

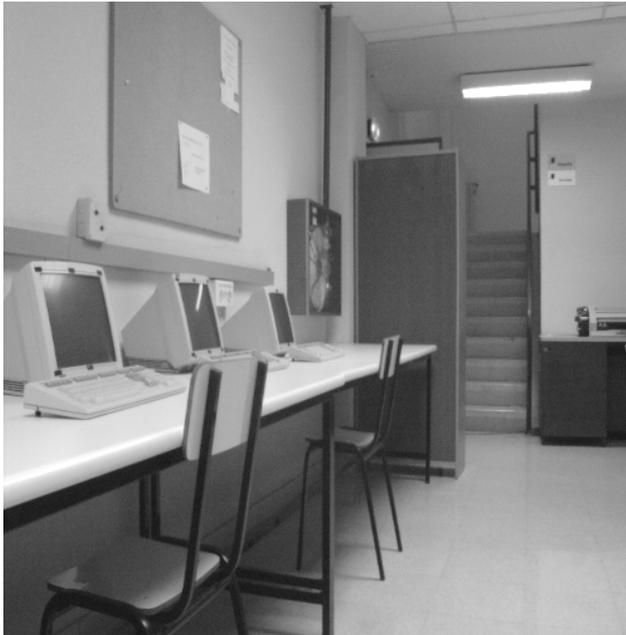
Tema 6: Contornos

1. Introducción
2. Filtrado de imágenes: Convolución
3. Detección de Contornos
 - Detectores de Gradiente: Sobel, Canny, ...
 - Cruces por Cero: Marr-Hildreth
4. Segmentación de Contornos
 - Seguimiento Local
 - Transformada de Hough



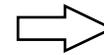
Detección de contornos

- Extracción de contornos: Separación de elementos significativos de la superficie de los objetos
- Los contornos resumen la mayor parte de la información de una imagen



Extracción de Contornos

- Detección de puntos de contorno
 - Cambios de luminosidad en la imagen
 - » Máximos del Gradiente
(1ª derivada de la luminosidad)
 - » Cruces por cero del Laplaciano
(2ª derivada de la luminosidad)



Máscaras
de
convolución

- Segmentación de contornos
 - Agrupación de puntos de contorno
 - Eliminación de ruidos
- Ajuste de rectas o curvas



Ejemplo: Contornos de Canny

- Operador de Canny: derivada de la Gaussiana

**Módulo y orientación
del gradiente**

- Detección de máximos

**Máximos locales
del gradiente**

- Segmentación de contornos:
 - Seguimiento local + Umbral con histéresis

**Cadenas de pixels
de contorno**

- Ajuste de rectas o curvas

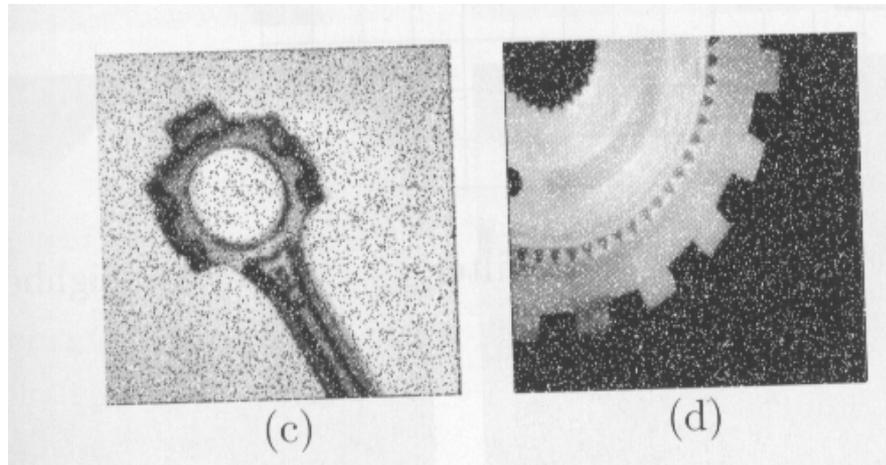
**Rectas y curvas
de contorno**



2. Filtrado de imágenes

- Ruidos Típicos en una imagen:
 - ruido de “sal y pimienta”: ocurrencias aleatorias de pixeles completamente blancos y completamente negros.
 - ruido gaussiano: el nivel de gris de **todos** los pixeles de la imagen es perturbado con una normal

$$I[i, j] = I_{\text{ideal}}[i, j] + p \quad p \approx N(0, \sigma^2)$$



Filtrado lineal: Convolución

- Cada pixel de la imagen resultado es combinación lineal de varios pixels vecinos de la imagen original
- Máscara de convolución (Kernel):

$$K[m, n] = \begin{bmatrix} k_{-a, -c} & \cdots & k_{-a, 0} & \cdots & k_{-a, d} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ k_{0, -c} & \cdots & k_{0, 0} & \cdots & k_{0, d} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ k_{b, -c} & \cdots & k_{b, 0} & \cdots & k_{b, d} \end{bmatrix}$$

$$g[i, j] = K[m, n] * f[i, j] = \sum_{m=-a}^b \sum_{n=-c}^d K[m, n] f[i + m, j + n]$$

- Normalmente, máscaras centradas
- Si no, se marca el pixel central



Convolución: Ejemplos

- Máscara descentrada
 - Diferencia hacia adelante

$$K = \begin{bmatrix} -1 & 1 \end{bmatrix} \quad g(i,j) = f(i,j+1) - f(i,j)$$

- Máscara centrada
 - Filtro binomial 3x3

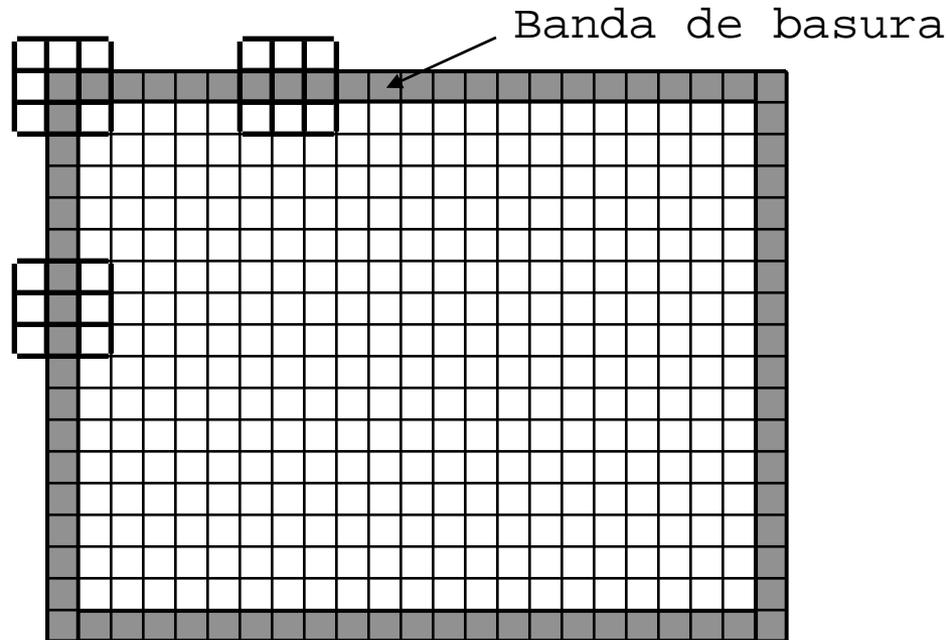
$$B_{3 \times 3} = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$$g(i,j) = \frac{1}{16} [f(i-1,j-1) + 2f(i-1,j) + f(i-1,j+1) + 2f(i,j-1) + 4f(i,j) + 2f(i,j+1) + f(i+1,j-1) + 2f(i+1,j) + f(i+1,j+1)]$$



Convolución: banda de basura

- La máscara “se sale” de la imagen



- Para una máscara $n \times n$ con n impar, el tamaño de la banda de basura es: $(n-1)/2$
 - Dejar valores no válidos (basura)
 - Mejor rellenarla con ceros

Suavizado (filtrado de ruidos)

- Filtro de media

- Todos los píxeles igual peso

$$K_{3 \times 3} = \frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

- Filtro binomial

- Valores sacados del triángulo de Pascal:

$$\begin{array}{cccccc} & & & & & 1 \\ & & & & 1 & & 1 \\ & & & 1 & & 2 & & 1 \\ & & 1 & & 3 & & 3 & & 1 \\ 1 & & 4 & & 6 & & 4 & & 1 \end{array}$$

- Píxeles más lejanos, menor peso:

$$B_{3 \times 3} = \frac{1}{16} \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

- Para no alterar el brillo medio, se divide por la suma de los valores de la máscara



Suavizado de la imagen

- Filtro Gaussiano

$$G_{\sigma}(i, j) = c \cdot e^{-\frac{(i^2 + j^2)}{2 \sigma^2}}$$

- La constante c se calcula para que los coeficientes sumen 1
 - A mayor σ , mayor suavizado
 - Tamaño de la máscara: $5\sigma - 7\sigma$
- Ejemplo: máscara 5×5 para $\sigma = 1$

