Ruidos y Detección de Bordes

Procesamiento Digital de Imágenes

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

1er Cuatrimestre 2021

Restauración

¿Qué es restauración de imágenes?

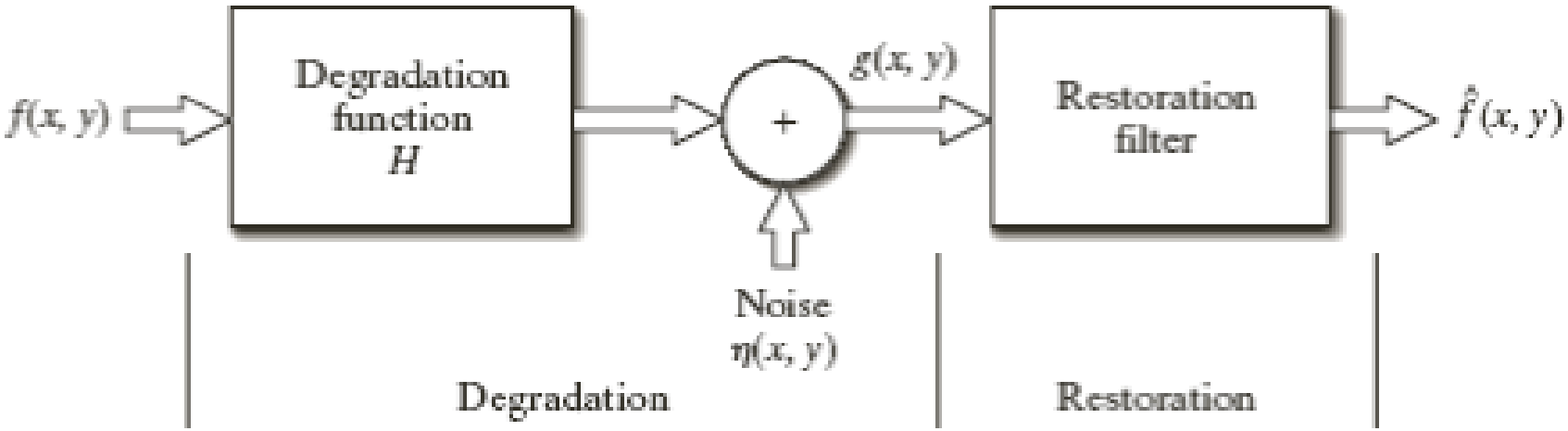
Es una clase de algoritmos que remueven o reducen distintos tipos de distorsiones producidas por:

- el sensor
- desenfoque (fuera de foco)
- movimiento de la cámara
- condiciones climáticas
- fotos antiguas deterioriadas

Objetivo

Minimizar el efecto de las degradaciones

Modelo de imagen



Ruido

Modelo con ruido aditivo

$$g(x, y) = f(x, y) + \eta(x, y),$$

donde

- $f(x, y)$ imagen original sin ruido
- $\eta(x, y)$ imagen de ruido.
- $g(x, y)$ imagen de salida corrupta con ruido aditivo

Ruido

Modelo con ruido multiplicativo

$$g(x, y) = f(x, y) \cdot \eta(x, y),$$

donde $\eta(x, y)$ ruido multiplicativo.

Ruido Uniforme

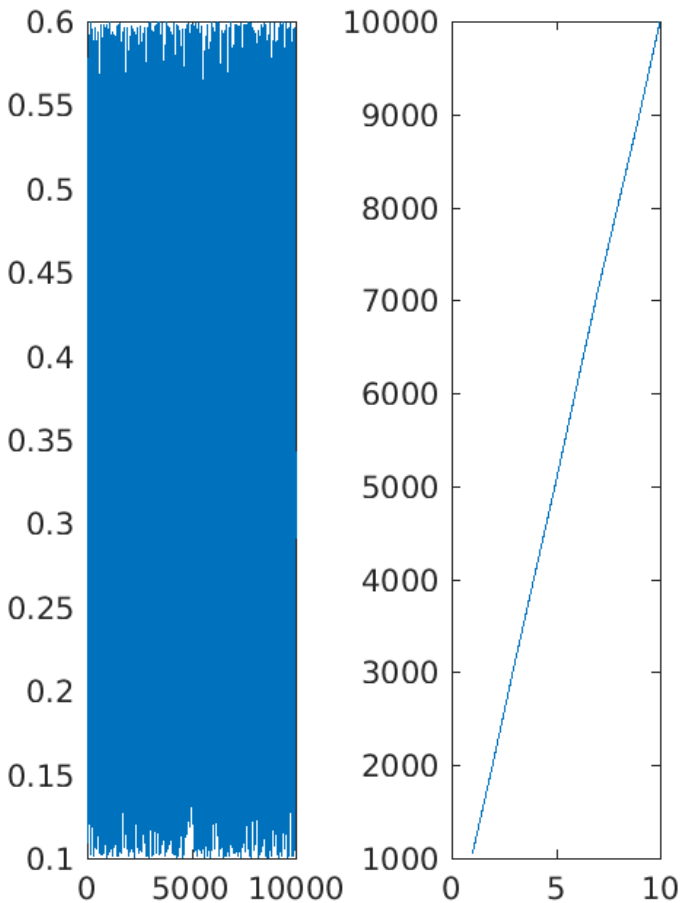
Función de densidad de probabilidad

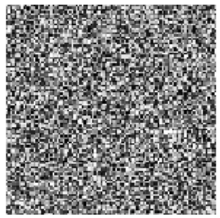
$$p(x) = \begin{cases} \frac{1}{b-a} & a \leq x \leq b \\ 0 & \text{otro caso} \end{cases}$$

- Media: $\frac{b-a}{2}$
- Varianza: $\sigma^2 = \frac{(b-a)^2}{12}$
- Generador: $x = a + (b-a)u, \qquad u \sim \mathcal{U}(0,1)$

Ruido Uniforme

Histograma, Acumulado, Imagen con valores con distribución $\mathcal{U}(0, 1)$





Ruido Gaussiano

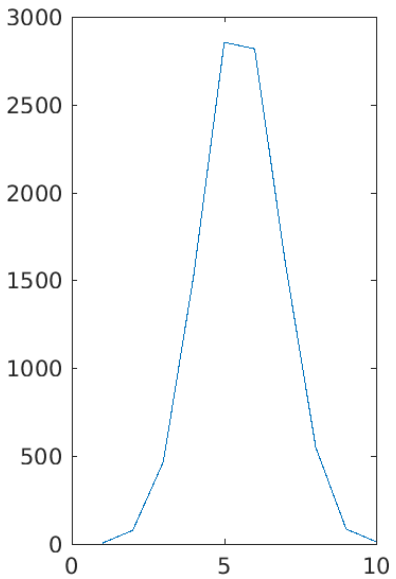
Función de densidad de probabilidad

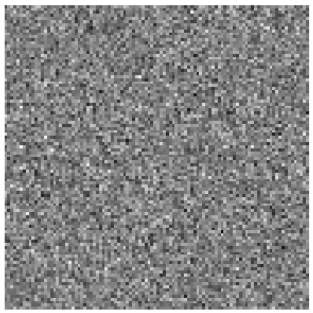
$$p(x) = \frac{1}{\sqrt{2\pi b}} e^{\frac{-(x-a)^2}{b^2}}, \qquad -\infty \leq x \leq \infty$$

