

# Segmentación de imágenes



Procesamiento de imágenes  
biomédicas  
Curso 2017

# Introducción

---

- Hasta ahora el procesamiento digital de una imagen implicaba, una imagen de entrada y una imagen de salida.
- Ahora con segmentación tenemos una imagen de entrada y la salida de este procesamiento son atributos extraídos de la imagen.

# Segmentación de imágenes

---

- La segmentación de imágenes divide la imagen en sus partes constituyentes hasta un nivel de subdivisión en el que se aíslan las regiones u objetos de interés.
- Los algoritmos de segmentación se basan en una de estas dos propiedades básicas de los valores del nivel de gris: discontinuidad o similitud entre los niveles de gris de píxeles vecinos.



Diseño de las propiedades de captura. Cámara, megapixels..



Reducir el entorno no deseado. Fondo, ruido...



Reconocer y extraer c/u de los objetos presentes en la imagen



Extraer las características apropiadas para la identificación de objetos.



Utilizar un modelo para la toma de decisión respecto de la categoría del objeto.

Clasificadores

Entrenamiento

# Ejemplo

---

- Problema: identificar frutas

Captura

Decidir como van a ser capturadas las imágenes, distancia, 800x600, 24 bits...

Pre-procesamiento

Quitar el fondo y dejar la fruta.

Segmentación

Utilizar algún operador de segmentación para extraer las frutas de la imagen.

Extracción de características

Para cada fruta se va a extraer la longitud y el índice de circularidad.

Identificación de objetos

Utilizar algún algoritmo de clasificación para saber que tipo de fruta es.

- 
- ❑ Los algoritmos de segmentación de imágenes monocromáticas se basan en alguna de las siguientes propiedades :
  - ❑ discontinuidad en los tonos de gris de los pixeles en el entorno que permite detectar puntos aislados, líneas y bordes. Es una construcción de regiones basada en fronteras.
  - ❑ similaridad de los tonos de gris de los pixeles en un entorno, que permite construir regiones por división y fusión.

---

➤ *Discontinuidad*: Se divide la imagen basándose en cambios bruscos de nivel de gris:

- ❖ Detección de puntos aislados
- ❖ Detección de líneas
- ❖ Detección de bordes



---

➤ *Similitud*: se divide la imagen basándose en la búsqueda de zonas que tengan valores similares, conforme a unos criterios prefijados:

- ❖ Umbralización

- ❖ Crecimiento de región

# Procesamiento de Imágenes

---

- Análisis de Imágenes
- Segmentación de Imágenes
  - Segmentación basada en características
  - Segmentación basada en transiciones
  - Segmentación basada en modelos
  - Segmentación basada en homogeneidad
- Etiquetado

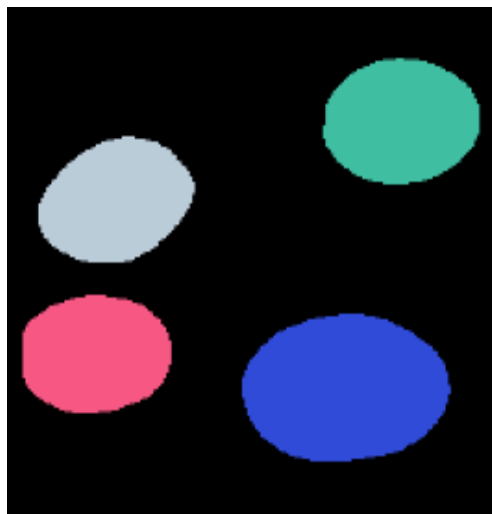
---

## □ Segmentación:

- División de la imagen en regiones con características similares.
- Cada una de las regiones de interés (que comparten ciertas propiedades) se denomina objeto.
- Resultado de la segmentación: separación de objetos.

## □ Etiquetado:

- Para diferenciar los objetos, éstos tendrán asignadas unas etiquetas.



# Segmentación de imágenes

---

- ❑ Las técnicas de segmentación son muy dependientes del propósito de la aplicación y del tipo de imágenes a analizar.
- ❑ Antes de segmentar es preciso definir qué objetos interesa determinar
- ❑ Tras la segmentación es posible realizar operaciones de filtrado (a nivel de objetos), así como determinar características que permitan clasificar los objetos.

# Segmentación de imágenes

---

- Una buena segmentación es difícil de evaluar.
- Fundamentalmente, lo que se busca es que diferentes objetos tengan valores claramente diferentes de la(s) característica(s) discriminante(s).

# Tipos de segmentación

## □ Segmentación basada en transiciones

---

- Detección de bordes

## □ Segmentación basada en modelos

- Transformada de Hough

## □ Segmentación basada en homogeneidad

- Fusión de regiones

## □ Segmentación basada en Morfología Matemática

## □ Segmentación basada en características

- Segmentación por niveles de gris
- Segmentación de imágenes en color
- Segmentación por textura

# Detección de discontinuidades: puntos aislados

---

Un punto aislado de una imagen tiene un tono de gris que difiere significativamente de los tonos de gris de sus píxeles vecinos, es decir, de los ocho píxeles de su entorno 3×3. Una máscara (Laplaciano) para detectar un punto aislado es la siguiente:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Diremos que un píxel es un punto aislado si el resultado de aplicar la máscara sobre el píxel (en valor absoluto) es mayor o igual que un cierto valor umbral  $T$ , fijado por el “toma la decisión”. Dicho valor depende de la aplicación que estemos realizando.



# Puntos aislados

---

- La máscara detecta un punto aislado (en el pixel central) con un valor alto de la intensidad.
- Decimos que un punto aislado ha sido detectado en la región donde está centrada la máscara, si
$$|R| \geq T$$
- La respuesta de la operación con la máscara será 0 cuando esté centrada en zonas de intensidad uniforme

# En Matlab

---

□  $g = \text{abs}(\text{imfilter}(\text{double}(f), w)) \geq T$

donde  $f$  es la imagen de entrada,  $w$  el filtro.

- La imagen resultante  $g$  es lógica, pues sus valores son 0 ó 1, dependiendo de  $T$ , tiene un 1 si el punto es aislado y 0 en otro caso.

## Si el valor de T no está dado

---

- `w=[-1,-1,-1;-1,8,-1;-1,-1,-1];`
  - `g=abs(imfilter(double(f), w));`
  - `T=max(g(:));`
  - `g=g>=T;`
- 
- Al seleccionar a T como el mayor de g, no hay puntos mayores que T, a lo sumo el aislado que es igual a T.

# Puntos aislados

## Ejemplo

mask		original image		convolved image																																																											
<table><tr><td>-1</td><td>-1</td><td>-1</td></tr><tr><td>-1</td><td>8</td><td>-1</td></tr><tr><td>-1</td><td>-1</td><td>-1</td></tr></table>	-1	-1	-1	-1	8	-1	-1	-1	-1	*	<table><tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>10</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	1	1	1	1	1	1	10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	=	<table><tr><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr><tr><td>-</td><td>72</td><td>-9</td><td>0</td><td>-</td></tr><tr><td>-</td><td>-9</td><td>-9</td><td>0</td><td>-</td></tr><tr><td>-</td><td>0</td><td>0</td><td>0</td><td>-</td></tr><tr><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr></table>	-	-	-	-	-	-	72	-9	0	-	-	-9	-9	0	-	-	0	0	0	-	-	-	-	-	-
-1	-1	-1																																																													
-1	8	-1																																																													
-1	-1	-1																																																													
1	1	1	1	1																																																											
1	10	1	1	1																																																											
1	1	1	1	1																																																											
1	1	1	1	1																																																											
1	1	1	1	1																																																											
-	-	-	-	-																																																											
-	72	-9	0	-																																																											
-	-9	-9	0	-																																																											
-	0	0	0	-																																																											
-	-	-	-	-																																																											

Dependiendo del valor de T, obtenemos

4 puntos, si (  $0 < T \leq 9$  )

1 punto, si (  $9 < T \leq 72$  )

Ningún punto, si (  $T > 72$  )

# Detección de discontinuidades : líneas

---

- El próximo paso en nivel de complejidad, es detectar líneas.
- Si miramos la primer máscara cuando es movida alrededor de una imagen, responde con mayor intensidad en sentido horizontal.
- Con un fondo constante la máxima respuesta la tenemos cuando la línea pasa por la fila del medio.

# Detección de líneas

✓ Frecuentemente estamos interesados en detectar líneas en una determinada dirección. Los píxeles que forman parte de una línea horizontal, vertical o diagonal, tendrán respuestas extremas (mayor respuesta en algún sentido) ante alguna de las máscaras siguientes:

-1	-1	-1	2	-1	-1	-1	2	-1	-1	-1	2
2	2	2	-1	2	-1	-1	2	-1	-1	2	-1
-1	-1	-1	-1	-1	2	-1	2	-1	2	-1	-1

Horizontal

+45°

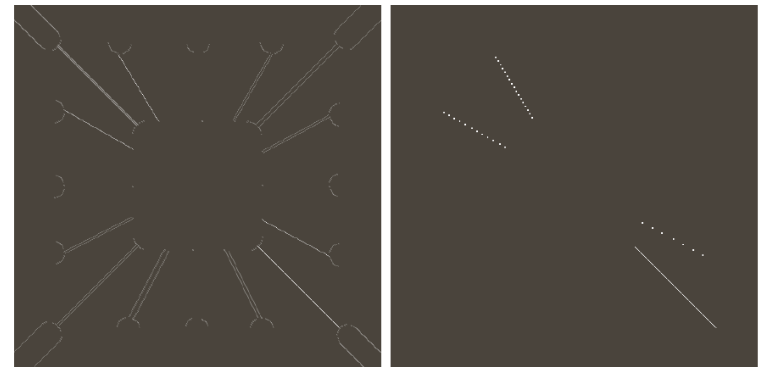
Vertical

-45°

# Detección de líneas

Ejemplo, supongamos que estamos interesados en las líneas de la imagen con una orientación de  $-45^\circ$ :

2	-1	-1
-1	2	-1
-1	-1	2



Valores absolutos de los resultados usando la máscara de detección de líneas con  $-45^\circ$  y posterior umbralización.