0.0183	0.0821	0.1353	0.0821	0.0183
0.0821	0.3679	0.6065	0.3679	0.0821
0.1353	0.6065	1.0000	0.6065	0.1353
0.0821	0.3679	0.6065	0.3679	0.0821
0.0183	0.0821	0.1353	0.0821	0.0183

$$c = 1/6.1689$$



Suavizado de la imagen

- Filtrado Gaussiano con $\sigma = 1, 2, 3, 4$

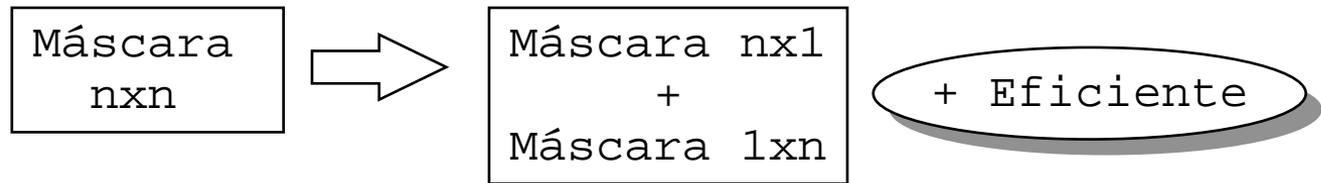


Máscaras separables

- Condición: $\text{rango}(K) = 1$

$$K(i, j) = K_y(i) \cdot K_x(j)$$

$$g(i, j) = K(i, j) * f(i, j) = K_y(i) * [K_x(j) * f(i, j)] = K_x(j) * [K_y(i) * f(i, j)]$$



- Filtro binomial:

$$B_{3 \times 3} = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \frac{1}{16} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

- Filtro Gaussiano:

$$G_{\sigma=1} = \frac{1}{6.1689} \begin{bmatrix} 0.1353 \\ 0.6065 \\ 1.0000 \\ 0.6065 \\ 0.1353 \end{bmatrix} \cdot \begin{bmatrix} 0.1353 & 0.6065 & 1.0000 & 0.6065 & 0.1353 \end{bmatrix}$$



Máscaras separables

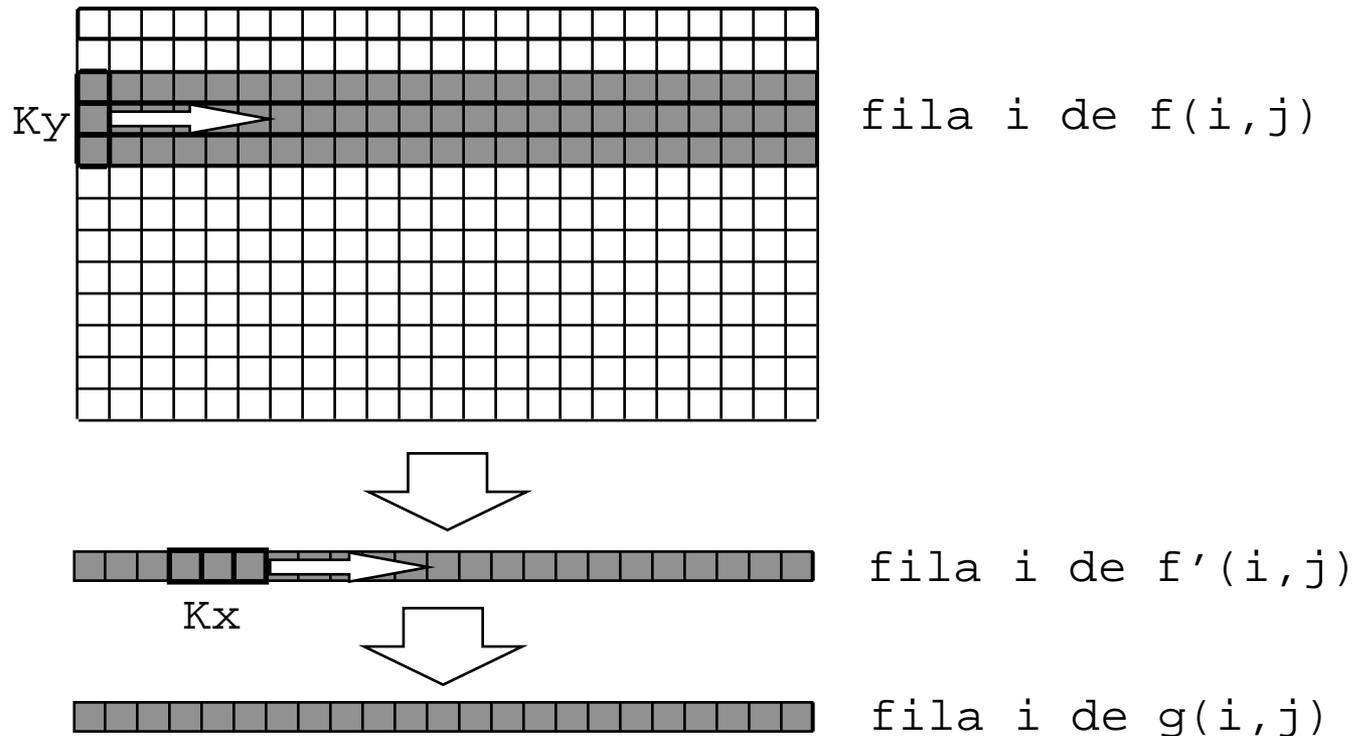
- Programación eficiente:

$$g(i, j) = K(i, j) * f(i, j) = K_x(j) * [K_y(i) * f(i, j)]$$

$$f'(i, j) = K_y(i) * f(i, j)$$

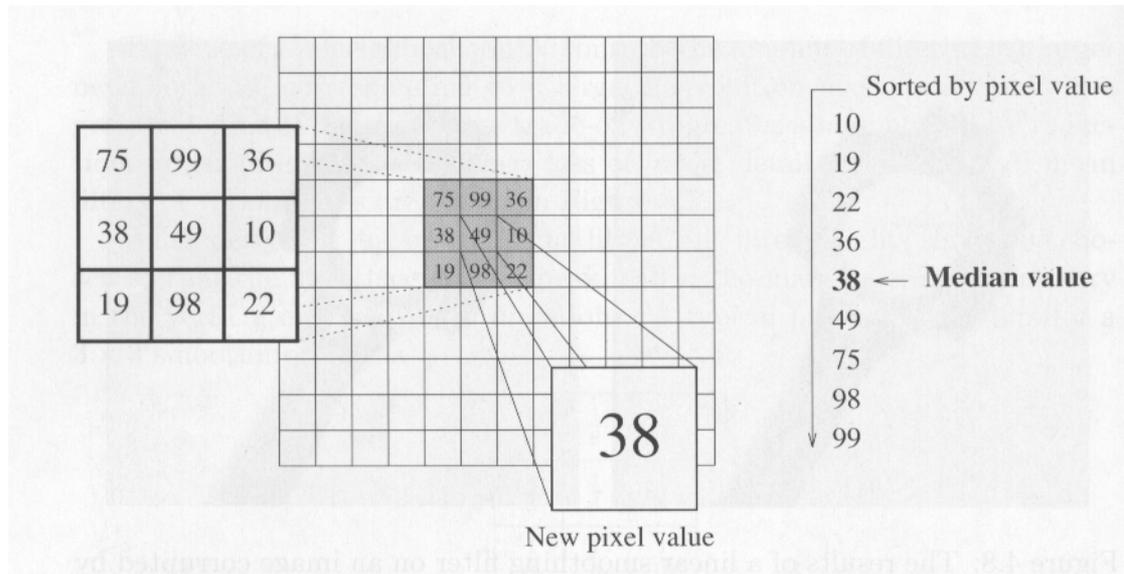
$$g(i, j) = K_x(j) * f'(i, j)$$

- Sólo se necesita una fila intermedia:

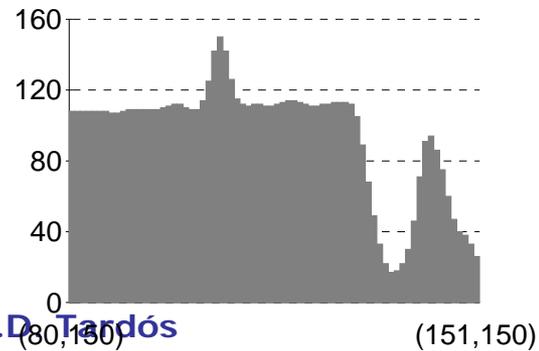
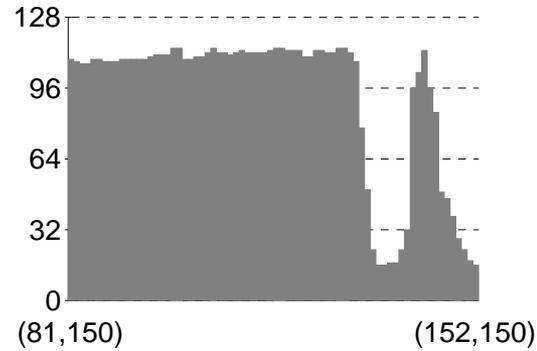
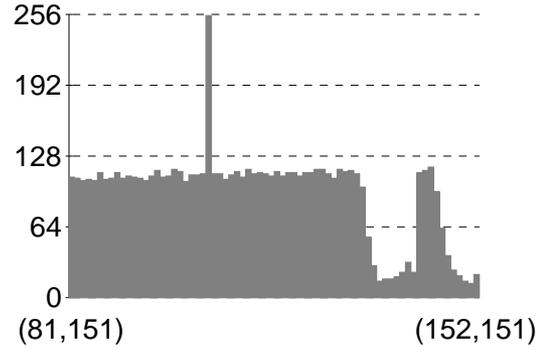