- Media: a
- Varianza: b^2
- Generador: $x = a + (b-a)n, \qquad n \sim \mathcal{N}(0,1)$

Ruido Gaussiano

Histograma, Imagen con valores con distribución $\mathcal{N}(0, 1)$





Ruido Rayleigh

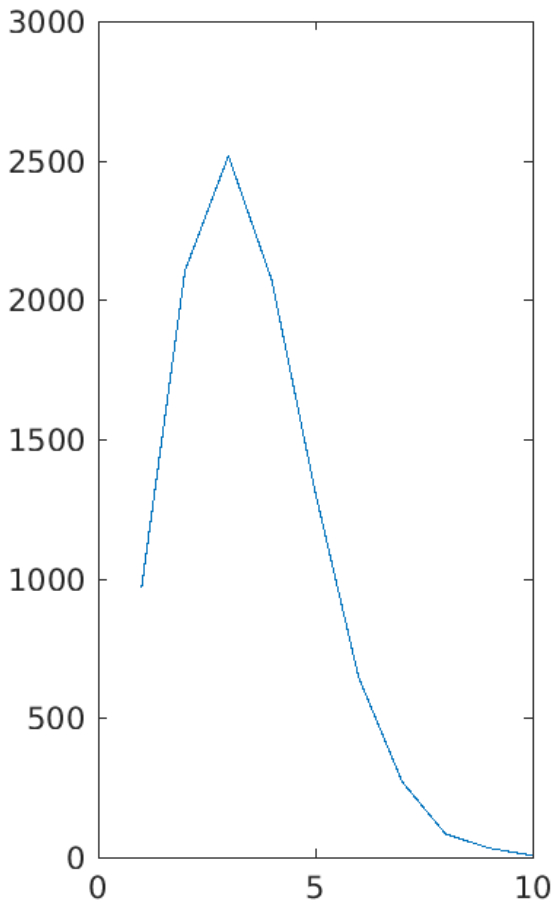
Función de densidad de probabilidad

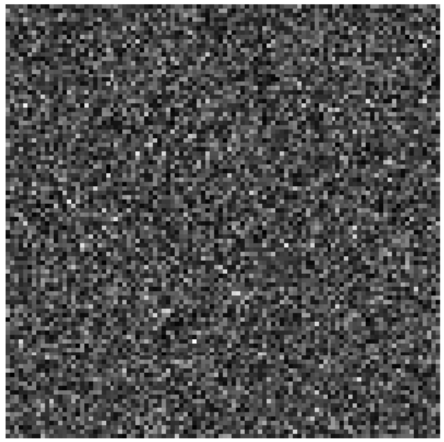
$$p(x) = \begin{cases} \frac{2}{b}(x-a)e^{\frac{-(x-a)^2}{b}} & x \geq a \\ 0 & x < a \end{cases}$$

- Media: $a + \sqrt{\frac{\pi b}{4}}$
- Varianza: $\sigma^2 = \frac{b(4-\pi)}{4}$
- Generador:$x = a + \sqrt{-b \ln[1-u]}, \qquad u \sim \mathcal{U}(0,1)$

Ruido Rayleigh

Histograma, Imagen con valores con distribución $\mathcal{R}$





Ruido Exponencial

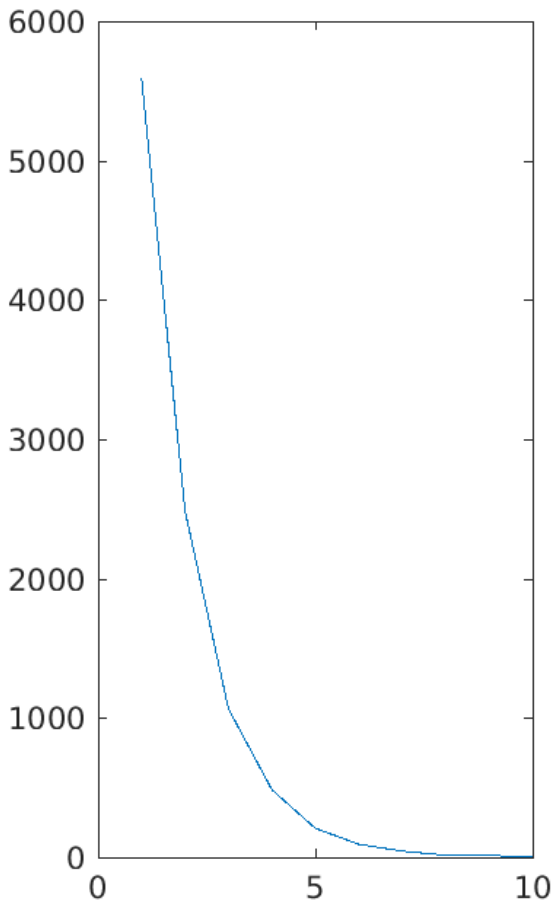
Función de densidad de probabilidad

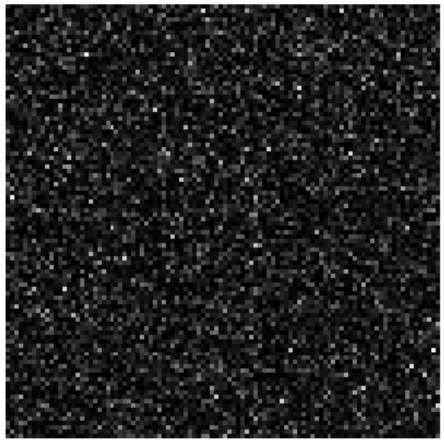
$$p(x) = \begin{cases} ae^{-ax}, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

- Media: $\frac{1}{a}$
- Varianza: $\sigma^2 = \frac{1}{a^2}$
- Generador: $x = -\frac{1}{a} - \ln[1-u], \qquad u \sim \mathcal{U}(0,1)$

Ruido Exponencial

Histograma, Imagen con valores con distribución $\mathcal{E}$





Ruido Impulsivo

Función de densidad de probabilidad

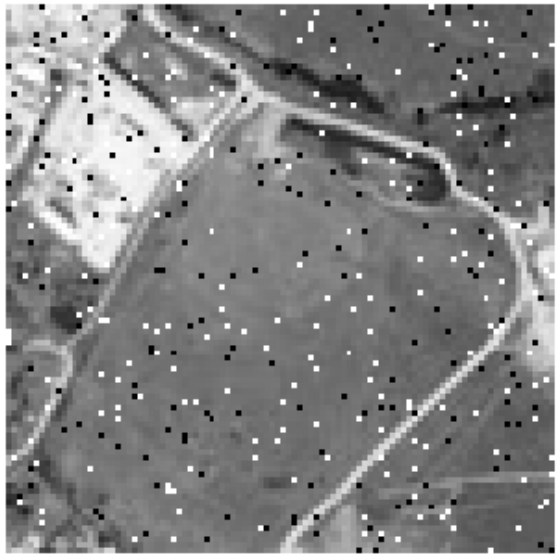
$$p(x) = \begin{cases} P_p & x = 0 \quad (pimienta) \\ P_s & x = 2^n - 1 \quad (sal) \\ 1 - (P_p + P_s) & x = k, \quad 0 < k < 2^n - 1 \end{cases}$$

Generador

para cada (x, y) se genera un número aleatorio, $u \sim \mathcal{U}(0, 1)$,

$$f(x, y) = \begin{cases} 0 & u < P_p \\ 2^n - 1 & u > P_s \\ f(x, y) & \text{otro caso} \end{cases}$$

Sal y Pimienta



Generadores

TABLE 4.1 Generation of random variables.