# Detección de discontinuidades: bordes

---

- Detección de borde es la aproximación más común en la detección de discontinuidades en la intensidad de la imagen.
- Recordemos la primera derivada (el gradiente)

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$$\nabla f = \text{mag}(\nabla f) = \left[ \left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2 \right]^{1/2}$$



# Detección de bordes

---

- El gradiente es cero en áreas de intensidad constante y los valores son proporcionales al grado de cambio de intensidad en áreas donde los valores de los pixeles son variables.
- Una propiedad del gradiente es que “apunta” en la máxima dirección de cambio de la imagen, el ángulo es:

$$\alpha(x, y) = \tan^{-1} \left( \frac{\partial f / \partial y}{\partial f / \partial x} \right)$$

# Detección de bordes

---

- La derivada segunda en el procesamiento de imágenes, generalmente es computada usando el Laplaciano 2D :

$$\nabla^2 f(x, y) = \frac{\partial^2 f(x, y)}{\partial x^2} + \frac{\partial^2 f(x, y)}{\partial y^2}$$

- Muy sensible al ruido.

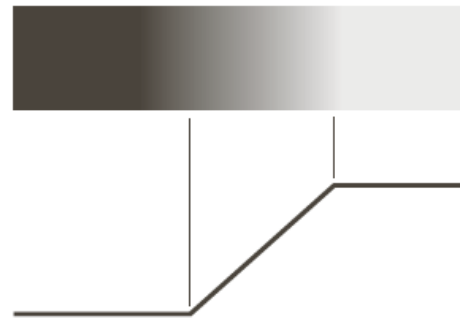
# Detección de bordes

---

## Borde de una imagen digital en tonos de grises



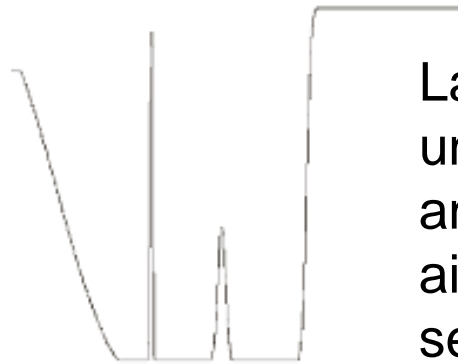
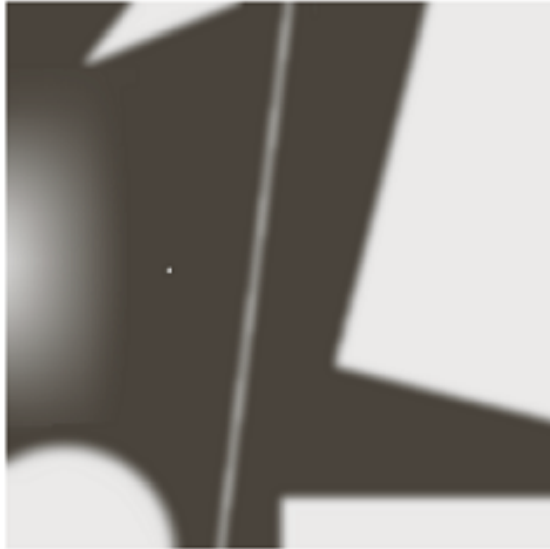
Borde ideal: forman un camino de un píxel de ancho, en los que se produce, perpendicularmente, un cambio en el nivel de gris.



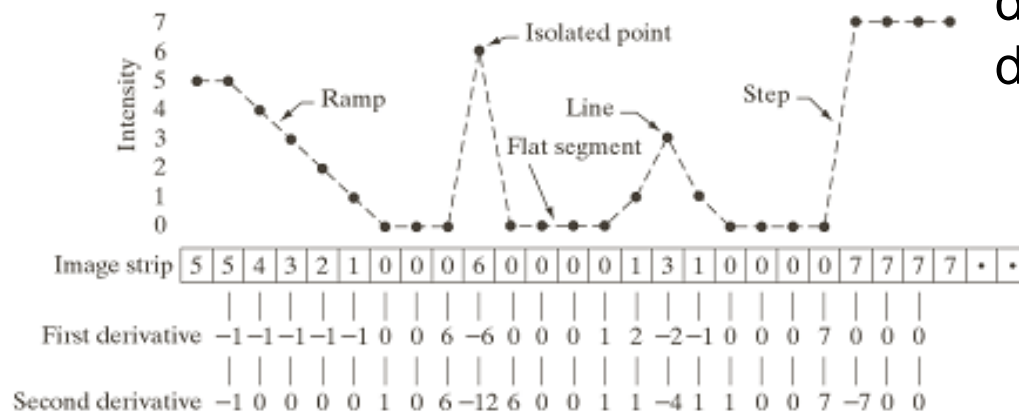
Borde "rampa": forman un conjunto de píxeles conexos en los que se produce, en una determinada dirección, una variación gradual en el nivel de gris.

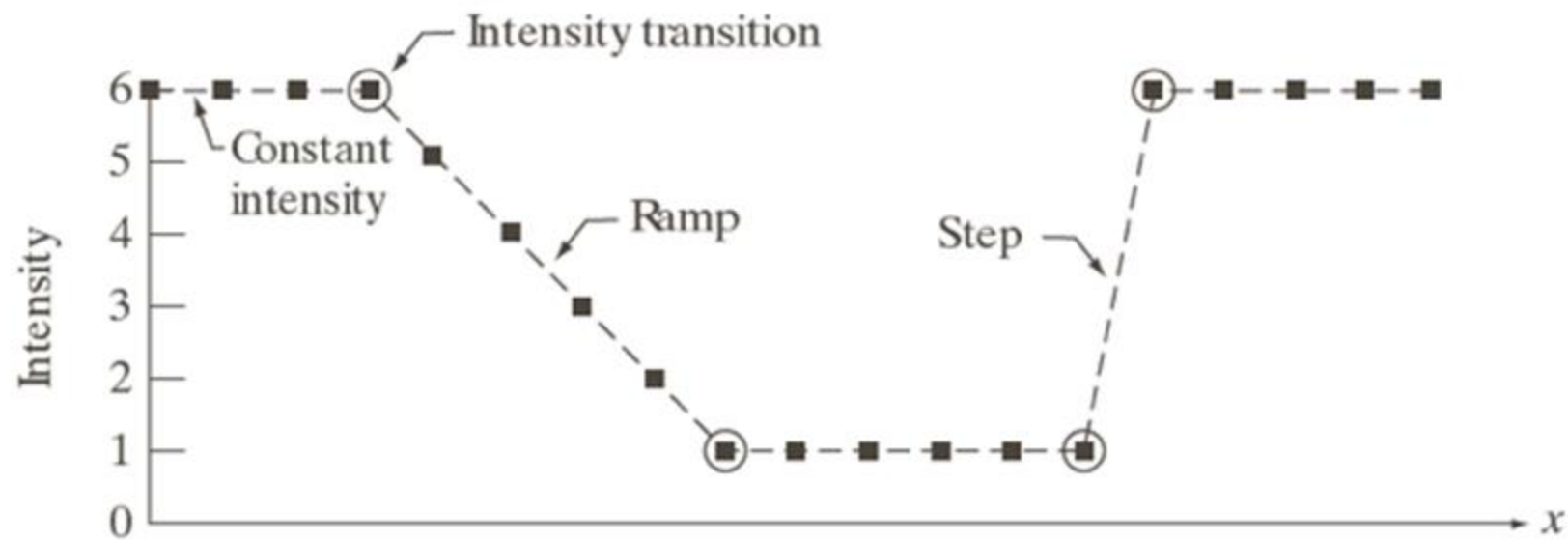
Un punto se dice que es del borde si su derivada primera dos-dimensional es mayor que un cierto valor umbral.

# 1er y 2da derivada



La segunda derivada tiene una respuesta más fuerte ante detalles como puntos aislados o líneas, por lo que se usa el Laplaciano para detectar estos tipos de discontinuidad.





Scan line	6	6	6	6	5	4	3	2	1	1	1	1	1	1	6	6	6	6	6	$\rightarrow x$
1st derivative	0	0	-1	-1	-1	-1	-1	0	0	0	0	0	0	5	0	0	0	0		
2nd derivative	0	0	-1	0	0	0	0	1	0	0	0	0	0	5	-5	0	0	0		