Filtro de mediana

- Filtro no lineal
 - ordenar los $n \times n$ vecinos por nivel de gris ascendente
 - asignar la mediana, esto es, el valor que ocupe la mitad en la lista ordenada
- Buena eliminación de ruido de sal y pimienta
- Los espurios, (outliers) no tienen ninguna influencia

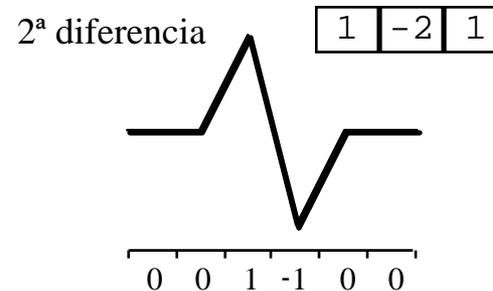
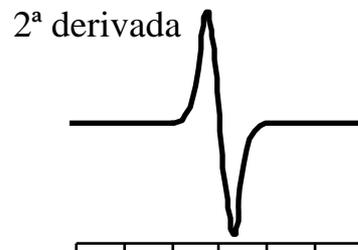
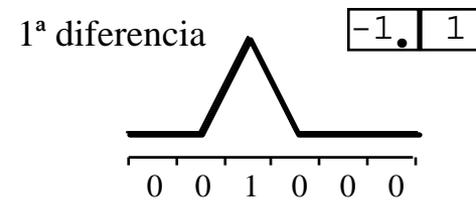
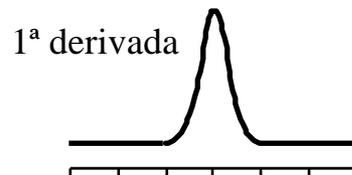
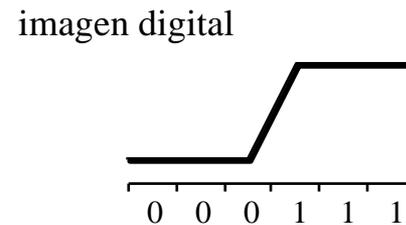
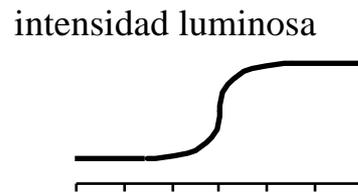


Filtro de mediana



3. Detección de Puntos de Contorno

- Máximo local del gradiente (1ª derivada)
- Cruce por cero del Laplaciano (2ª derivada)



3.1 Detección con el Gradiente

- Gradiente:

Gradiente $\vec{\nabla}f(x, y) = (\nabla_x, \nabla_y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right)$

Módulo $|\nabla f| = \sqrt{(\nabla_x)^2 + (\nabla_y)^2}$

Orientación $\theta = \text{atan2}(\nabla_y, \nabla_x)$

- Operadores:

∇_x	∇_y
-1. 1	1. -1

1ª diferencia

Roberts

0. 1	1. 0
-1 0	0 -1

Prewitt

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

Sobel

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

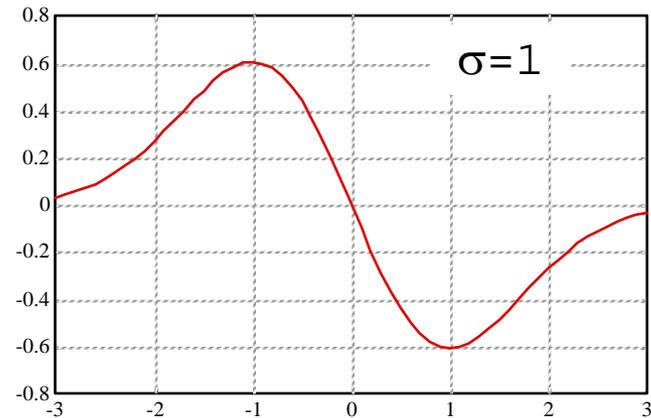
Dividir por la suma de los valores positivos



Gradiente: Operador de Canny

- Optimiza tres criterios:
 - Robustez de detección frente al ruido
 - Precisión en la localización
 - Unicidad de la respuesta
- Similar a la derivada de una Gaussiana

$$G'_\sigma(x) = \frac{-x}{\sigma^2} \exp\left(\frac{-x^2}{2\sigma^2}\right)$$



- Tamaño de la máscara: $5\sigma-7\sigma$
- Ejemplo, para $\sigma = 1$

0.2707	0.6065	0.0	-0.6065	-0.2707
--------	--------	-----	---------	---------

Dividir por la suma de los valores positivos



Gradiente: Operador de Canny

- Cálculo de gradiente en x e y
 - Pueden utilizarse máscaras separadas

$$\nabla_x = G'_\sigma(x) * G_\sigma(y) * f(i, j)$$

$$\nabla_y = G_\sigma(x) * G'_\sigma(y) * f(i, j)$$

– Para el caso $\sigma = 1, n = 5$

$$\nabla_x = \frac{1}{K} \begin{bmatrix} -0.2707 & -0.6065 & 0.0 & 0.6065 & 0.2707 \end{bmatrix} * \begin{bmatrix} 0.1353 \\ 0.6065 \\ 1.0 \\ 0.6065 \\ 0.1353 \end{bmatrix} * f(i, j)$$

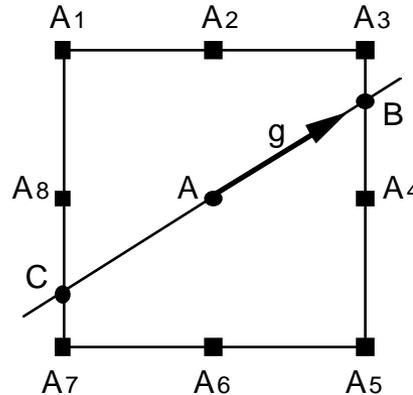
$$\nabla_y = \frac{1}{K} \begin{bmatrix} 0.1353 & 0.6065 & 1.0 & 0.6065 & 0.1353 \end{bmatrix} * \begin{bmatrix} 0.2707 \\ 0.6065 \\ 0.0 \\ -0.6065 \\ -0.2707 \end{bmatrix} * f(i, j)$$



Máximos del Gradiente

Máximos locales del módulo en la dirección del Gradiente

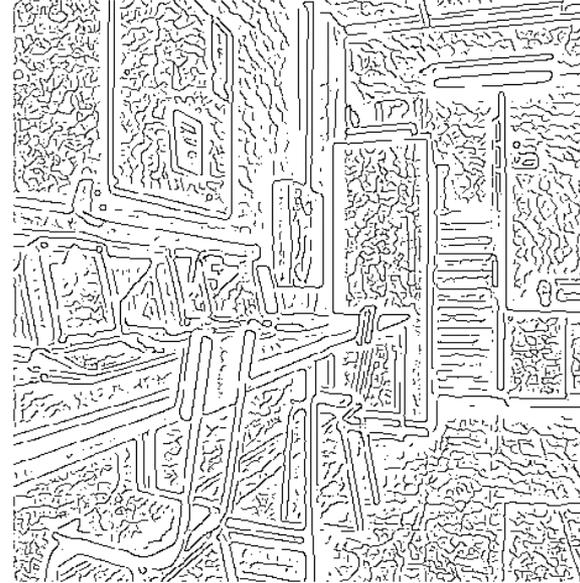
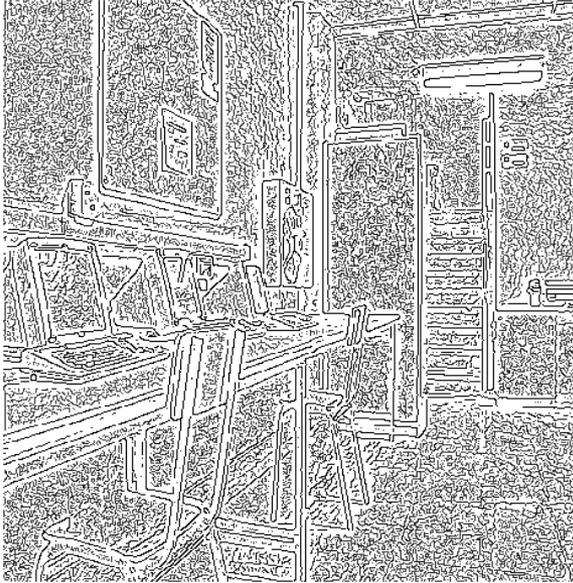
- Para cada pixel con módulo mayor que un umbral, se toma una ventana 3x3



