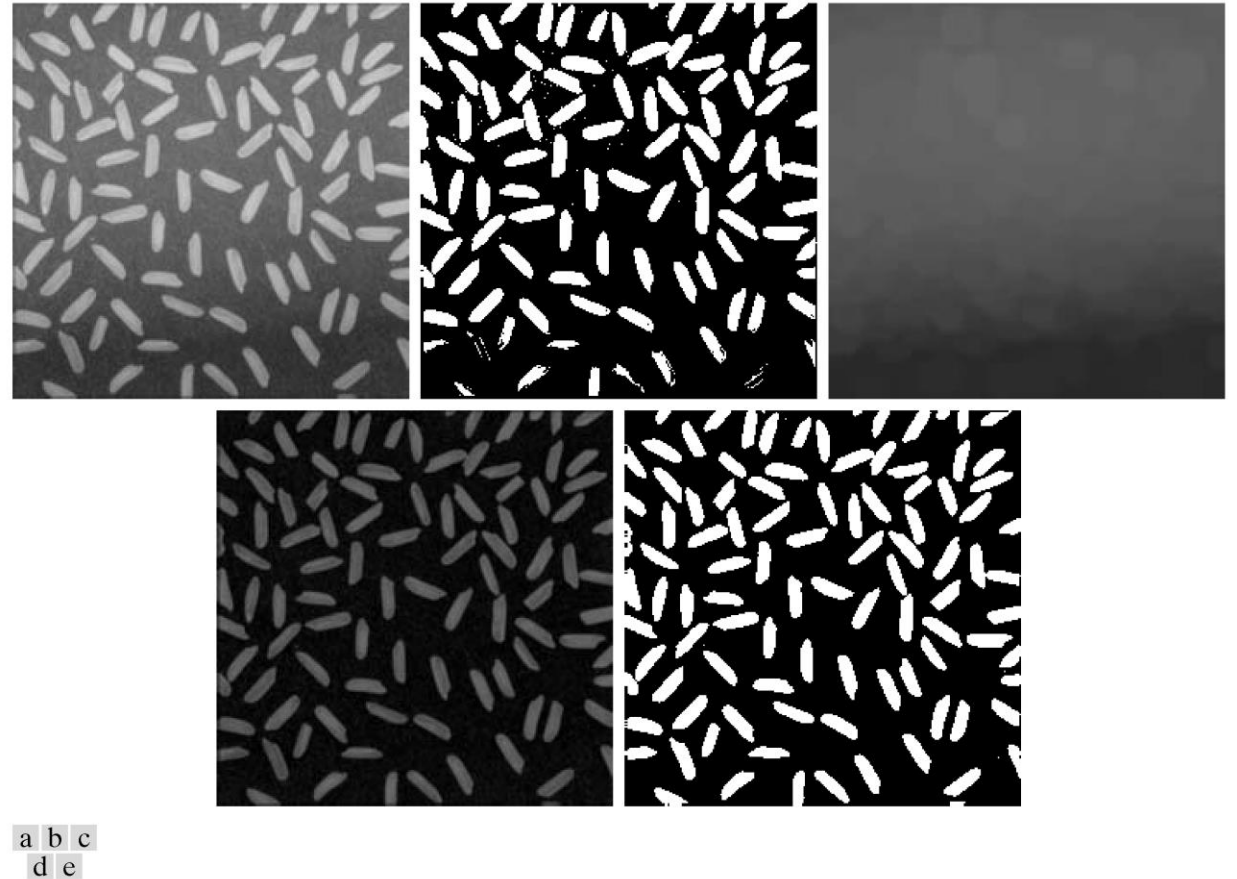


a	b
c	d

**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.

# Análisis Morfológico en grises

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



**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.)