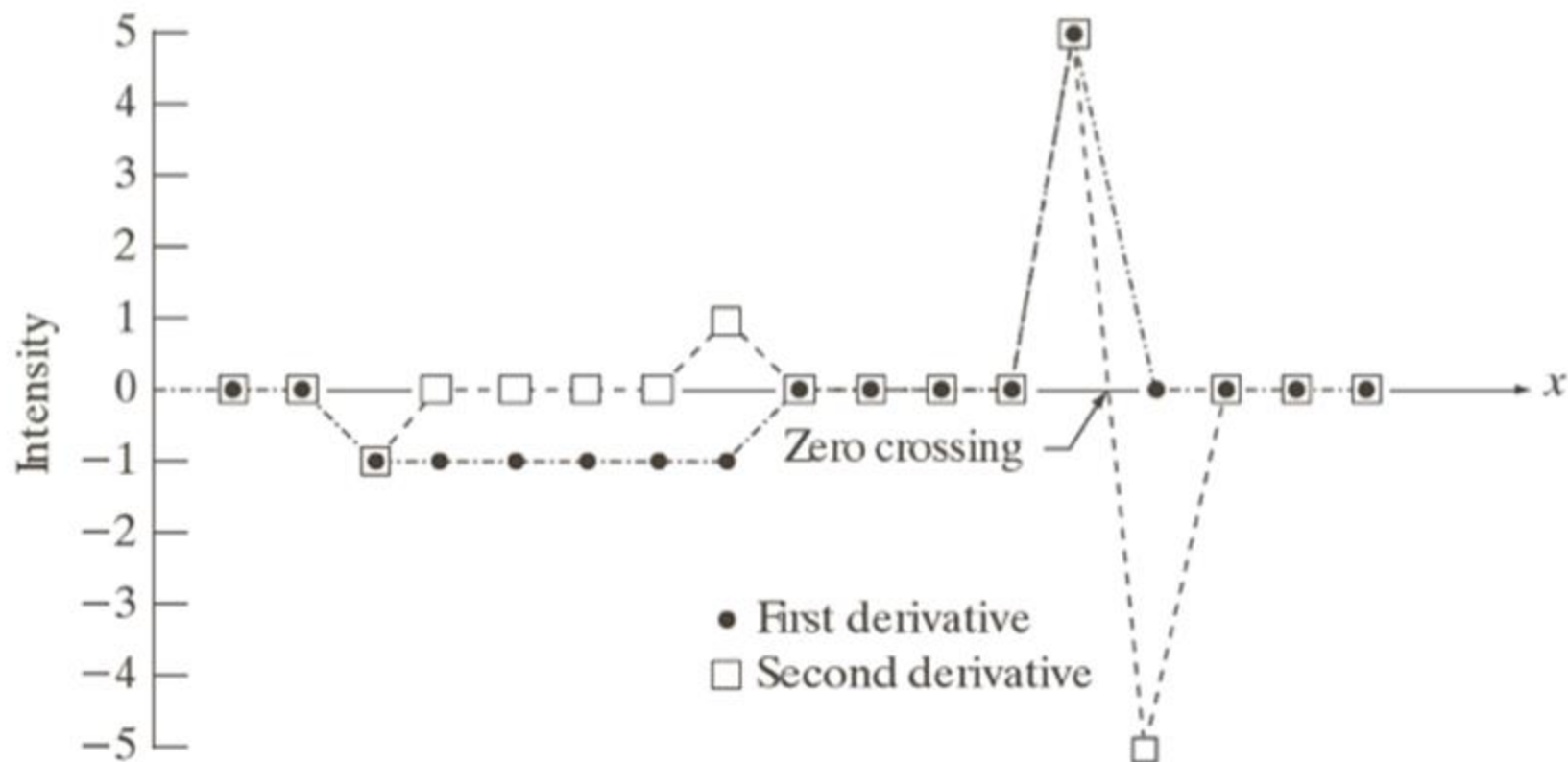
Scan line      

6	6	6	6	5	4	3	2	1	1	1	1	1	1	6	6	6	6	6
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

 $\rightarrow x$

1st derivative    0   0 -1 -1 -1 -1 -1   0   0   0   0   0   5   0   0   0   0

2nd derivative    0   0 -1   0   0   0   0   1   0   0   0   0   5 -5   0   0   0



# Detección de bordes

---

- La idea en la mayoría de las técnicas de detección de bordes, es calcular un operador local de derivación y decir que un pixel pertenece a un borde si se produce un cambio brusco de los niveles de gris de sus vecinos.
- Ya vimos filtros que detectan bordes, los mencionaremos nuevamente y recordemos que también “detectarán” al ruido.

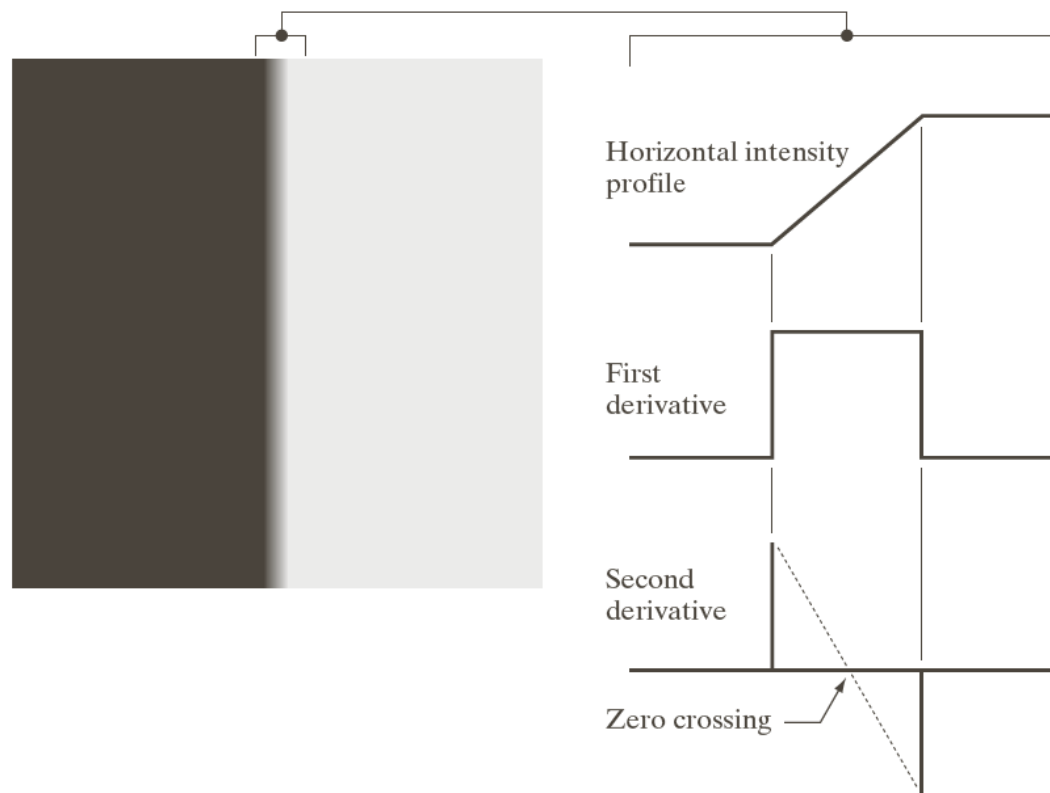
# Detección de bordes

---

- En general los pasos a seguir son:
- Encontrar los lugares donde la primera derivada es mayor que un umbral.
- Encontrar los lugares donde la segunda derivada tiene un cruce por cero.



# Detección de bordes 1ra y 2da derivada



- 
- ❑ En general, no hay forma de conocer si los píxeles detectados como parte del borde son correctos o no (intuitivamente hablando).
  - ❑ Es lo que se llama falso positivo (el detector devuelve un píxel cuando en realidad no pertenecía a ningún borde) y falso negativo (el detector no devuelve un píxel cuando en realidad pertenecía a un borde).

# En Matlab

---

- `[g,t] = edge(f, 'método', parámetro)`
- donde `f` es la imagen de entrada
- método: Sobel, Prewitt, Roberts, Laplaciano, Cruce por cero, Canny.
- `g` es un arreglo lógico con 1s donde se detectó un borde y 0 en otro lugar.
- parámetro `t` es opcional, usado por el gradiente para saber que punto pertenece al borde (umbral).

---


$$\nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right] = [G_x, G_y]$$

$$|\nabla f| \approx |G_x| + |G_y|$$

$$\alpha(x, y) = \tan^{-1} \left( \frac{G_y}{G_x} \right)$$

-1	0	0	-1
0	1	1	0

Roberts

-1	-1	-1	-1	0	1
0	0	0	-1	0	1
1	1	1	-1	0	1

Prewitt

-1	-2	-1	-1	0	1
0	0	0	-2	0	2
1	2	1	-1	0	1

Sobel



a	b
c	d

**FIGURE 10.16**

(a) Original image of size

$834 \times 1114$  pixels, with intensity values scaled to the range  $[0, 1]$ .

(b)  $|g_x|$ , the component of the gradient in the  $x$ -direction, obtained using the Sobel mask in Fig. 10.14(f) to filter the image.

(c)  $|g_y|$ , obtained using the mask in Fig. 10.14(g).

(d) The gradient image,  $|g_x| + |g_y|$ .



---

a	b
c	d

**FIGURE 10.18**  
Same sequence as in Fig. 10.16, but with the original image smoothed using a  $5 \times 5$  averaging filter prior to edge detection.

---

- 
- Detección de bordes en imágenes en escala de grises: Marr-Hildreth
  - Los pasos a seguir son:
    - Aplicar un filtro Gaussiano.
    - Calcular el Laplaciano de la imagen resultante.
    - Determinar los píxeles de “paso por cero”.

## Detección de bordes en imágenes en escala de grises: Marr-Hildreth

### PASO 1

Realizar una convolución a la imagen I con una matriz G que modeliza una función gaussiana bidimensional:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

donde  $\sigma$  representa la desviación típica.

Ejemplo de máscara 5x5  
para el filtro gaussiano con  
 $\sigma=1.0$ :

	1	4	7	4	1
	4	16	26	16	4
$\frac{1}{273}$	7	26	41	26	7
	4	16	26	16	4
	1	4	7	4	1

Recuérdese que la convolución de una imagen con una función de esta forma emborrona la imagen con un grado de emborronamiento proporcional a  $\sigma$  (por tanto, se produce una reducción de ruido).



## Detección de bordes en imágenes en escala de grises: Marr-Hildreth

### PASO 2

Calcular el Laplaciano de la imagen resultante.