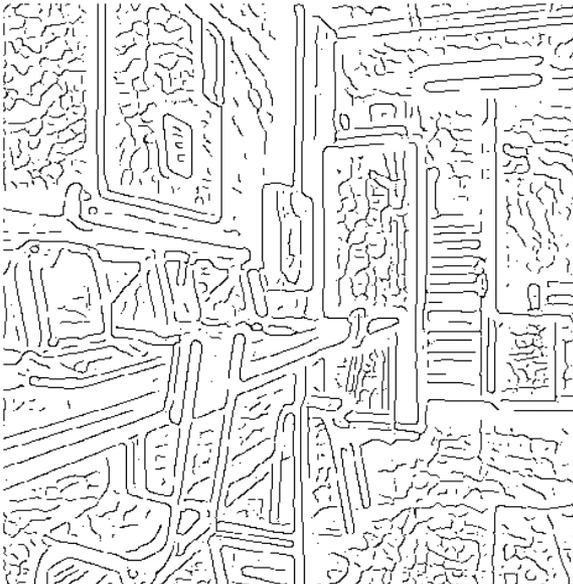
- Se calcula el módulo del gradiente en B y C, interpolando linealmente entre los valores de A3-A4 y A7-A8
- El pixel A es punto de contorno si su módulo es mayor que el de B y mayor o igual que el de C



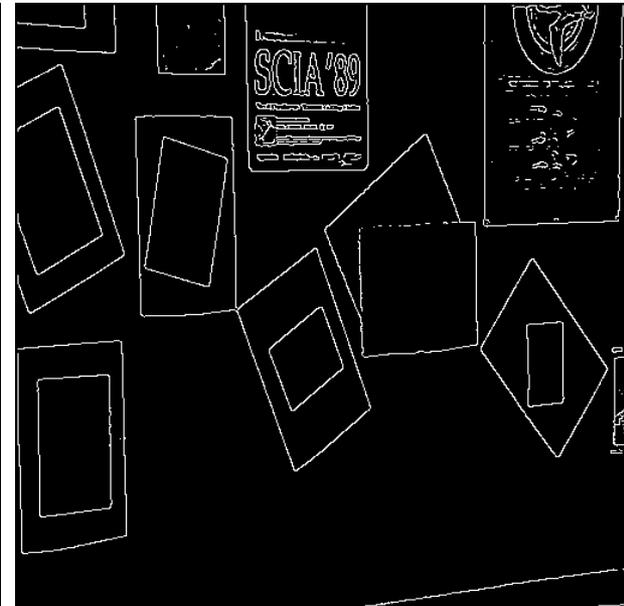
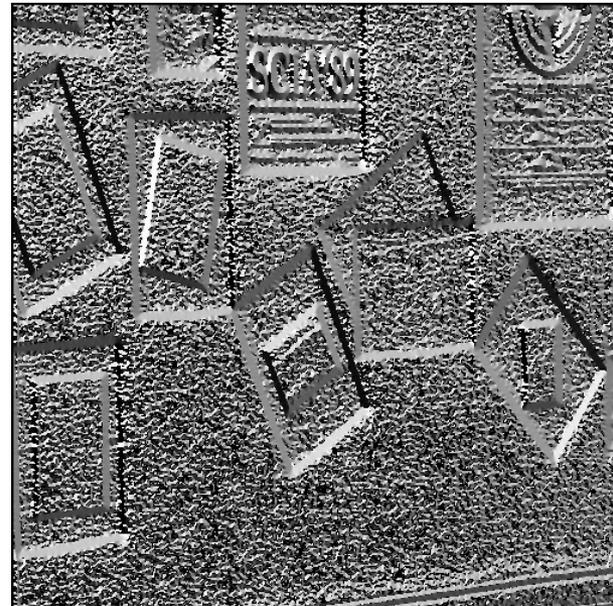
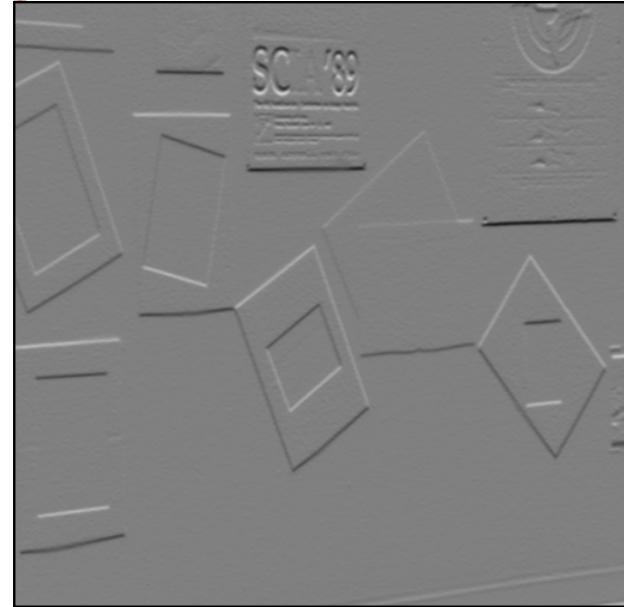
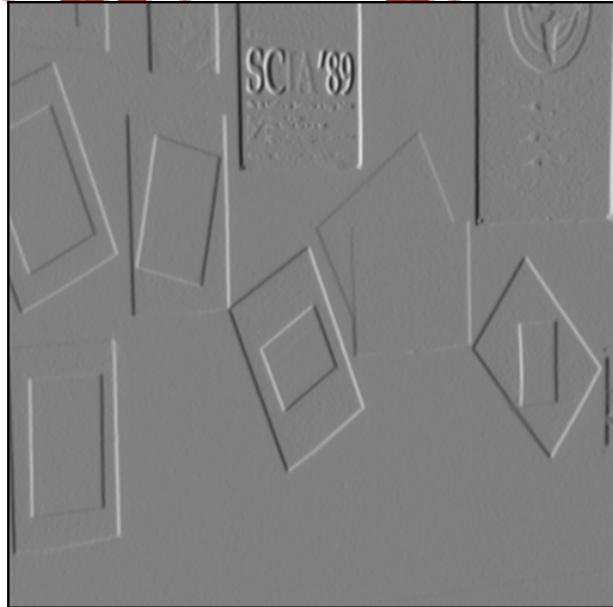
Máximos del Gradiente