Name	PDF	Mean and Variance	CDF	Generator[†]
Uniform	$p(z)=\begin{cases}\frac{1}{b-a} & \text{if } 0\le z\le b\\ 0 & \text{otherwise}\end{cases}$	$m=\frac{a+b}{2},\ \sigma^2=\frac{(b-a)^2}{12}$	$F(z)=\begin{cases}0 & z<a\\ \frac{z-a}{b-a} & a\le z\le b\\ 1 & z>b\end{cases}$	MATLAB function `rand`.
Gaussian	$p(z)=\frac{1}{\sqrt{2\pi}b}e^{-(z-a)^2/2b^2}$ $-\infty<z<\infty$	$m=a,\ \sigma^2=b^2$	$F(z)=\int_{-\infty}^{z}p(v)\,dv$	MATLAB function `randn`.
Lognormal	$p(z)=\frac{1}{\sqrt{2\pi}bz}e^{-\left[\ln(z)-a\right]^2/2b^2}$ $z>0$	$m=e^{a+\left(b^2/2\right)},\ \sigma^2=\left[e^{b^2}-1\right]e^{2a+b^2}$	$F(z)=\int_{0}^{z}p(v)\,dv$	$z=e^{bN(0,1)+a}$
Rayleigh	$p(z)=\begin{cases}\frac{2}{b}(z-a)e^{-(z-a)^2/b} & z\ge a\\ 0 & z<a\end{cases}$	$m=a+\sqrt{\pi b/4},\ \sigma^2=\frac{b(4-\pi)}{4}$	$F(z)=\begin{cases}1-e^{-(z-a)^2/b} & z\ge a\\ 0 & z<a\end{cases}$	$z=a+\sqrt{-b\ln\left[1-U(0,1)\right]}$
Exponential	$p(z)=\begin{cases}ae^{-az} & z\ge 0\\ 0 & z<0\end{cases}$	$m=\frac{1}{a},\ \sigma^2=\frac{1}{a^2}$	$F(z)=\begin{cases}1-e^{-az} & z\ge 0\\ 0 & z<0\end{cases}$	$z=-\frac{1}{a}\ln\left[1-U(0,1)\right]$
Erlang	$p(z)=\frac{a^b z^{b-1}}{(b-1)!}e^{-az}$ $z\ge 0$	$m=\frac{b}{a},\ \sigma^2=\frac{b}{a^2}$	$F(z)=\left[1-e^{-az}\sum_{n=0}^{b-1}\frac{(az)^n}{n!}\right]$ $z\ge 0$	$z=E_1+E_2+\cdots+E_b$ (The E's are exponential random numbers with parameter a.)
Salt & Pepper[‡]	$p(z)=\begin{cases}P_p & \text{for } z=0\text{ (pepper)}\\ P_s & \text{for } z=2^n-1\text{ (salt)}\\ 1-(P_p+P_s) & \text{for } z=k\\ & (0<k<2^n-1)\end{cases}$	$m=(0)P_p+k(1-P_p-P_s)+(2^n-1)P_s$ $\sigma^2=(0-m)^2P_p+(k-m)^2(1-P_p-P_s)+(2^n-1-m)^2P_s$	$F(z)=\begin{cases}0 & \text{for } z<0\\ P_p & \text{for } 0\le z<k\\ 1-P_s & \text{for } k\le z<2^n-1\\ 1 & \text{for } 2^n-1\le z\end{cases}$	MATLAB function `rand` with some additional logic.

[†] $N(0,1)$ denotes normal (Gaussian) random numbers with mean 0 and variance 1. $U(0,1)$ denotes uniform random numbers in the range $(0,1)$.

[‡] As explained in the text, salt-and-pepper noise can be viewed as a random variable with three values, which in turn are used to modify the image to which noise is applied. In this sense, the mean and variance are not as meaningful as for the other noise types; we include them here for completeness (the 0s in the equation for the mean and variance

Filtros Pasa bajo

Suavizan la imagen, dejan pasar las frecuencias bajas y atenúan las frecuencias altas.

- media

$$A = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad B = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad C = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

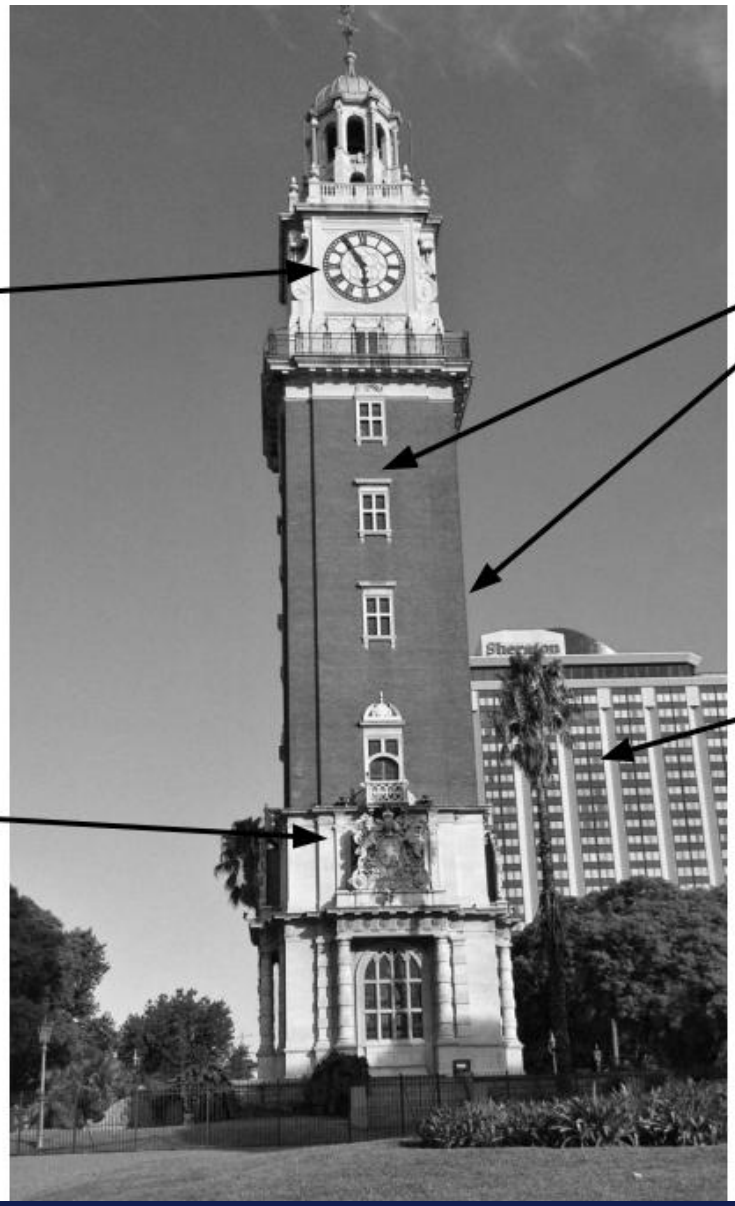
- mediana: filtro no-lineal, ordena la vecindad del pixel y lo reemplaza por el valor central.