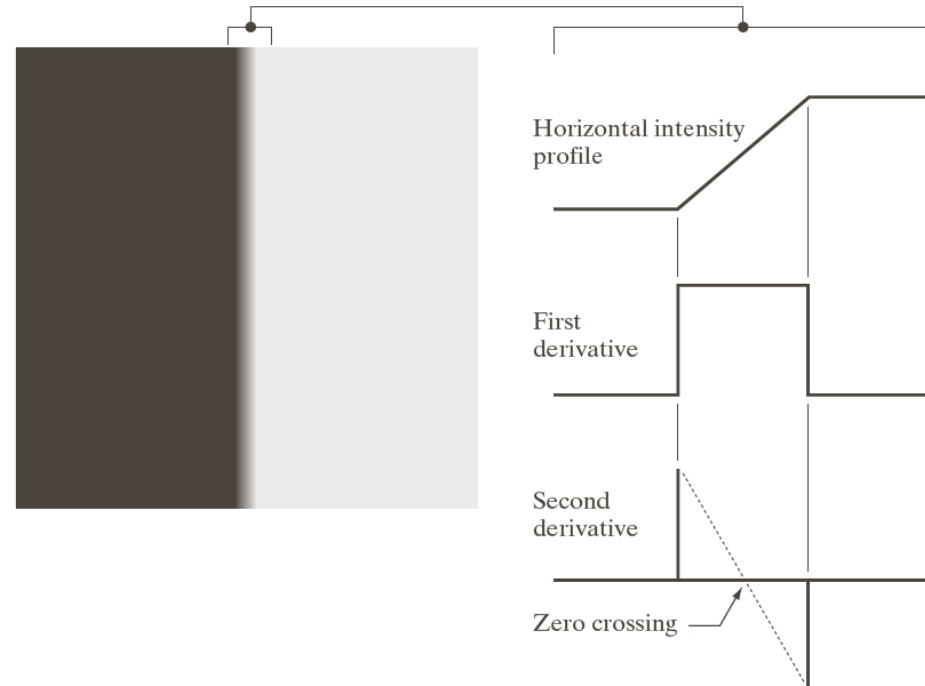
$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 1 & 2 & -16 & 2 & 1 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

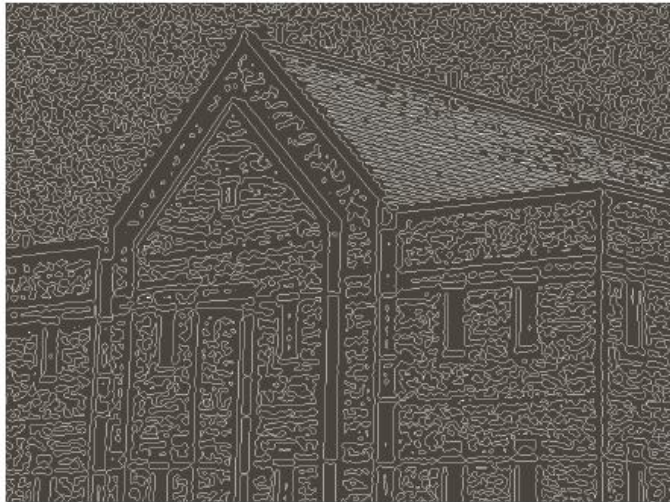
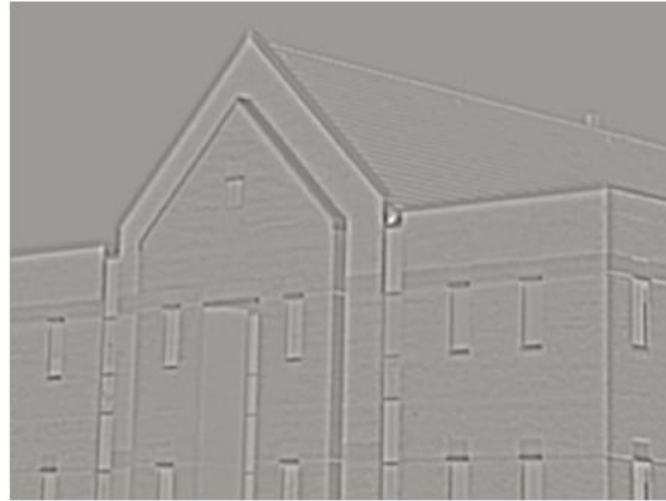
## Detección de bordes en imágenes en escala de grises: Marr-Hildreth

### PASO 3

Determinar los píxeles de “paso por cero”.



Los píxeles del borde son aquellos tal que el Laplaciano de dos de sus vecinos en posiciones opuestas tienen distinto signo (píxeles de paso por cero). Normalmente se considera un valor umbral para el valor absoluto de la diferencia numérica entre posiciones opuestas para considerar que un píxel es de paso por cero.



a	b
c	d

**FIGURE 10.22**

(a) Original image of size  $834 \times 1114$  pixels, with intensity values scaled to the range  $[0, 1]$ . (b) Results of Steps 1 and 2 of the Marr-Hildreth algorithm using  $\sigma = 4$  and  $n = 25$ . (c) Zero crossings of (b) using a threshold of 0 (note the closed-loop edges). (d) Zero crossings found using a threshold equal to 4% of the maximum value of the image in (b). Note the thin edges.

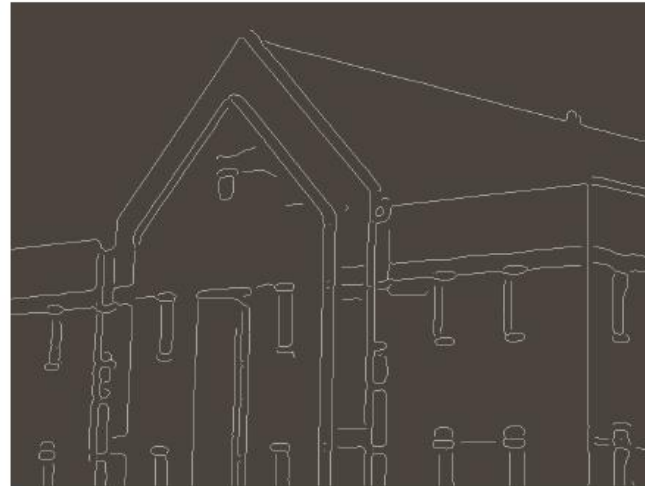
## Detección de bordes en imágenes en escala de grises: Canny

---

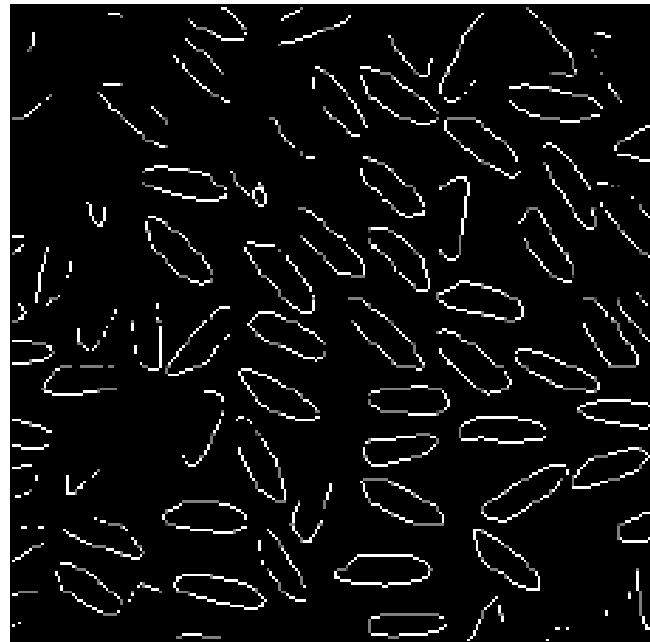
Es el detector de bordes más potente que existe actualmente.

Los pasos principales del algoritmo son:

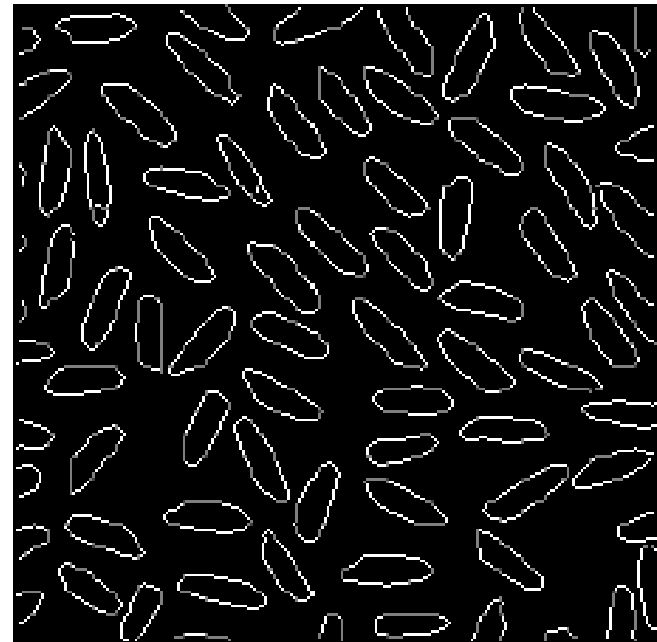
1. Se realiza una convolución con un filtro gaussiano. De esta forma la imagen se suaviza (eliminación de ruidos).
2. Se calcula el gradiente de la imagen suavizada, para determinar los píxeles donde se produce máxima variación (mayor módulo del vector gradiente). También se determina la dirección del vector gradiente.
3. La matriz  $M$  correspondiente al módulo del gradiente de la función gaussiana tendrá valores grandes donde la variación de la intensidad sea grande. Se eliminan (igualan a cero) aquellos píxeles que no son máximos locales en la dirección del gradiente (que es perpendicular al borde).
4. Se realiza un proceso de doble umbralización para determinar los píxeles del borde: se marcan los píxeles con valor por encima de un umbral  $T_1$ ; se marcan aquellos píxeles conectados a los primeros cuyo valor esté por encima de un segundo umbral  $T_2$  ( $T_2 < T_1$ ). Esto eliminará falsos bordes o bordes dobles.



El filtro Gaussiano se ha realizado para  $\sigma=4$  y una máscara de tamaño  $25 \times 25$ . Los umbrales considerados han sido  $T1=0.1$  y  $T2=0.04$



Sobel Filter



Canny Filter

---

## **Detección de bordes en imágenes en escala de grises**

En resumen:

- (1) La detección de bordes usando operadores de aproximación del gradiente tiende a funcionar bien en los casos en que se involucran imágenes con transiciones de intensidad claramente definidas y ruidos relativamente bajos.
- (2) Los pasos por cero ofrecen una alternativa en los casos en que los bordes están emborronados o cuando está presente un alto contenido de ruido. El paso por cero ofrece fiabilidad en las localizaciones de bordes y la propiedad de suavizado de la convolución gaussiana reduce los efectos del ruido. El precio a pagar por estas ventajas es el incremento de complejidad de cálculo y tiempo.
- (3) El algoritmo de Canny es el que ofrece mejores resultados para bordes de imágenes con ruido gaussiano.