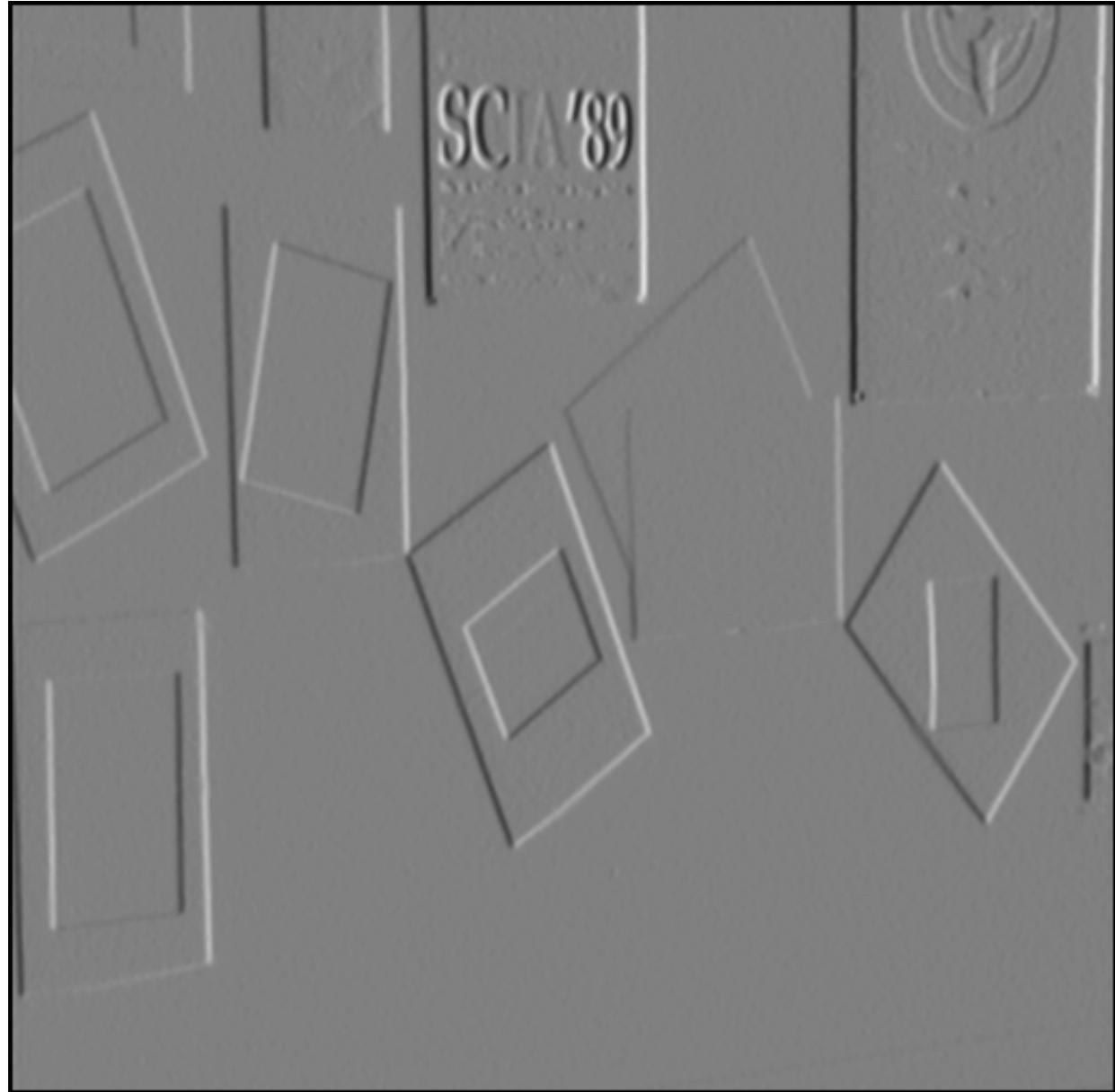
$\sigma = 1, 2, 3, 4$



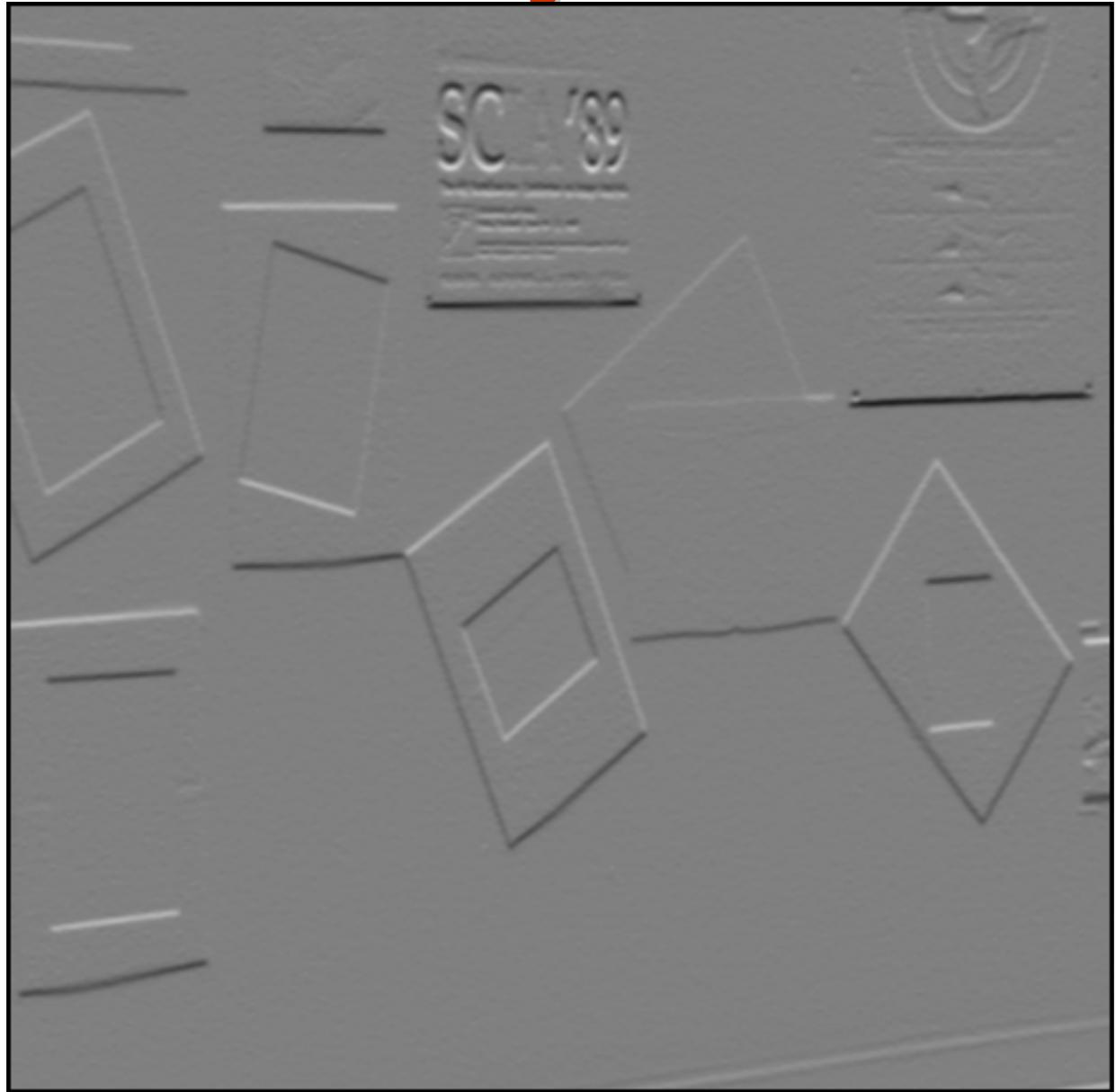
Imagen, gx, gy, mag, ori, contornos



Gradiente: G_x



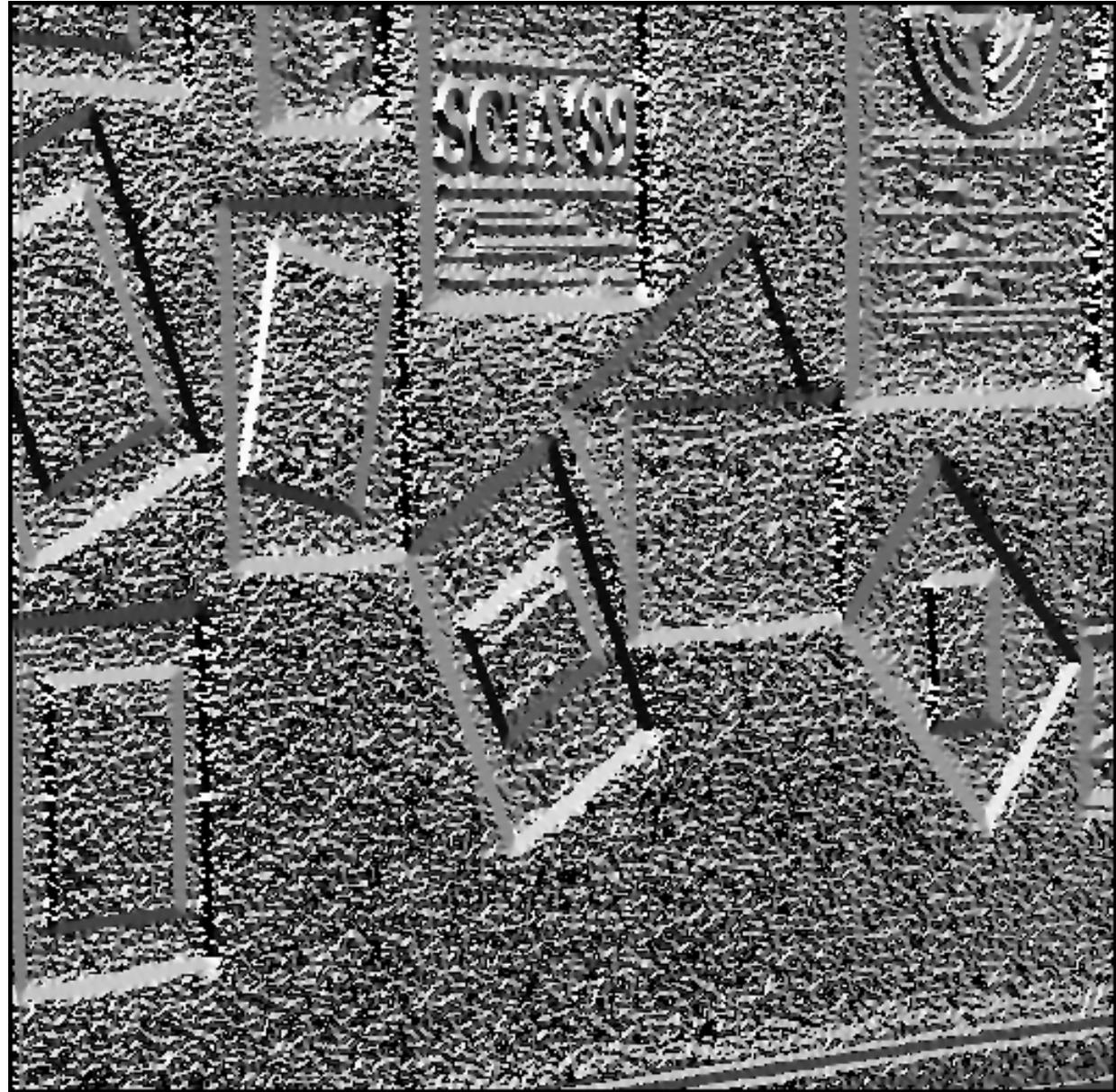
Gradiente: Gy



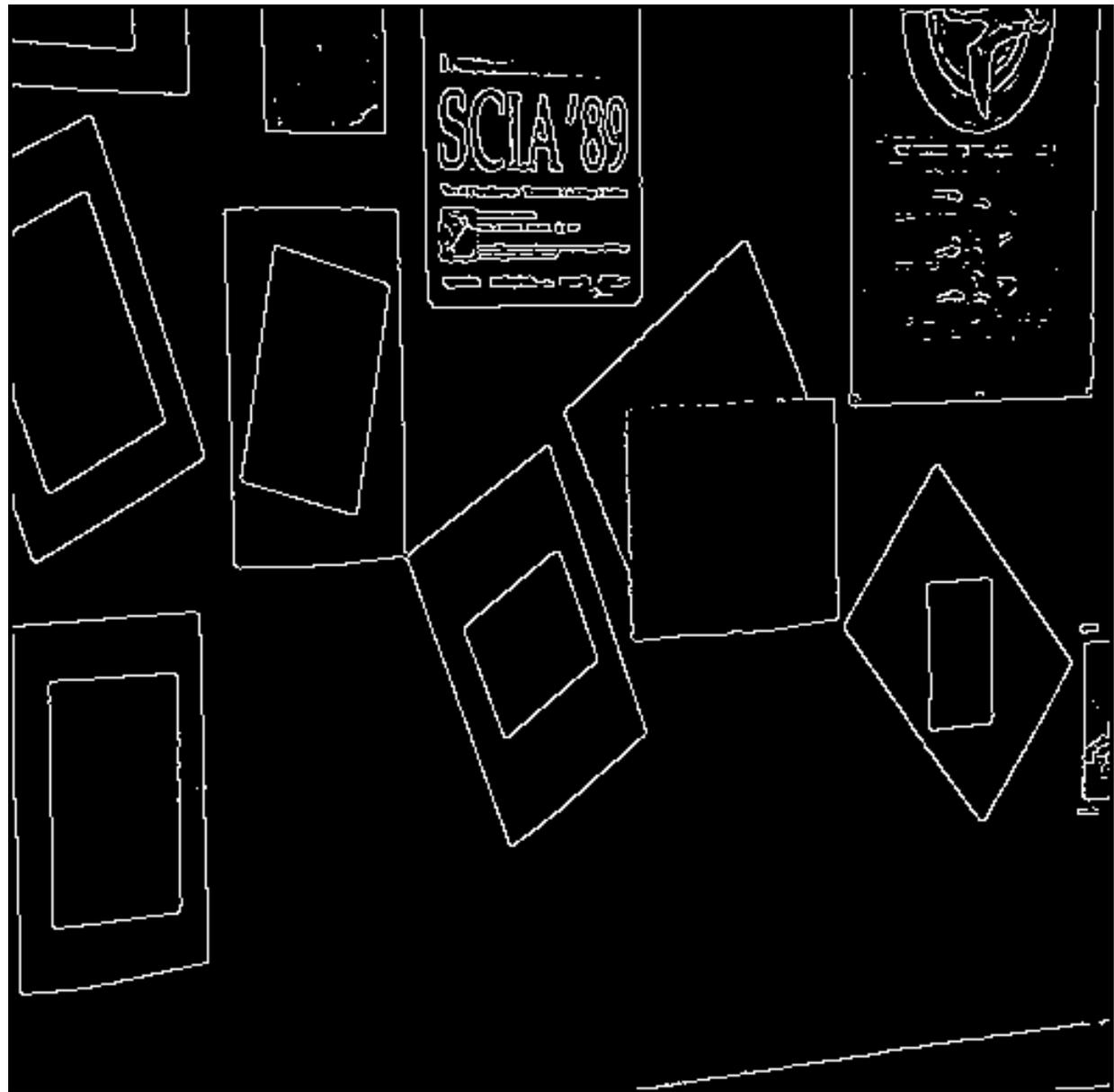
Magnitud del Gradiente



Orientación del Gradiente



Contornos



3.2 Detección con la 2ª derivada

- Laplaciano: $\nabla^2 f(x,y) = \nabla^2_x + \nabla^2_y = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

2ª diferencia

$$\nabla^2_x \quad \nabla^2_y$$

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

1
-2
1

- Operadores para el Laplaciano

0	1	0
1	-4	1
0	1	0

1	4	1
4	-20	4
1	4	1

- Cruces por cero: contornos conexos, cerrados, y delgados
- La 2ª derivada amplifica mucho el ruido: necesario filtrar

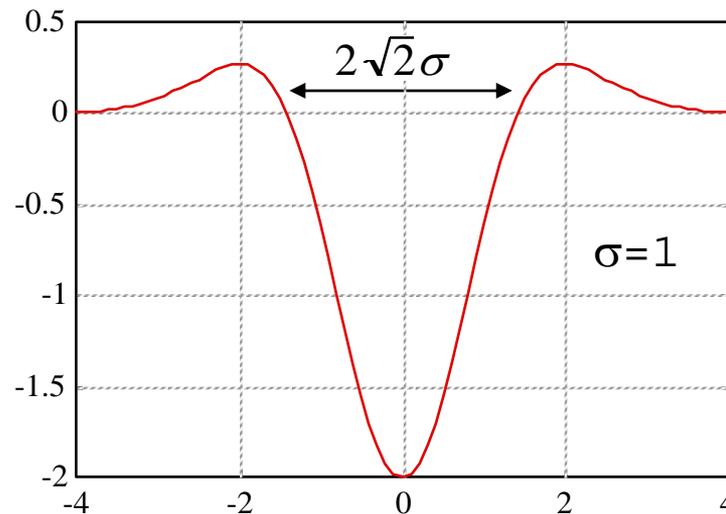


Operador de Marr-Hildreth