Detección de Bordes

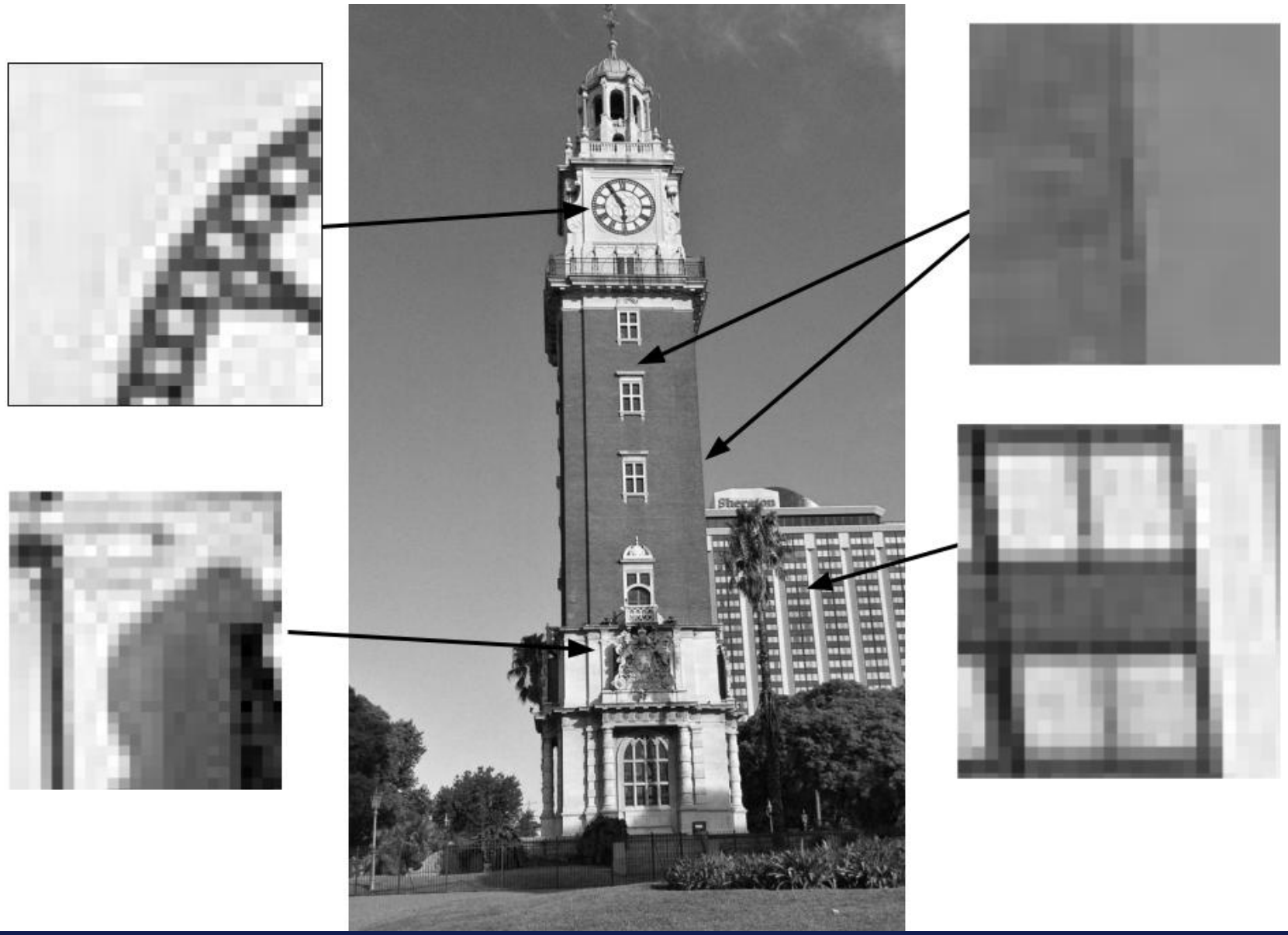
¿Cuál es el objetivo de detectar bordes?

- mapear una imagen 2D a un conjunto de curvas o segmentos de línea o contornos.
- IDEA: buscar segmentos definidos, líneas, curvas posterior al procesamiento.

Qué hace que en una imagen haya bordes?

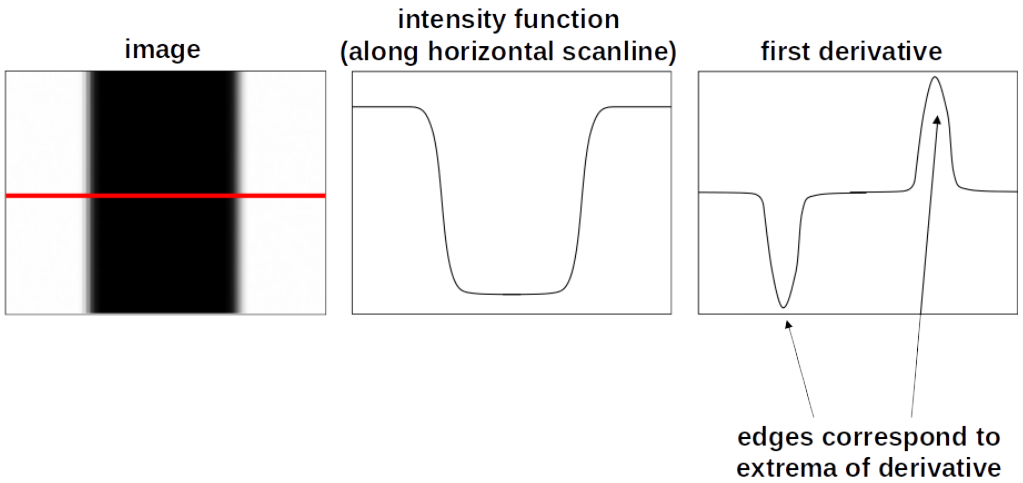


Dónde se distinguen bordes y dónde no?



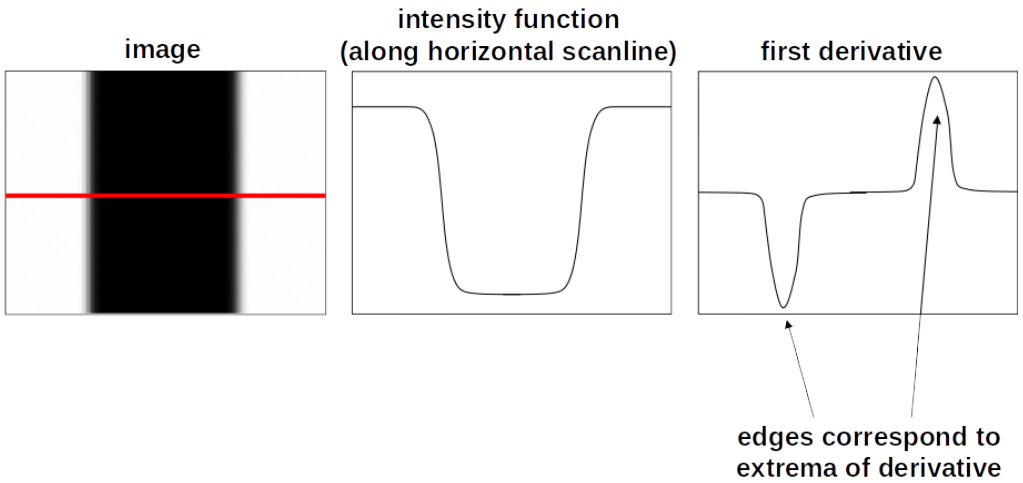
Derivadas y Bordes

- Un borde es el límite entre dos regiones con propiedades de nivel de gris relativamente distintas.
- La base de la mayoría de las técnicas de detección de bordes es el cálculo de un operador derivado local.