# Detección de rectas



Transformada de Hough



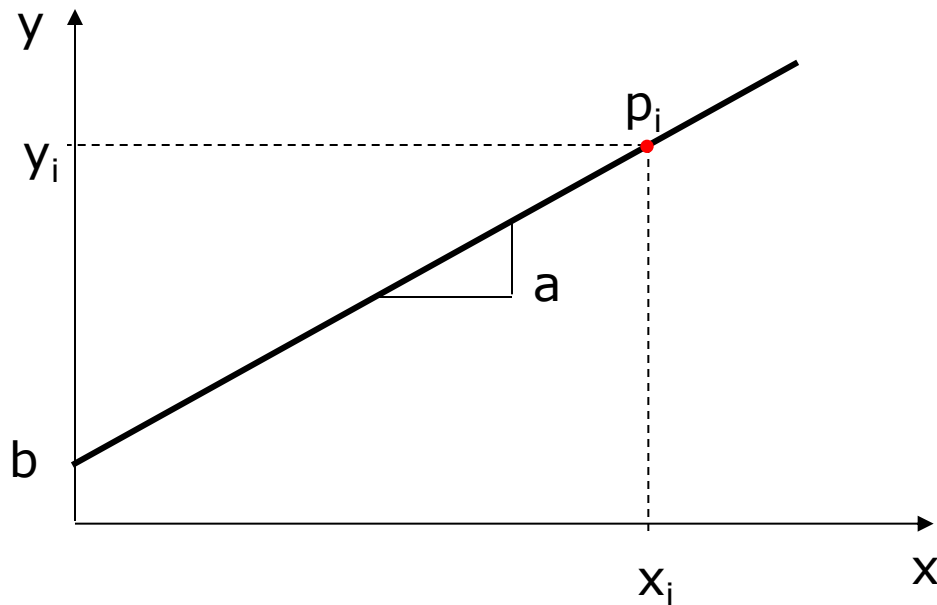
# Extracción de líneas rectas

- Considerar puntos (bordes) en el plano imagen.
- Todas las líneas en ese plano se pueden escribir como:

$$y = ax + b$$

- Si se considera un punto  $(x_i, y_i)$ , sobre una determinada recta  $(a, b)$ :

$$y_i = a x_i + b$$



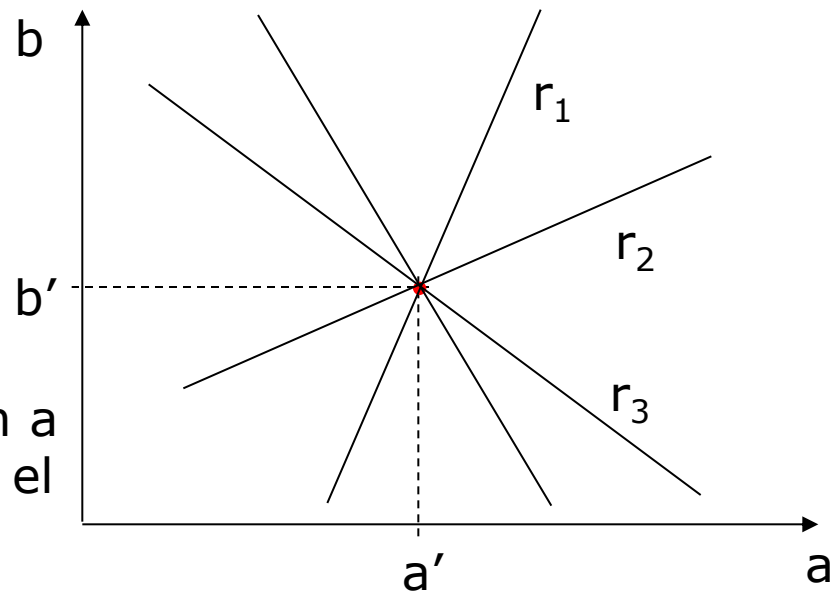
- ✓ Si la ecuación de una recta cualquiera se escribe (espacio de parámetros):

$$b = -x_i a + y_i$$

- ✓ Por cada punto  $(x_i, y_i)$  situado sobre una recta en el plano imagen (a-b), se obtiene una recta ( $r_i$ ) en el espacio de parámetros:

$$y = ax + b \rightarrow \begin{cases} (x_1, y_1) \rightarrow r_1 : b = -x_1 a + y_1 \\ (x_2, y_2) \rightarrow r_2 : b = -x_2 a + y_2 \\ \dots \quad \dots \quad \dots \\ (x_n, y_n) \rightarrow r_n : b = -x_n a + y_n \end{cases}$$

Si los N puntos  $(x_i, y_i)$  elegidos pertenecen a una misma recta  $y = a'x + b'$ , las rectas en el espacio de parámetros se cortarán en un punto de coordenadas  $(a', b')$ .



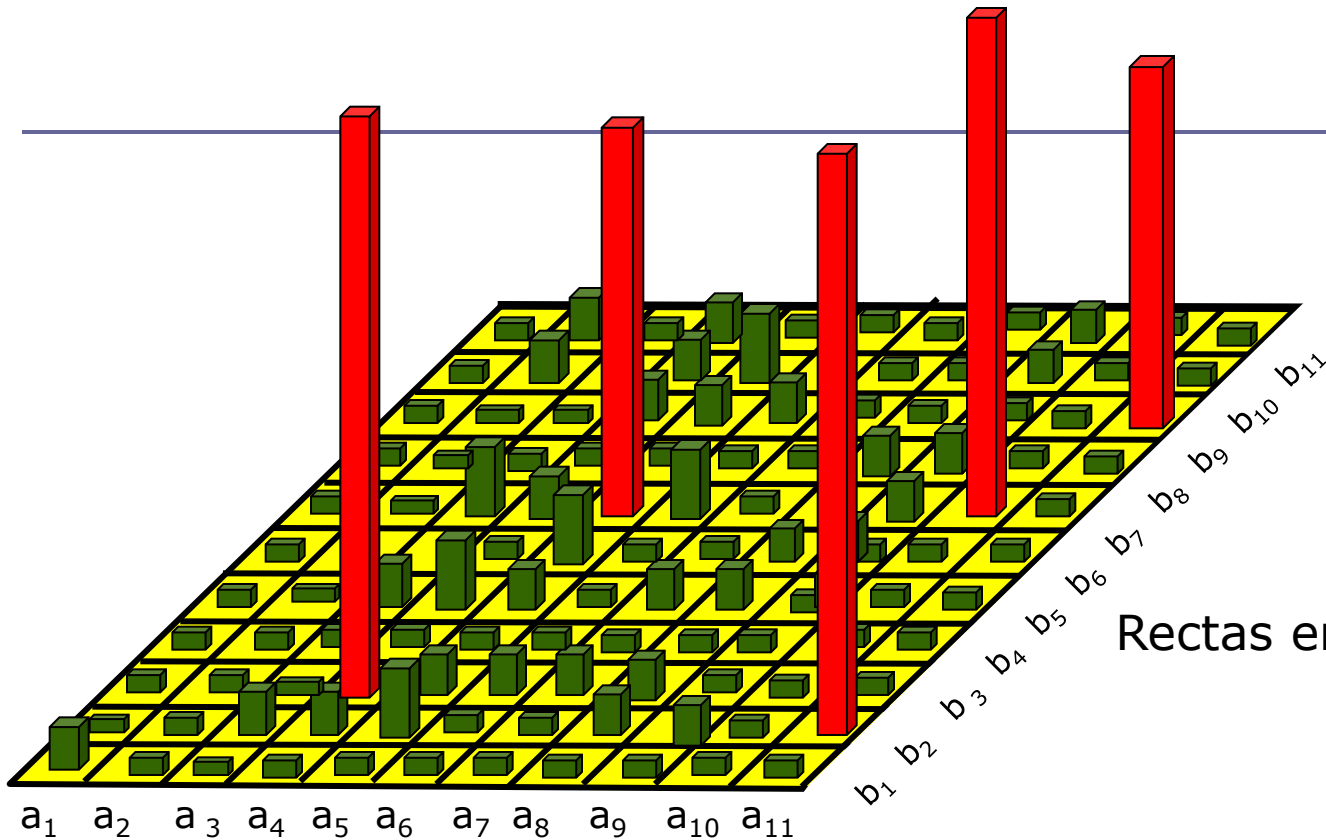
Cuantos más puntos se tomen sobre una línea en el plano imagen, más rectas se cruzan en el punto  $(a', b')$  del espacio de parámetros

- ✓ Aplicación al procesamiento de imágenes:
  - 1. Subdividir (cuantificar) "a" y "b" en intervalos apropiados, dentro de los rangos de variación ( $a_{\text{mín}}, a_{\text{máx}}$ ) y ( $b_{\text{máx}}, b_{\text{mín}}$ ). Es importante decidir que son "intervalos apropiados".
- 
- 2. Crear un array de acumulación  $H(a,b)$ . Inicialmente todos las celdas del array se ponen a cero.
  - 3. Para cada punto  $(x_i, y_i)$  de borde del plano imagen (cuya magnitud supere un umbral), se obtienen los valores discretos de  $(a, b)$  a partir de la ecuación:

$$b = -x_i a + y_i \rightarrow \begin{cases} a = a_1 \rightarrow b_1 = -x_i a_1 + y_i \\ a = a_2 \rightarrow b_2 = -x_i a_2 + y_i \\ a = a_n \rightarrow b_n = -x_i a_n + y_i \end{cases}$$

- 4. Incrementar todas las celdas de  $H(a,b)$  que resulten de la ecuación anterior.
- Obsérvese que  $H$  es un histograma bidimensional.
- 5. Máximos locales de  $H(a_i, b_i)$  se corresponden con puntos sobre rectas en el plano imagen. Los parámetros de las rectas en el plano imagen serán:  $a_i, b_i$ .