- Filtro Gaussiano + Operador Laplaciano

$$\nabla^2 * G_\sigma * f(i,j) = (\nabla^2 G_\sigma) * f(i,j)$$

$$\nabla^2 G_\sigma(r) = \frac{r^2 - 2\sigma^2}{\sigma^4} \exp\left(\frac{-r^2}{2\sigma^2}\right)$$



Sombrero mexicano
(invertido)



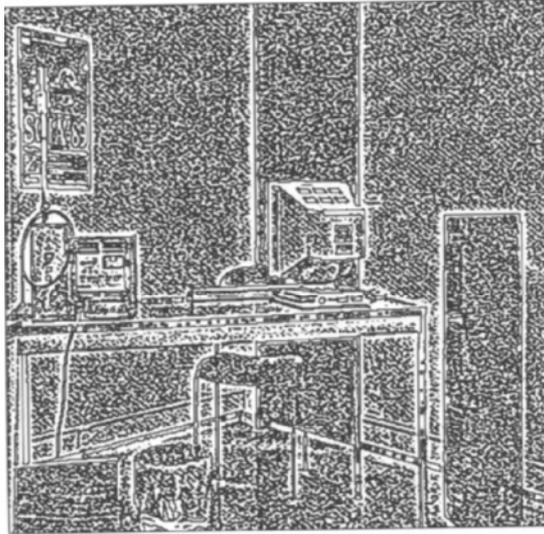
Contornos: Cruces por cero

- Se consideran contornos los cruces por cero de magnitud superior a un umbral

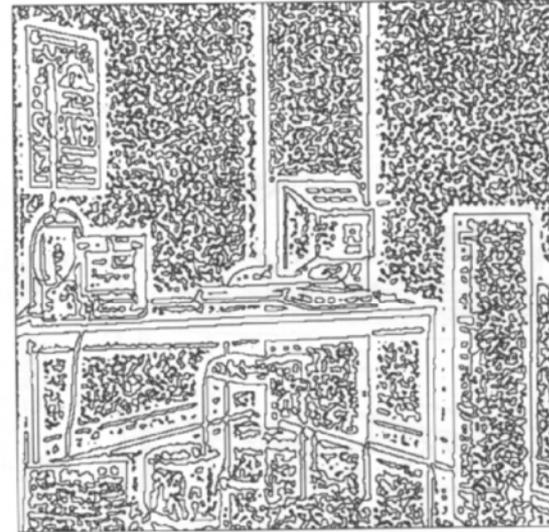
```
my:= 0
mx:= 0
si g(i,j) >= 0 entonces
  si g(i+1,j) < 0 entonces
    my:= g(i,j) - g(i+1,j)
  fsi
  si g(i,j+1) < 0 entonces
    mx:= g(i,j) - g(i,j+1)
  fsi
sino
  si g(i+1,j) >= 0 entonces
    my:= g(i+1,j) - g(i,j)
  fsi
  si g(i,j+1) >= 0 entonces
    mx:= g(i,j+1) - g(i,j)
  fsi
fsi
magnitud(i,j) := max(my,mx)
```



Contornos: Cruces por cero



Zero crossings of $\nabla^2 G$ with $\sigma=1$.