Derivadas y Bordes

- Un borde es el límite entre dos regiones con propiedades de nivel de gris relativamente distintas.
- La base de la mayoría de las técnicas de detección de bordes es el cálculo de un operador derivado local.
- La magnitud de la primera derivada calculada dentro de una vecindad alrededor del píxel de interés, se puede utilizar para detectar la presencia de un borde en una imagen.



Métodos de basados en el cálculo del gradiente

- Se trata de localizar los cambios abruptos en la función intensidad de la imagen.
- Los cambios en las funciones continuas se describen mediante las derivadas.
- En una imagen se considera el gradiente que describe la variación de la función.

Gradiente

- $$\nabla f = \begin{pmatrix} f_x \\ f_y \end{pmatrix} = \begin{pmatrix} f(x+1,y) - f(x,y) \\ f(x,y+1) - f(x,y) \end{pmatrix}$$

  donde $f_x = \frac{\partial f}{\partial x}$ y $f_y = \frac{\partial f}{\partial y}$
- *Magnitud* o intensidad del borde
- *Dirección* del borde

Magnitud y Dirección de ∇f

- *Magnitud*

$$\left|\nabla f\right| = \sqrt{f_x^2 + f_y^2}$$

- *Dirección*

$$\alpha = \arctan\left(\frac{f_x}{f_y}\right)$$

La magnitud y la dirección de f son dos imágenes del mismo tamaño que f.

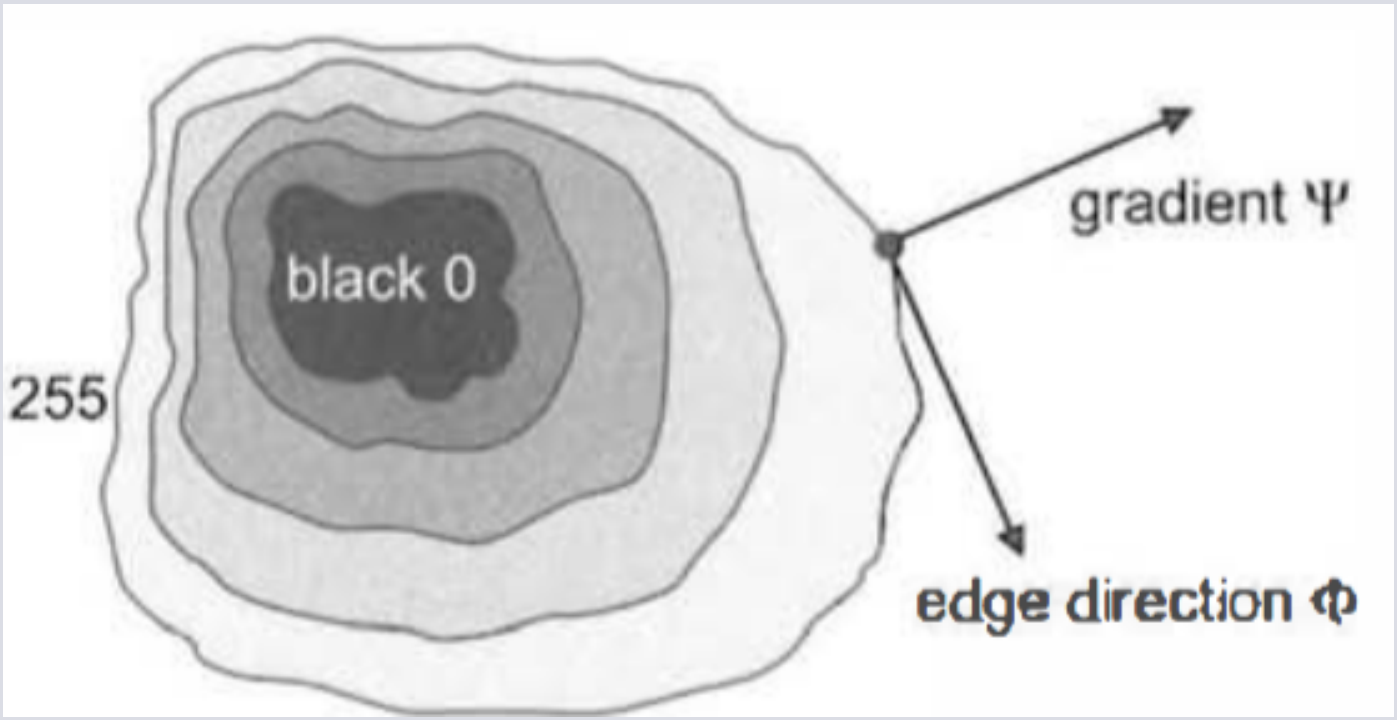
Detección del borde

Detectar un borde consiste en encontrar, para cada (x, y):

- $\left|\nabla f(x, y)\right|$,
- $\alpha(x, y)$, **ángulo** que forma el vector gradiente con el eje x

Dirección del borde

El gradiente es perpendicular a la dirección del borde



La dirección del gradiente es la dirección de máximo crecimiento de la función, o sea, es la dirección de máxima variación.
Los pixels con más alto valor de magnitud son puntos de borde.

Si consideramos una ventana de 2×2, alrededor de cada punto (i, j) de modo que:

$$\left(\begin{array}{c|c} f(i,j) & f(i,j+1) \\ \hline f(i+1,j) & f(i+1,j+1) \end{array} \right)$$

en este caso una manera de aproximar la derivada primera es mediante:

$$\begin{aligned} \frac{\partial f}{\partial x} &= f(i+1,j) - f(i,j) \\ \frac{\partial f}{\partial y} &= f(i,j+1) - f(i,j) \end{aligned}$$

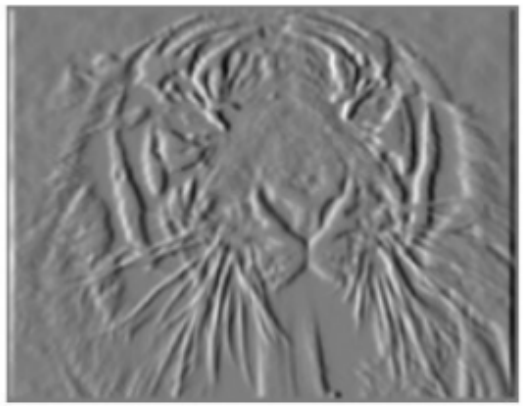
otra forma es haciendo diferencias cruzadas:

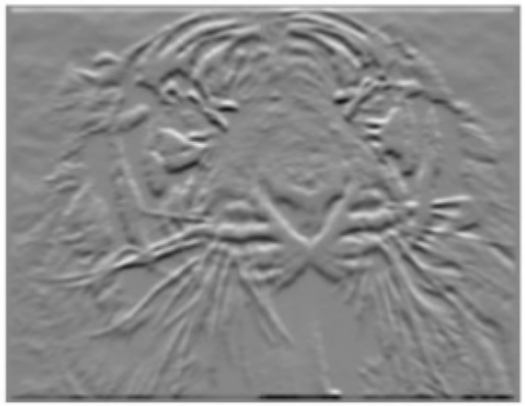
$$\begin{aligned} \frac{\partial f}{\partial x} &= f(i,j) - f(i+1,j+1) \\ \frac{\partial f}{\partial y} &= f(i,j+1) - f(i+1,j) \end{aligned}$$

Derivadas parciales de una imagen

$$\frac{\partial f(x,y)}{\partial x}$$

$$\frac{\partial f(x,y)}{\partial y}$$





-1	1

Cuál muestra cambios respecto del eje x?