## Array de acumulación



Rectas en el plano imagen:5

$$y = a_4 x + b_3$$

$$y = a_{11} x + b_2$$

$$y = a_5 x + b_7$$

$$y = a_{10} x + b_7$$

$$y = a_{11} x + b_9$$

Nº de operaciones: Para "n" puntos de borde, si "a" se cuantifica con M valores, el nº de operaciones suma y producto son:  $n \times M$

La celda de coordenadas  $(i,j)$ , con valor acumulador  $A(i,j)$ , corresponde al cuadrado asociado a las coordenadas del espacio paramétrico  $(a_i,b_j)$  . Inicialmente, estas celdas están puestas a cero. Para cada punto  $(x_k,y_k)$  en el plano imagen, igualamos el parámetro "a" a cada uno de los valores de la subdivisión permitidos en el eje a, y resolvemos el valor "b" correspondiente, según la ecuación  $b = -x_k \cdot a + y_k$  . Los valores de " b " resultantes son aproximados en el eje b al valor permitido más cercano.

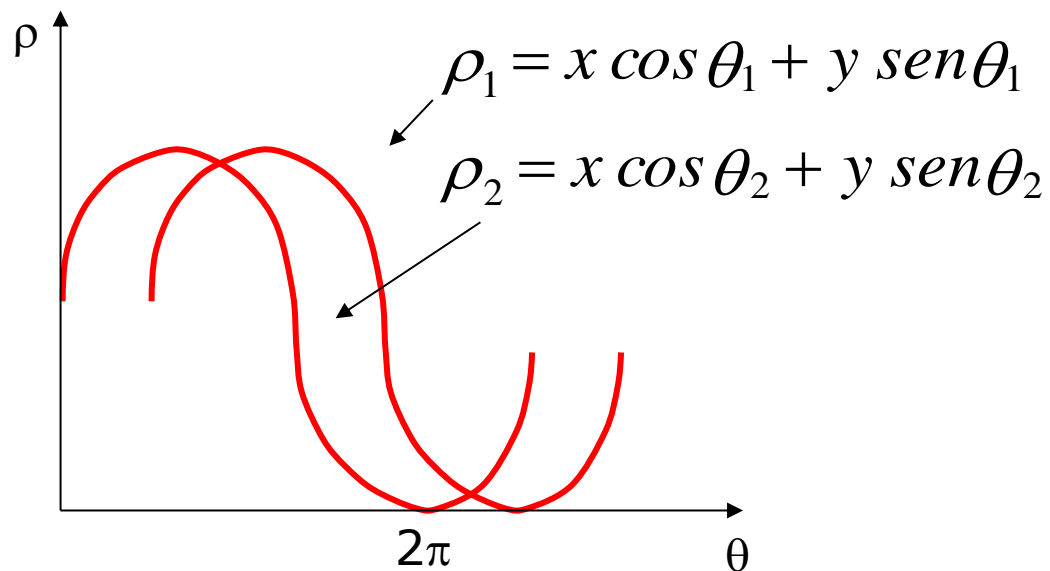
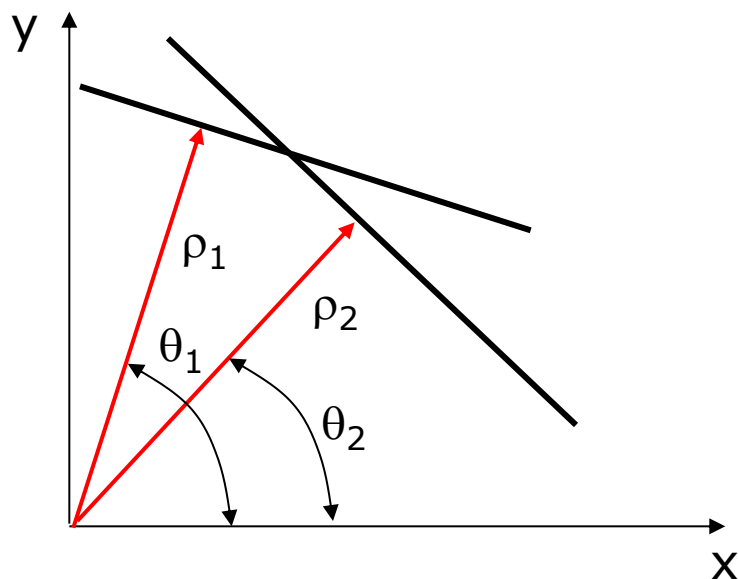
Si una elección de  $a_p$  da lugar al valor  $b_q$ ,  
hacemos:  $A(p,q) := A(p,q) + 1$  (se  
incrementa de uno en uno el valor  
acumulador correspondiente). Así, al final  
del proceso un valor de  $M$  en  $A(i,j)$   
corresponde a  $M$  puntos en el plano  $xy$   
sobre la recta  $y = a_i x + b_j$ . La precisión en la  
linealidad de estos puntos se establece por  
el número de subdivisiones en el plano  $ab$ .  
Si subdividimos el eje  $a$  en  $k$  partes,  
entonces para cada punto  $(x_k, y_k)$   
obtenemos  $k$  valores de " $b$ " que  
corresponden a los  $k$  posibles valores de  
" $a$ "

# Transformada de Hough: Problemas

- Un problema de la representación es que la pendiente (a) como la ordenada en y cambia conforme la recta se acerca a posición vertical.
- Para evitar este problema se usa la siguiente ecuación de la recta:

$$\rho = x \cos \theta + y \sin \theta$$

$$0 \leq \rho \leq (x_{\text{máx}}^2 + y_{\text{máx}}^2)^{1/2}, \quad 0 \leq \theta \leq \pi$$



## Transformada de Hough: representación alternativa

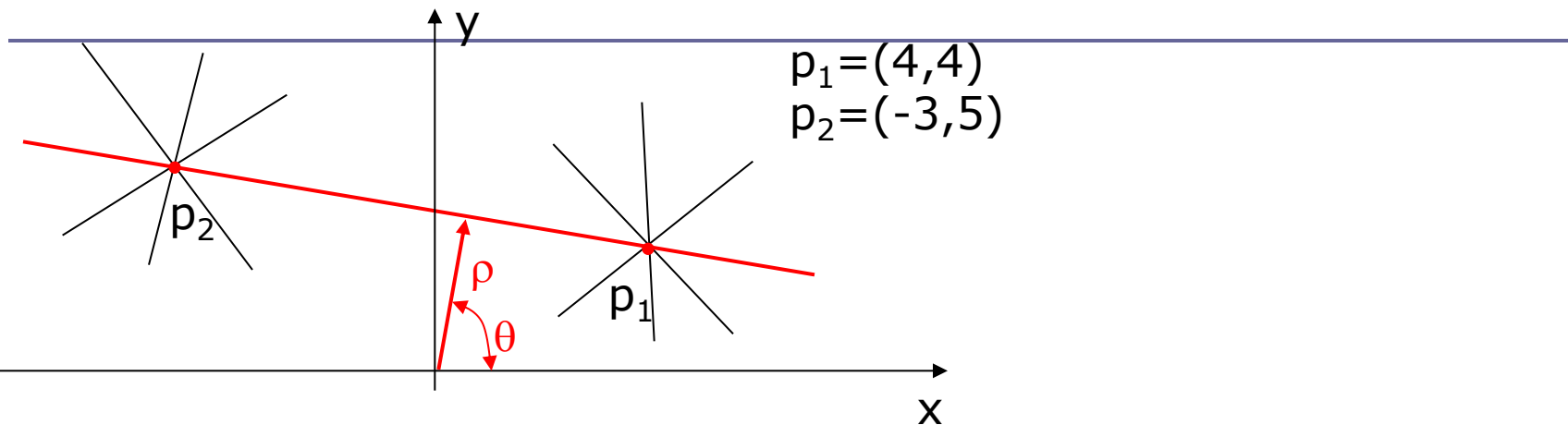
---

- La forma de construir el acumulador en el plano  $\rho$ - $\theta$  es similar al primer algoritmo.
- La única diferencia está en que, en vez de líneas rectas, se obtendrán curvas sinusoidales.
- Así,  $N$  puntos colineales  $(x,y)$  pertenecientes a una recta en el plano imagen:

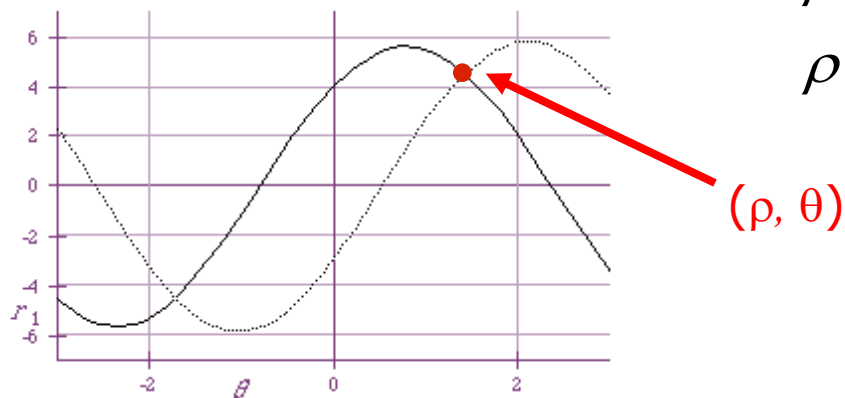
$$\rho_k = x \cos \theta_k + y \sin \theta_k \quad \text{para los } N \text{ puntos } (x, y)$$

- Darán lugar a  $N$  curvas sinusoidales que se cortarán, en el espacio de parámetros, en el punto  $(\rho_k, \theta_k)$ .