Zero crossings of $\nabla^2 G$ with $\sigma=2$.



Zero crossings of $\nabla^2 G$ with $\sigma=3$.



4. Segmentación de Contornos

- Objetivos:
 - Extracción de rectas o curvas
 - Eliminación de contornos espúreos
- Técnicas:
 - Análisis local:
 - » Seguimiento de contornos
 - Umbral con histéresis
 - » Ajuste de rectas o curvas
 - División recursiva
 - Mínimos cuadrados
 - Transformada de Hough
 - Método de Burns: segmentación de rectas usando la orientación del gradiente

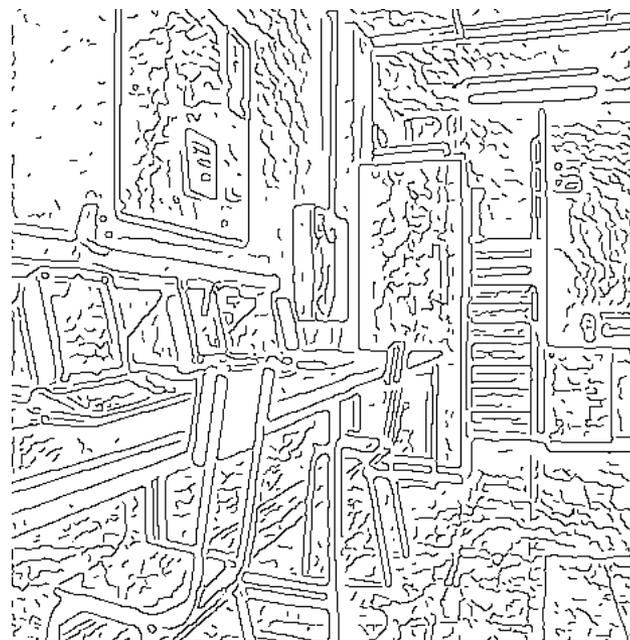


4.1 Seguimiento de Contornos

- Objetivo: cadenas de puntos de contorno de una cierta "intensidad" (módulo del gradiente o magnitud del cruce por cero)
- Elegir un valor de umbral es difícil



Umbral = 8
Contornos rotos



Umbral = 4
Demasiado ruido



Umbral con histéresis

- Cada cadena de contorno debe tener:
 - Todos los pixels $\geq U_{inf}$
 - Al menos un pixel $\geq U_{sup}$
- Además filtrado por longitud del contorno



$U_{inf}=4, U_{sup}=8$



$U_{inf}=4, U_{sup}=8, Long > 30$



Seguimiento de Contornos

```
procedimiento encadenar_contornos
  para i:= 1 hasta nfilas
    para j:= 1 hasta ncolumnas
      inicial:=(i,j)
      si modulo(inicial) >= Umbral_sup entonces
        n:= 0
        seguir(inicial,cadena,n)
        reverse(cadena, n)
        seguir(siguiete(inicial),cadena,n))
      si n >= longitud_minima entonces
        guardar(cadena,n)
      fsi
    fsi
  fpara
fpara
fin encadenar_contornos

procedimiento seguir(actual,var cadena,var n)
  mientras actual <> nulo
    n:=n+1
    cadena(n):= actual
    modulo(actual):= 0
    actual:= siguiete(actual)
  fmientras
fin seguir

funcion siguiete(actual) devuelve pixel
  para sigte en vecinos(actual)
    si modulo(sigte) >= Umbral_inf entonces
      devolver sigte
  fsi
  fpara
  devolver nulo
fin siguiete
```

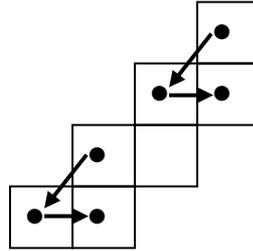
En qué orden?



Seguimiento: en qué orden?

- Orden de lectura

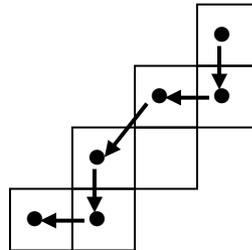
1	2	3
4		5
6	7	8



Contono Roto

- Primero los 4-vecinos

5	1	6
2		3
7	4	8



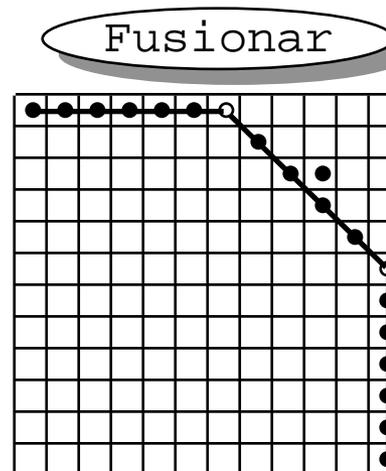
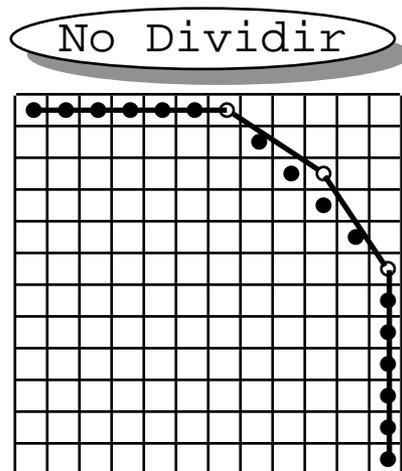
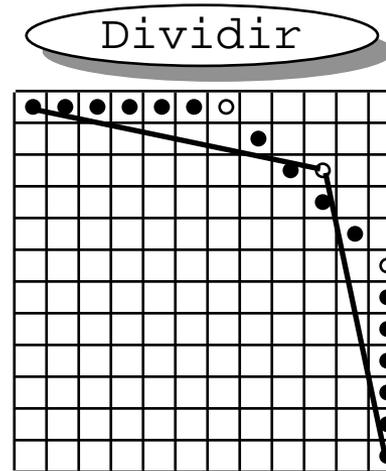
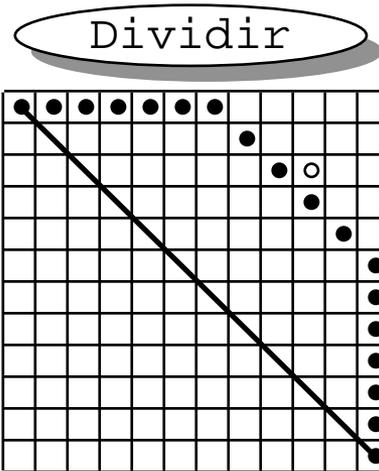
División recursiva en rectas

1. División recursiva de una cadena:
 - Recta que une los extremos de la cadena
 - Punto que más se separa de la recta
 - Si separación $>$ umbral (ej: 2 pixels), partir la cadena por dicho punto
 - Repetir para las dos subcadenas obtenidas
2. Fusión de segmentos:
 - Dos segmentos adyacentes se fusionan si la desviación del pixel más alejado de la recta es menor que el umbral
3. Eliminar segmentos muy cortos o con gradiente pequeño
4. Recta definida por los extremos de la cadena, o ajustada por mínimos cuadrados



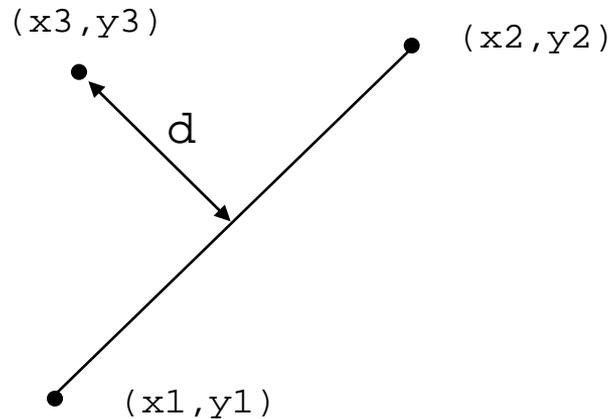
División recursiva en rectas

- Ejemplo



División recursiva en rectas

- Distancia de un punto a la recta que pasa por otros dos puntos:



$$d = \frac{|x_3(y_1 - y_2) - y_3(x_1 - x_2) + y_2x_1 - y_1x_2|}{\sqrt{(y_1 - y_2)^2 + (x_1 - x_2)^2}}$$