-1
1

?
or

1
-1

Roberts

Original Image

$$h_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad h_2 = \begin{bmatrix} 1 & -1 \end{bmatrix}$$

$$h_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad h_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Si en cambio son de 3×3, alrededor del punto (i, j)

$$\left(\begin{array}{c|c|c} f(i-1,j-1) & f(i-1,j) & f(i-1,j+1) \\ \hline f(i,j-1) & f(i,j) & f(i,j+1) \\ \hline f(i+1,j-1) & f(i+1,j) & f(i+1,j+1) \end{array}\right)$$

que se puede aproximar:

$$f_x = f_{x_3} - f_{x_1} \qquad f_y = f_{y_3} - f_{y_1}$$

donde:

$f_{x_3} = f(i+1,j-1) + f(i+1,j) + f(i+1,j+1)$
$f_{x_1} = f(i-1,j-1) + f(i-1,j) + f(i-1,j+1)$
$f_{y_3} = f(i,j+1) + f(i,j+1) + f(i+1,j+1)$
$f_{y_1} = f(i-1,j-1) + f(i,j-1) + f(i+1,j-1)$
f_{x_k} son las filas y f_{y_k} son las columnas, con $k = 1, 2, 3$.

Operadores gradientes

Roberts

Utiliza una vecindad del pixel de 2×2, las máscaras de convolución son:

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

Es simple de calcular la magnitud del borde, pero tiene la desventaja de ser muy sensitivo al ruido, debido a usar muy pocos pixels para aproximar el gradiente.
La magnitud del gradiente se puede calcular como:

$$\left|f(i,j) - f(i+1,j+1)\right| + \left|f(i,j+1) - f(i+1,j)\right|$$

Operadores gradientes isotrópicos

Prewitt

$$\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

Sobel

$$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

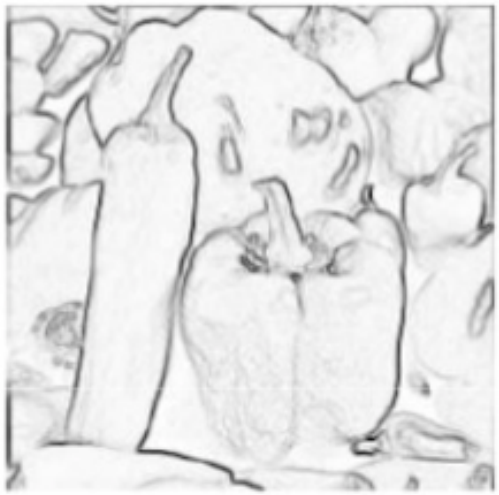
Isotrópico

$$\begin{pmatrix} -1 & 0 & 1 \\ -\sqrt{2} & 0 & \sqrt{2} \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{pmatrix}$$

Propiedades

- Los operadores Prewitt, Sobel e Isotrópico calculan las diferencias horizontales y verticales de las sumas en un vecindario del pixel.
- Esto reduce el efecto del ruido en la imagen, y tienen la propiedad de ser cero en las regiones de intensidad uniforme.

Ejemplo Prewitt-Sobel



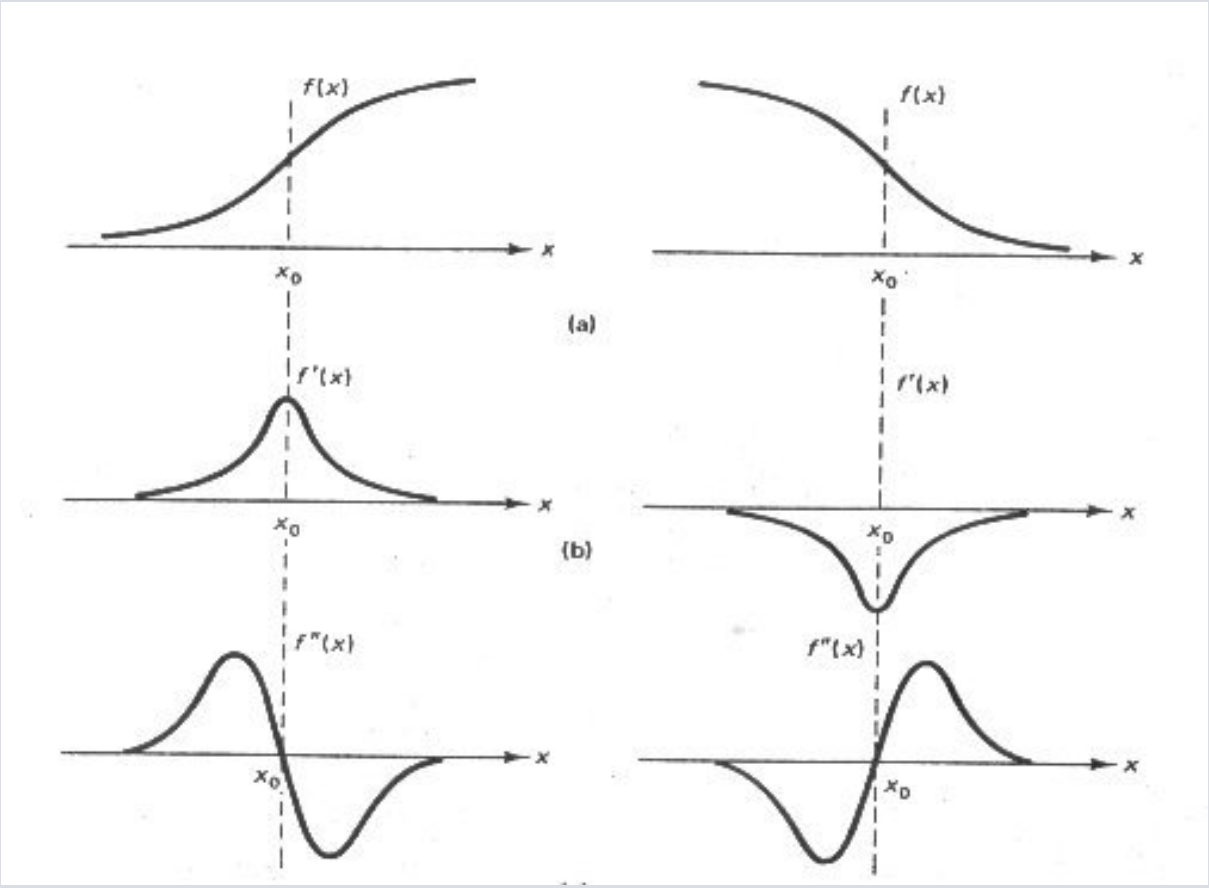
Prewitt

$$h_1 = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad h_2 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Sobel

$$h_1 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad h_2 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Métodos basados en la derivada segunda



Laplaciano

Si solamente nos interesa la magnitud (sin la orientación) definimos el operador lineal Laplaciano:

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

- tiene las mismas propiedades en todas las direcciones
- es invariante a rotaciones.
- se utiliza para el zero-crossing (cruce por cero de la derivada segunda)
- desventajas: más sensitivo al ruido y produce bordes dobles.