Espacio  $(\rho, \theta)$



$$\left. \begin{array}{l} \rho = -3\cos\theta + 5\text{sen}\theta \\ \rho = 4\cos\theta + 4\text{sen}\theta \end{array} \right\} \rightarrow \left\{ \begin{array}{l} \theta = 1.4289 \\ \rho = 4.5255 \end{array} \right.$$

# Resumen

---

- ❑ Discretizar el espacio de parámetros, estableciendo valores máximos y mínimos de  $\rho$  y  $\theta$ , así como el número de valores de  $\rho$  y  $\theta$ .
- ❑ Generar el acumulador  $H(\rho, \theta)$ ; poner todos sus valores a cero.
- ❑ Para todos los puntos de borde  $(x_i, y_i)$ 
  - Calcular la dirección del vector gradiente  $\theta$ .
  - Obtener  $\rho$  de la ecuación  $\rho = u_i \cos \theta + v_i \sin \theta$ .
  - Incrementar  $H(\rho, \theta)$ .
- ❑ Para todas las celdas en el acumulador
  - Buscar los valores máximos del acumulador.
  - Las coordenadas  $(\rho, \theta)$  dan la ecuación de la recta de la imagen

# Detección de similitudes



Umbralización

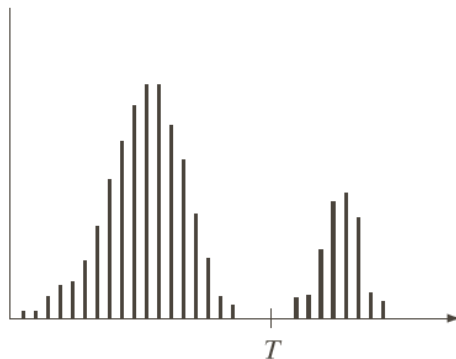
Crecimiento de regiones

# Umbralización

---

Un método básico para diferenciar un objeto del fondo de la imagen es mediante una simple binarización.

A través del histograma obtenemos una gráfica donde se muestran el número de píxeles por cada nivel de gris que aparece en la imagen. Para binarizar la imagen, se deberá elegir un valor adecuado (umbral) dentro de los niveles de grises, de tal forma que el histograma forme un valle en ese nivel. Todos los niveles de grises menores al umbral calculado se convertirán en negro y todos los mayores en blanco.



$$g(x, y) = \begin{cases} 1 & \text{si } f(x, y) \geq T \\ 0 & \text{si } f(x, y) < T \end{cases}$$

Existen muchos métodos de búsqueda del umbral. A continuación exponemos algunos de ellos.

# Umbralización

---

- ❑ Pixeles etiquetados con 1 corresponden a objetos y pixeles etiquetados con 0 corresponden al fondo.
- ❑ Cuando  $T$  es cte, se llama umbralización global.
- ❑ Cuando permitimos que  $T$  varíe la llamamos umbralización local.

# Umbralización global

---

- El umbral sólo depende de los valores de nivel de gris.
- Un umbral automático puede ser realizado por el siguiente proceso iterativo :
  1. Seleccionar un estimado inicial para  $T_0$  (punto medio entre máx y mín intensidades de la imagen).
  2. Segmentar la imagen usando  $T$ . Esto genera dos grupos de pixeles ( $G_1$  y  $G_2$ ).
  3. Calcular los promedios  $u_1$  y  $u_2$  de  $G_1$  y  $G_2$ .
  4. Calcular un nuevo valor umbral
$$T = 1/2(u_1 + u_2)$$
  5. Repetir los pasos anteriores 2 a 4 hasta que la diferencia entre  $T$  y  $T_0$  sea menor que un valor prefijado.

# En Matlab

---

- La función que calcula el umbral es:  
graythresh
- Usa el método de Otsu para calcular el T.
- Recordemos

$$p_q(r_q) = \frac{n_q}{n} \quad q = 0, 1, 2, \dots, L-1$$

- Donde  $n$  es el número total de pixels de la imagen,  $n_q$  el número de pixels con intensidad  $r_q$ ,  $L$  el número total de intensidades distintas en la imagen.

- 
- Supongamos un umbral  $k$ , tal que  $C_0$  es el conjunto de pixels con intensidades  $[0,1,2,\dots,k-1]$  y  $C_1$  el conjunto de pixeles con intensidades  $[k,k+1,\dots,L-1]$ .
  - El método de Otsu usa el umbral que maximiza la varianza “entre clases”  $\sigma_B^2$  que es definida como

$$\sigma_B^2 = \omega_0 (\mu_0 - \mu_T)^2 + \omega_1 (\mu_1 - \mu_T)^2$$



$$\omega_0 = \sum_{q=0}^{k-1} p_q(r_q)$$

---

$$\omega_1 = \sum_{q=k}^{L-1} p_q(r_q)$$

$$\mu_0 = \sum_{q=0}^{k-1} qp_q(r_q) / \omega_0$$

$$\mu_1 = \sum_{q=k}^{L-1} qp_q(r_q) / \omega_1$$

$$\mu_T = \sum_{q=0}^{L-1} qp_q(r_q)$$

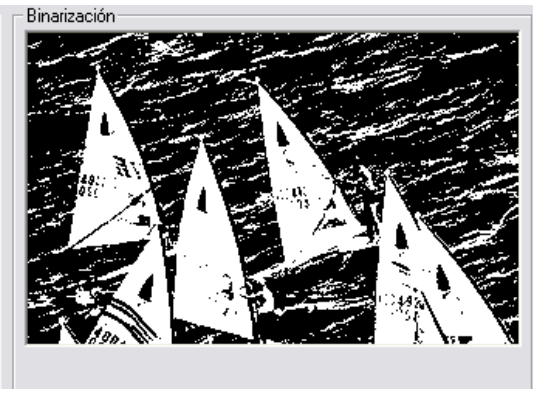
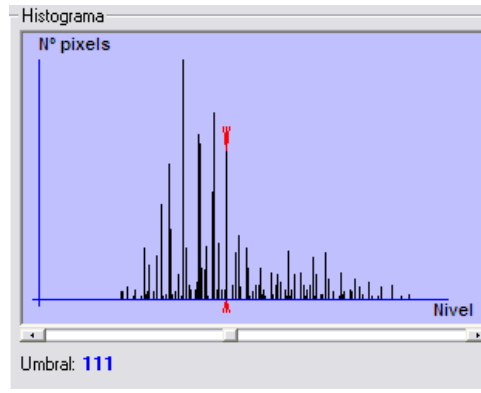
# Finalmente

---

- $T = \text{graythresh}(f)$
- Donde  $f$  es la imagen de entrada y  $T$  es el umbral devuelto por la función.
- Para segmentar este valor tiene que ser usado con la función `im2bw`.
- El umbral está normalizado en el rango  $[0,1]$ ; si la imagen es `uint8` hay que multiplicar al umbral por 255 antes de usarlo.

# Otros métodos: valor medio

Se usa el nivel medio de gris de la imagen como valor umbral. Esta umbralización tendrá éxito si el objeto y el fondo ocupan áreas comparables en tamaño en la imagen.



# Método del porcentaje de píxeles negros

---

- ❑ Dado un histograma, y un porcentaje de píxeles negros deseados, se determina el número de píxeles negros multiplicando el porcentaje por el número total de píxeles.
- ❑ A continuación se cuentan el número de píxeles de cada nivel del histograma, empezando por el nivel cero, hasta llegar al número de píxeles negros deseados.
- ❑ El umbral será el nivel de gris del histograma, en el que la cuenta llegue al número de píxeles negros deseados.

# Método de los dos picos

---

- Si el histograma muestra al menos dos picos, el valor umbral más apropiado suele ser (según se ve en la práctica) el menor valor entre esos dos picos del histograma.
- Seleccionar el umbral automáticamente consiste en:
  - Encontrar los dos picos más altos.
  - Encontrar el menor valor entre ellos.

# Umbralización local

---

- T depende de  $I(x,y)$  y de alguna propiedad local del pixel, ej. el nivel de gris medio de una vecindad del pixel.

$$g(x, y) = \begin{cases} 1 & \text{si } f(x, y) \geq T(x, y) \\ 0 & \text{si } f(x, y) < T(x, y) \end{cases}$$

- Más adelante...

# Segmentación basada en regiones

---

- El objetivo de la segmentación es dividir una imagen en regiones.
- Hasta ahora encontramos límites entre regiones por discontinuidades en niveles de intensidad (tonos de grises) ó usando la distribución de las intensidades de pixeles (umbral en histograma).
- Ahora intentaremos encontrar regiones directamente.

# Principios básicos

---

- Si  $R$  representa a la imagen entera, podemos ver a la segmentación como un proceso que particiona  $R$  en distintas subregiones  $R_1, R_2, R_3, \dots, R_n$ . Tal que:

$$(a) \quad \bigcup_{i=1}^n R_i = R$$

(b)  $R_i$  es una región conectada

(c)  $R_i \cap R_j = \Phi$  para todo  $i \neq j$

(d)  $P(R_i) = TRUE$  para  $i = 1, 2, \dots, n$

(e)  $P(R_i \cup R_j) = FALSE$  para cualesquiera regiones  
adyacentes



# Condiciones

---

- (a) La segmentación tiene que ser completa. Cada pixel tiene que estar en una región.
- (b) Los puntos tienen que estar conectados.
- (c) Las regiones tienen que ser disjuntas.
- (d) Propiedad que tienen que satisfacer los pixels de una región.
- (e) Regiones adyacentes son diferentes en cuanto a (d).

# Crecimiento de regiones

---

- Es un procedimiento que agrupa los píxeles o subregiones de la imagen en regiones mayores basándose en un criterio prefijado. Normalmente se empieza con unos puntos “semillas” para formar una determinada región, añadiendo aquellos píxeles vecinos que cumplan la propiedad especificada (por ejemplo, que estén en un rango de nivel de gris determinado).
- La propiedad considerada en el crecimiento de regiones debe tener en cuenta la información sobre conectividad o adyacencia de la imagen.