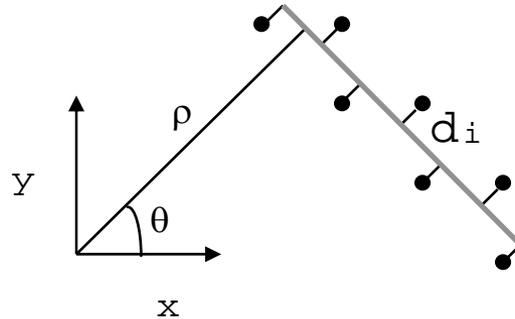
División y ajuste de rectas

- Cadenas de contornos y rectas obtenidas



Ajuste de rectas

- Mínimos cuadrados (regresión total)
 - Minimizar la suma del cuadrado de las distancias pixel-recta



Ecuación de la recta: $x \cos \theta + y \sin \theta - \rho = 0$

Distancia del pixel i : $d_i = x_i \cos \theta + y_i \sin \theta - \rho$

Minimizar: $D^2 = \sum_i (x_i \cos \theta + y_i \sin \theta - \rho)^2$

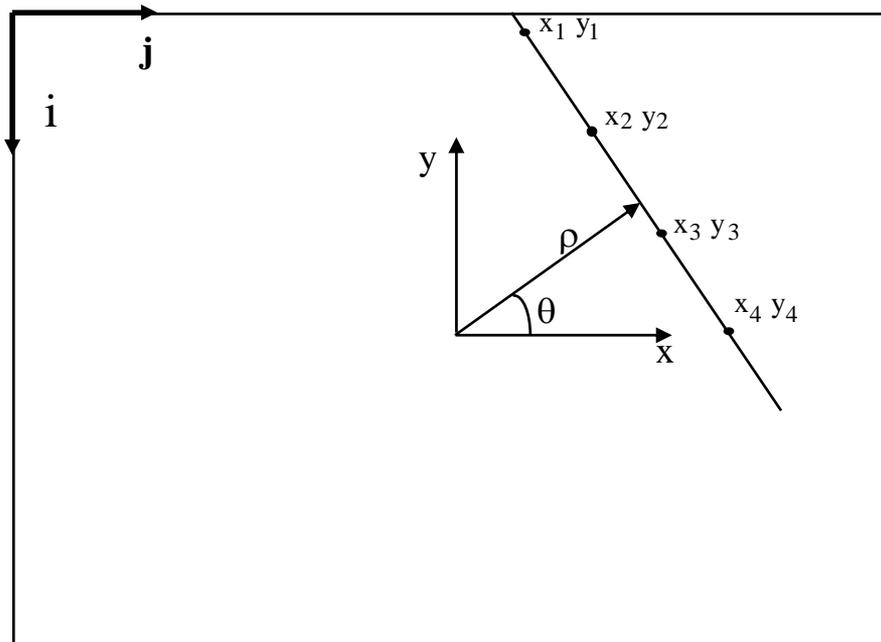
- Solución: eje de "minima inercia"
 - Igual que el calculo de la posición y orientación de un blob.
- Extremos: proyección del pixel inicial y final sobre la recta obtenida



4.2 Transformada de Hough

- Detección global de rectas o curvas
- Se representan de forma paramétrica
- Parámetros para las rectas: ρ y θ

$$\rho = x \cos\theta + y \operatorname{sen}\theta$$

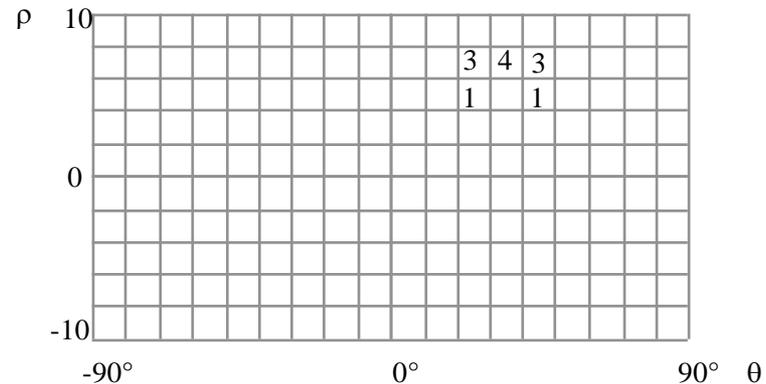
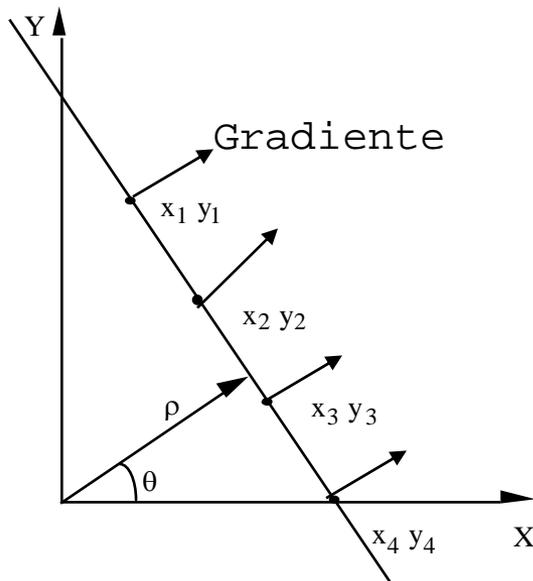


$$x = j - \text{ncolumnas}/2$$
$$y = \text{nfilas}/2 - i$$



Tr. Hough: Uso de la orientación

- Tiempo de cálculo excesivo
- Mejora: cada pixel solo vota en orientaciones similares a la orientación de su gradiente



Tr. Hough: Uso de la orientación

Sin Orientación

```
para i:= 1 hasta nfilas
  para j:= 1 hasta ncolumnas
    si modulo(i,j)>=Umbral
      x:= j - ncolumnas/2
      y:= nfilas/2 - i
      para  $\theta$ := $\theta$ min hasta  $\theta$ max
         $\rho$ := x*cos( $\theta$ ) + y*sin( $\theta$ )
        votar_recta(i,j, $\rho$ , $\theta$ )
      fpara
    fsi
  fpara
fpara
```

Con Orientación

```
para i:= 1 hasta nfilas
  para j:= 1 hasta ncolumnas
    si modulo(i,j)>=Umbral
      x:= j - ncolumnas/2
      y:= nfilas/2 - i
       $\theta$ := orientacion(i,j)
       $\rho$ := x*cos( $\theta$ ) + y*sin( $\theta$ )
      votar_recta(i,j, $\rho$ , $\theta$ )
    fsi
  fpara
fpara
```



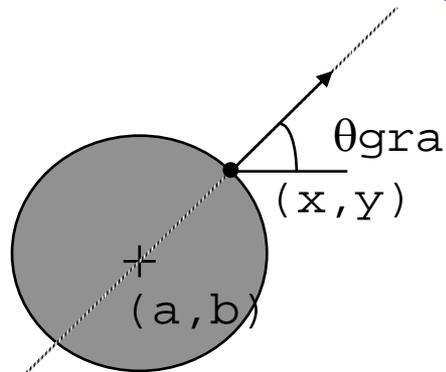
Tr. de Hough: Círculos

- También sirve para curvas, pero el número de parámetros es mayor
- Ejemplo: círculos
 - Radio y Centro (r, a, b)
 - Forma paramétrica (polar)
 - Usando la orientación del gradiente:

$$r^2 = (x - a)^2 + (y - b)^2$$

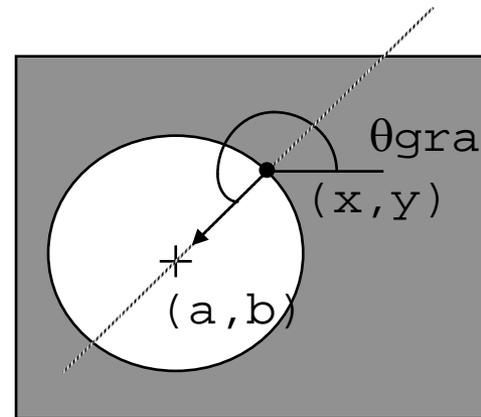
$$x = a + r \cos \theta$$

$$y = b + r \sin \theta$$



$$a = x - r \cos \theta$$

$$b = y - r \sin \theta$$



$$a = x + r \cos \theta$$

$$b = y + r \sin \theta$$

Tr. de Hough: Círculos

- Tabla de acumulación 3D: (r,a,b)

Con Orientación

```
para i:= 1 hasta nfilas
  para j:= 1 hasta ncolumnas
    si modulo(i,j)>=Umbral entonces
      x:= j - ncolumnas/2
      y:= nfilas/2 - i
       $\theta$ := orientacion(i,j)
      para r:= rmin hasta rmax
        a:= x - r*cos( $\theta$ )
        b:= y - r*sin( $\theta$ )
        votar_circulo(i,j,r,a,b)
      fpara
    fsi
  fpara
fpara
```



Aplicación: punto de fuga de un pasillo

- Cada punto de contorno vota a un punto de la línea del horizonte