Aproximación derivada segunda

$$\nabla^2 f(i,j) = f_{xx}(i,j) + f_{yy}(i,j)$$

donde

$$f_x(i,j) = f(i+1,j) - f(i,j)$$

$$f_{xx}((i,j) = f_x((i,j)) - f_x(i-1,j)$$

$$f_{yy}((i,j) = f_y((i,j)) - f_y(i,j-1)$$

Luego,

$$\nabla^2 f(i,j) = -4f(i,j) + f(i-1,j) + f(i,j-1) + f(i,j+1) + f(i+1,j)$$

Máscara del Laplaciano

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

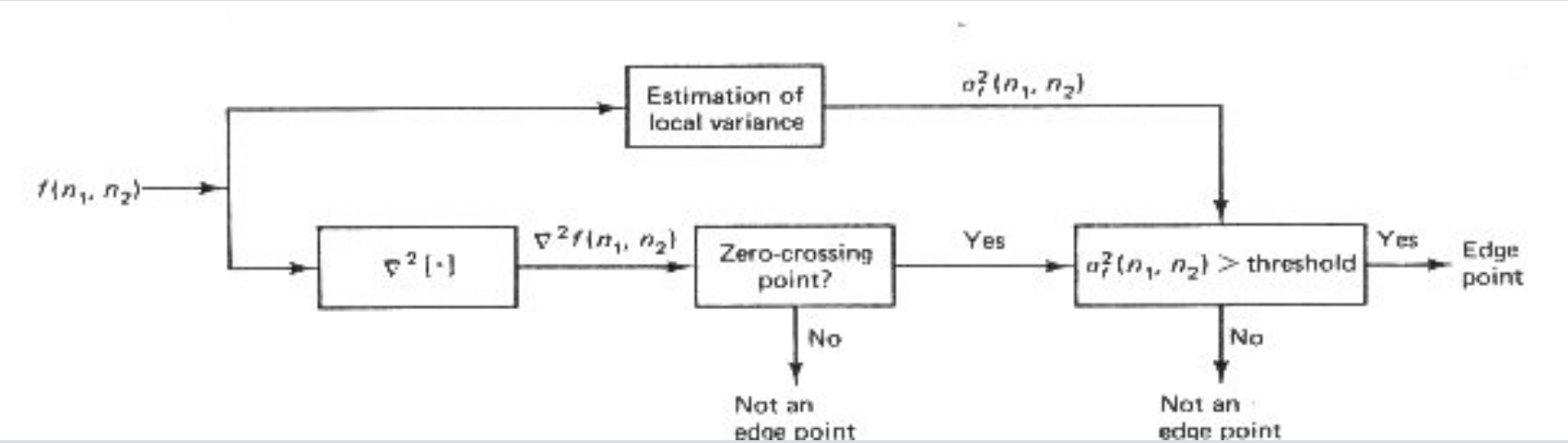
- La ubicación de los bordes se encuentra haciendo *zero crossing*.

Métodos basados en el Laplaciano

- Los resultados del Laplaciano generan muchos bordes falsos, en general en regiones donde la varianza local es pequeña.
- Zero crossing es donde aparecen cambios de signo, indice que hay un borde.
- Un método para remover falsos contornos consiste en chequear que la varianza local sea suficientemente grande en esos puntos.

Detección de Bordes basada en el Laplaciano

Sistema basado en el laplaciano, reduce la cantidad de bordes falsos.



Varianza local

La varianza local $\sigma_f^2(n_1, n_2)$ puede estimarse como:

$$\sigma_f^2(n_1, n_2) = \frac{1}{(2M+1)^2} \sum_{k_1=n_1-M}^{n_1+M} \sum_{k_2=n_2-M}^{n_2+M} [f(n_1, n_2) - m_f(k_1, k_2)]^2$$

donde

$$m_f(n_1, n_2) = \frac{1}{(2M+1)^2} \sum_{k_1=n_1-M}^{n_1+M} \sum_{k_2=n_2-M}^{n_2+M} f(k_1, k_2)$$

que es el valor medio, y M cercano a 2.

- Como $\sigma_f^2(i,j)$ se compara con un umbral, el factor de escala $\frac{1}{(2M+1)^2}$ puede suprimirse.
- La varianza local σ_f^2 se evalúa para aquellos (i,j) que son zero crossing para $\nabla^2 f(i,j)$
- En el esquema anterior se observa que la varianza está estrechamente relacionada con la magnitud del gradiente.
- Comparar $\sigma_f^2(i,j)$ con un umbral es similar a comparar el gradiente con un umbral.
- Si $\nabla^2 f(i,j) = 0$ existe la posibilidad de que allí haya un borde.
- El sistema evalúa si $\sigma_f^2(n_1,n_2)$ supera cierto umbral predeterminado, solamente en los puntos zero crossing de $\nabla^2 f(i,j)$.
- Para reducir el ruido en el cálculo de la segunda derivada, primero hay que suavizar la imagen.

Laplaciano del Gaussiano

LoG

Se suaviza la imagen con un filtro Gaussiano, se halla el Laplaciano y luego se calcula los puntos del cruce por cero.

$$\nabla^2(f(x,y) * h(x,y)) = f(x,y) * [\nabla^2 h(x,y)] =$$

$$f(x,y) * [\frac{\partial^2 h(x,y)}{\partial x^2} + \frac{\partial^2 h(x,y)}{\partial y^2}]$$

$$\nabla^2 h(x,y) = \frac{e^{-(x^2+y^2)/(2\pi\sigma^2)}}{(\pi\sigma^2)^2}(x^2+y^2-2\pi\sigma^2)$$

$$F[\nabla^2 h(x,y)] = -2\pi^2\sigma^2 e^{-\pi\sigma^2(\Omega_x^2+\Omega_y^2)/2}(\Omega_x^2+\Omega_y^2)$$

Detector de bordes Canny

Criterios de detección de Canny

- Criterio de Detección: no perder bordes y que no haya respuestas espúreas.
- Criterio de Localización: minimizar la distancia entre la posición real y el borde detectado.
- Criterio de una sola Respuesta: minimiza múltiples respuestas múltiples correspondientes a un único borde.

Algoritmo de Canny

- Obtención del gradiente: obtener magnitud y orientación del vector gradiente en cada pixel.
- Supresión de no máximos: Conseguir el adelgazamiento del ancho de los bordes, obtenidos con el gradiente, hasta lograr bordes de un pixel de ancho.
- Histéresis de umbral: Aplicar esta función de histéresis que está basada en dos umbrales; se quiere reducir la posibilidad de bordes falsos.

Obtención del gradiente

Aplicar un filtro Gaussiano para eliminar ruido y suavizar la imagen gradiente en cada pixel.