# En Matlab

---

- `[g, NR, SI, TI]=regiongrow(f, S, T)`
- Donde `f` es la imagen a segmentar
- `S` puede ser un arreglo (del mismo tamaño que `f`) ó un escalar. Si es un arreglo tiene 1s en los puntos que son semillas y 0s en otro lugar. Si es un escalar, define un valor de intensidad que todos los puntos de `f` con ese valor son semillas.

# Seguimos....

---

- T es un umbral, que puede ser un arreglo ó un escalar. Este umbral se utiliza para determinar la similitud de un pixel respecto de la semilla a la cual está 8-conectado.
- g es la imagen segmentada, cada pixel perteneciente a una región está etiquetado con un número entero.
- NR es el número de regiones.
- SI puntos semilla (mismo tamaño que f)
- TI puntos que satisfacen el test del umbral.

Ejemplo:

7	7	1	1	1	1	1	1	2	2	2	2
2	2	1	2	2	1	1	1	2	2	2	2
1	3	3	3	4	2	2	2	2	2	2	2
1	2	3	4	5	2	2	2	2	2	3	2
2	2	2	2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	2	2	2	2	2
1	3	4	4	4	4	5	6	7	3	4	1
1	2	3	3	3	3	3	3	3	3	3	3
4	4	2	3	4	5	6	8	8	8	8	8
1	2	3	4	4	4	8	8	8	8	8	8
1	2	3	4	5	6	8	8	8	8	8	8

$S = 8$        $T = 2$

Imagen Original

Ejemplo:

7	7	1	1	1	1	1	1	2	2	2	2
2	2	1	2	2	1	1	1	2	2	2	2
1	3	3	3	4	2	2	2	2	2	2	2
1	2	3	4	5	2	2	2	2	2	3	2
2	2	2	2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	2	2	2	2	2
1	3	4	4	4	4	5	6	7	3	4	1
1	2	3	3	3	3	3	3	3	3	3	3
4	4	2	3	4	5	6	8	8	8	8	8
1	2	3	4	4	4	8	8	8	8	8	8
1	2	3	4	5	6	8	8	8	8	8	8

$$S = 8 \quad T = 2$$

1º Paso: Seleccionamos los puntos que son semilla.

Ejemplo:

7	7	1	1	1	1	1	1	2	2	2	2
2	2	1	2	2	1	1	1	2	2	2	2
1	3	3	3	4	2	2	2	2	2	2	2
1	2	3	4	5	2	2	2	2	2	3	2
2	2	2	2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	2	2	2	2	2
1	3	4	4	4	4	5	6	7	3	4	1
1	2	3	3	3	3	3	3	3	3	3	3
4	4	2	3	4	5	6	8	8	8	8	8
1	2	3	4	4	4	8	8	8	8	8	8
1	2	3	4	5	6	8	8	8	8	8	8

S = 8      T = 2

2º Paso: Seleccionamos los puntos que cumplen la condición del límite T.

Ejemplo:

7	7	1	1	1	1	1	1	2	2	2	2
2	2	1	2	2	1	1	1	2	2	2	2
1	3	3	3	4	2	2	2	2	2	2	2
1	2	3	4	5	2	2	2	2	2	3	2
2	2	2	2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	2	2	2	2	2
1	3	4	4	4	4	5	6	7	3	4	1
1	2	3	3	3	3	3	3	3	3	3	3
4	4	2	3	4	5	6	8	8	8	8	8
1	2	3	4	4	4	8	8	8	8	8	8
1	2	3	4	5	6	8	8	8	8	8	8

**S = 8      T = 2**

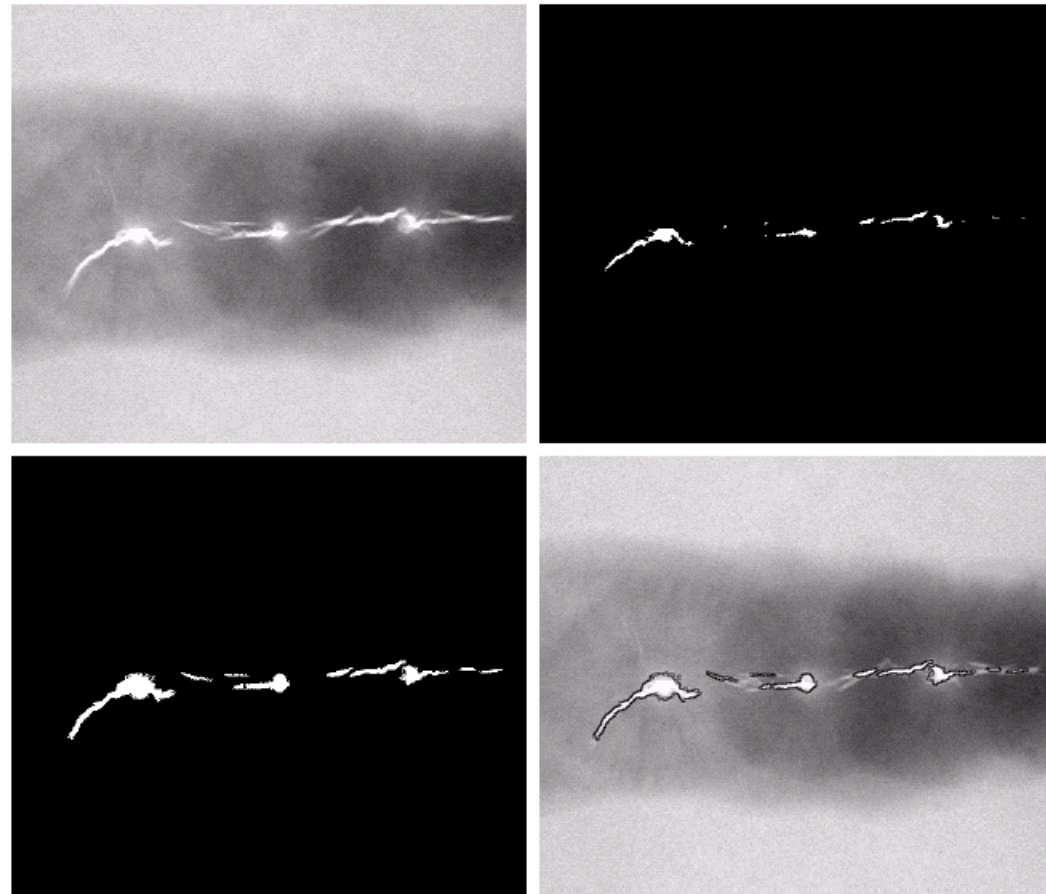
3º Paso: Nos quedamos solamente con los puntos que tengan conexión 8 con respecto a los puntos semilla.



a	b
c	d

**FIGURE 10.40**

(a) Image showing defective welds. (b) Seed points. (c) Result of region growing. (d) Boundaries of segmented defective welds (in black). (Original image courtesy of X-TEK Systems, Ltd.).



## Ejemplo

Puntos semilla:  
aquéllos con valor  
255.

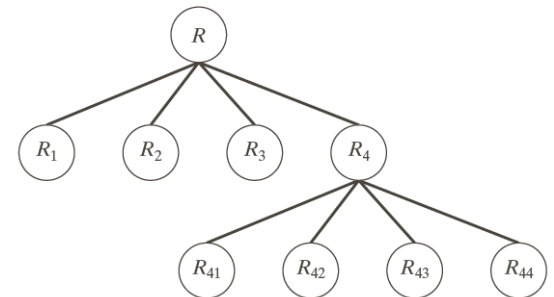
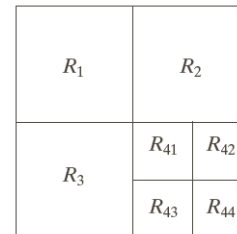
Criterios para  
aumentar una región:  
(1) diferencia en valor  
absoluto con píxel  
semilla menor que  
65; (2) 8-adyacencia  
con algún píxel de la  
región.

# División y fusión

Se subdivide la imagen inicialmente en un conjunto de regiones disjuntas, dentro de las cuales, se volverá a realizar una subdivisión o bien una fusión entre ellas, dependiendo de si se verifican las condiciones prefijadas.

La estructura más usada para la subdivisión es el árbol cuaternario. Los pasos a seguir son:

1. Se define un test de homogeneidad
2. Se subdivide la imagen en los cuatro cuadrantes disjuntos
3. Se calcula la medida de homogeneidad para cada cuadrante
4. Se fusionan dos regiones si la condición de homogeneidad se verifica para la unión de las mismas.
5. Si una región no verifica la condición, se vuelve a subdividir en sus cuatro cuadrantes y se repite el proceso hasta que todas las regiones pasan el test de homogeneidad.



# qtdecomp

---

- $S = \text{qtdecomp}(I, \text{umbral})$
- Divide a la imagen en 4 cuadrados de igual tamaño y trata de encontrar algún criterio de homogeneidad. Si se verifica algún criterio este cuadrado no se divide. Se repite el proceso.
- $S(k,m)$  si es distinto de cero  $(k,m)$  es la esquina superior y el tamaño del bloque está dado por  $S(k,m)$ .
- Divide en cuadrados si el valor mayor-menor es mayor que el umbral.

```

I=[1    1    1    1    2    3    6    6
    1    1    2    1    4    5    6    8
    1    1    1    1   10   15    7    7
    1    1    1    1   20   25    7    7
    20   22   20   22    1    2    3    4
    20   22   22   20    5    6    7    8
    20   22   20   20    9   10   11   12
    22   22   20   20   13   14   15   16];

```

```

S = qtdecomp(I,5);

```

```

full(S)

```

```

ans =

```

```

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

```

# qtgetblk

---

- `[vals,r,c] = qtgetblk(I, S, dim)`
- Vals= devuelve un arreglo dim x dim
- dim x dim x k
- r= fila
- c= columna
- Esquina superior izquierda

[vals,r,c] = qtgetblk(I,S,4)

vals(:,:,1) =

---

1	1	1	1
1	1	2	1
1	1	1	1
1	1	1	1

r =

1  
5

c =

1  
1

vals(:,:,2) =

20	22	20	22
20	22	22	20
20	22	20	20
22	22	20	20