(a)

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

(b)

$$\frac{1}{115}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

Máscaras Gaussianas sugeridas para suavizar que se obtiene promediando los valores de intensidad de los pixels en el entorno de vecindad con una máscara de convolución con un cierto σ.

Algoritmo para obtener el gradiente

Entrada: I imagen, máscara de convolución H.
Salida: I_m magnitud del gradiente, I_o orientación del gradiente
1. Suavizar la imagen I mediante el filtro Gaussiano H y obtener la imagen de salida J.
2. Para cada pixel (i, j) en J, obtener la magnitud y orientación del gradiente.
3. Obtener I_m e I_o

Supresión no máximos del gradiente

- Entradas I_m e I_o.
- Las orientaciones son: $0°$, $45°$,$90°$, y $135°$ con respecto al eje horizontal.
- Para cada pixel se encuentra la dirección que mejor aproxime a la dirección del ángulo de gradiente.
- Si el valor de la magnitud de gradiente es menor que al menos uno de sus dos vecinos en la dirección del ángulo obtenida en el paso anterior, entonces se asigna el valor 0 a dicho pixel, sino se asigna el valor que tenga la magnitud del gradiente.
- La salida de este segundo paso es la imagen I_n de bordes, es decir, $I_m(i, j)$, después de la supresión no máxima de puntos de borde.

Algoritmo: Supresión no máximos

Entrada: I_m e I_o
Salida: I_n
Dadas cuatro direcciones d_1, d_2, d_3 y d_4 que representan a $0°$, $45°$,$90°$, y $135°$ con respecto al eje horizontal.

- 1. Para cada (i, j) :
  - 1.1. Encontrar la dirección d_k que mejor se aproxima a la dirección $I_o(i, j)$, o sea la perpendicular al borde.
  - 1.2. Si $I_m(i, j)$ es menor a al menos uno de sus dos vecinos en la dirección d_k , al pixel (i, j) de I_n se le asigna el valor 0 o sea $I_n(i, j) = 0$ (supresión), sino $I_n(i, j) = I_m(i, j)$.
- 2. Devolver I_n

3. Umbral por histéresis

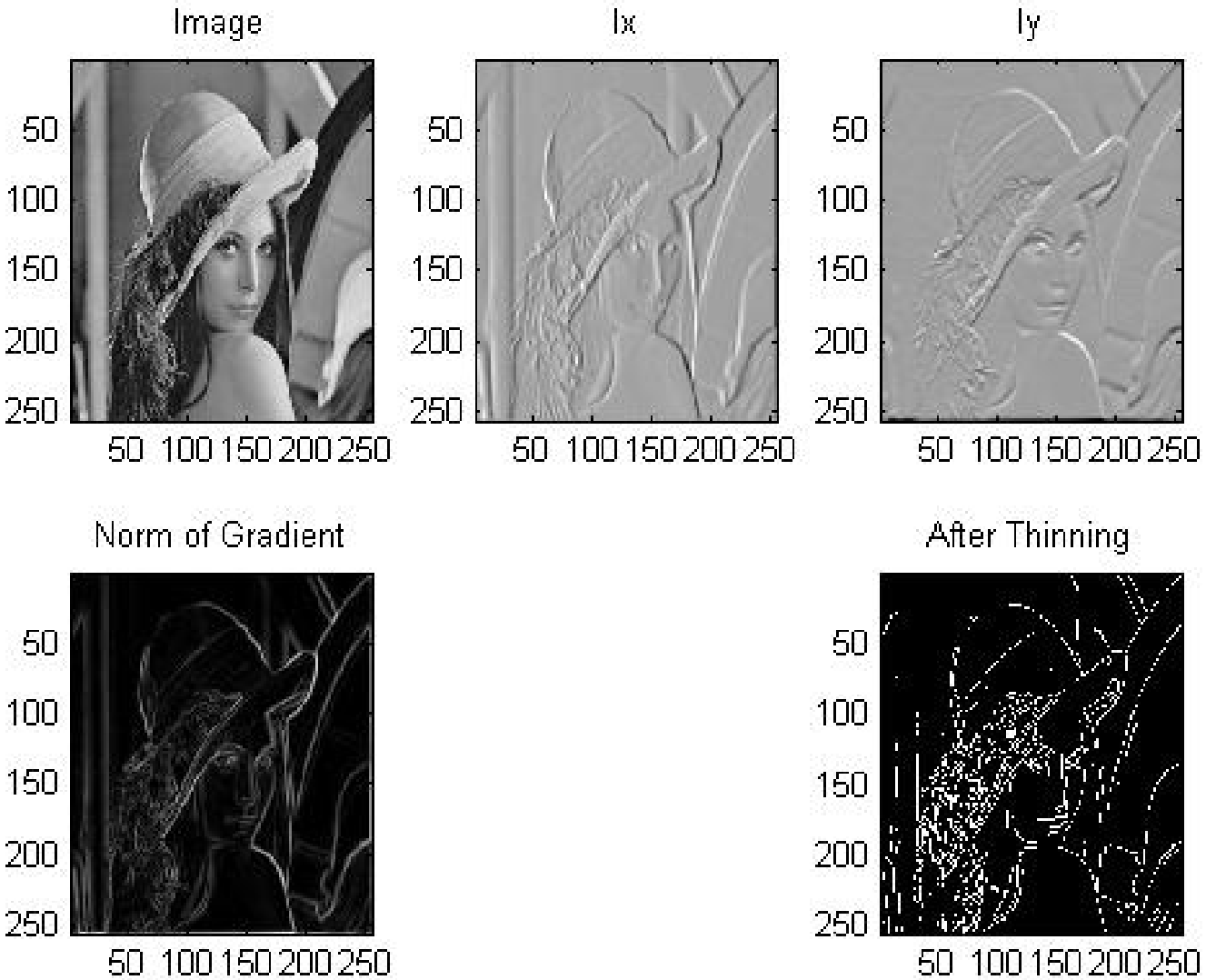
I_n suele contener máximos locales creados por el ruido. Para dar solución y eliminar dicho ruido se calcula el umbral por histéresis:

- considerar la imagen obtenida del paso anterior.
- considerar la matriz de orientaciones de los puntos de borde de la imagen.
- tomar dos umbrales U_{min} y U_{max}.
- para cada pixel de la imagen, si el valor de la magnitud del pixel considerado es mayor a U_{max}, entonces ese pixel es marcado como borde.
- se considera el siguiente pixel vecino no explorado que está en la dirección perpendicular al gradiente y que sea mayor a U_{min} y se lo marca como borde.
- a partir de dicho pixel seguir las cadenas de máximos locales conectados en ambas direcciones perpendiculares a la normal del borde, siempre que sean mayores a U_{min}.
- se marcan todos los pixels explorados y se almacena la lista de todos los pixels en el contorno conectado. Es así como en este paso se logra eliminar las uniones en forma de Y de los segmentos que confluyan en un pixel.

Cerrar los contornos

- consiste en cerrar los contornos que pudiesen haber quedado abiertos por problemas de ruido.
- método muy utilizado es el algoritmo de Deriche y Cocquerez. Este algoritmo utiliza como entrada una imagen binarizada de contornos de un pixel de ancho.
- el algoritmo busca los extremos de los contornos abiertos y sigue la dirección del máximo gradiente hasta cerrarlos con otro extremo abierto.

Ejemplos de Canny



Material

Links

- http://links.uwaterloo.ca/Repository.html
-