

# Filtrado no lineal

Visión por Computador, curso 2024-2025

---

Silvia Martín Suazo, silvia.martin@u-tad.com

3 de octubre de 2024

U-tad | Centro Universitario de Tecnología y Arte Digital



## Filtrado lineal y no lineal

Los distintos algoritmos de **filtrado** vistos hasta ahora se conocen como **filtrados lineales**. Esto se debe a que para calcular la transformación se realiza una **combinación lineal** de los valores de los píxeles del vecindario.

Sin embargo, también existen otro tipo de transformaciones cuyo cálculo no se realiza mediante una combinación lineal.

## Filtrado lineal y no lineal

Los **filtros lineales** se utilizan para suavizado(reducción de nitidez o efecto difuminado) uniforme con el fin de reducir el ruido. Se pueden perder detalles.

Por el contrario, los **filtros no-lineales** suavizan la imagen, pero se preservan los detalles y bordes. Por lo tanto, se consideran más eficaces dependiendo del tipo de ruido.

# Notebook de ejemplos de filtros

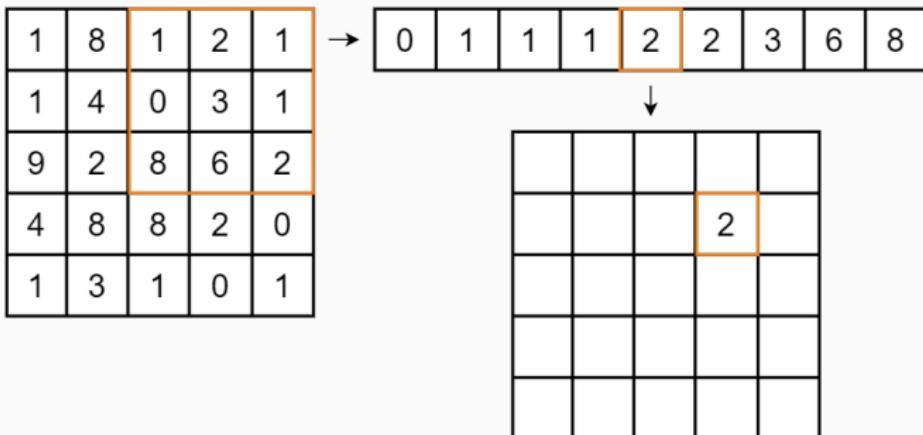
Los filtros que se explicarán a continuación pueden ser observados de manera práctica en el siguiente notebook.



- 02.05-Filtrado no-lin.ipynb

# Filtro de la mediana

El **filtro de la mediana** se basa en calcular la mediana de los píxeles del vecindario de cada posición de la imagen.



El kernel debe tener dimensión **ímpar**.

## Filtro de la mediana

Este filtro es especialmente útil para la eliminación de **ruido impulsivo** en las imágenes. Al evitar que los píxeles ruidosos contribuyan al **promediado**.



[1]

# Filtro de la mediana

Imagen Original



Filtro de la mediana



Kernel de 31x31

## Filtro de la mediana híbrido

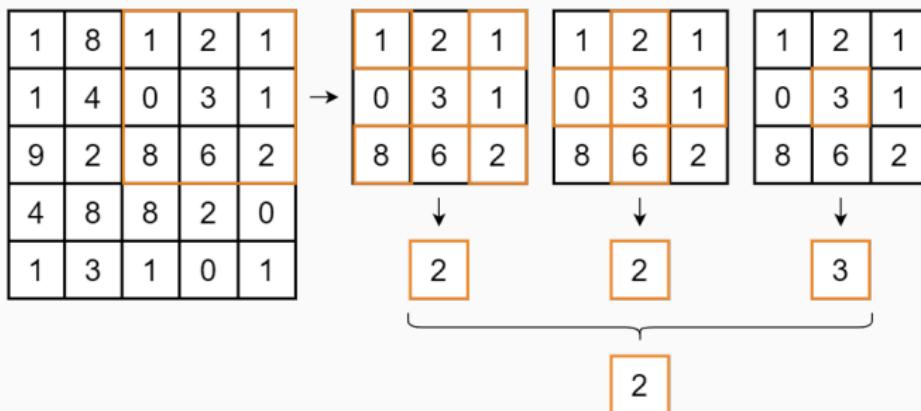
El algoritmo para aplicar el **filtro de la mediana híbrido** es el siguiente:

1. Se calcula la mediana de la ventana en **forma de cruz**.
2. Se calcula la mediana de la ventana en **forma de x**.
3. Se toma el **píxel central**.
4. La salida es la **mediana** de los pasos 1, 2 y 3.

# Filtro de la mediana híbrido

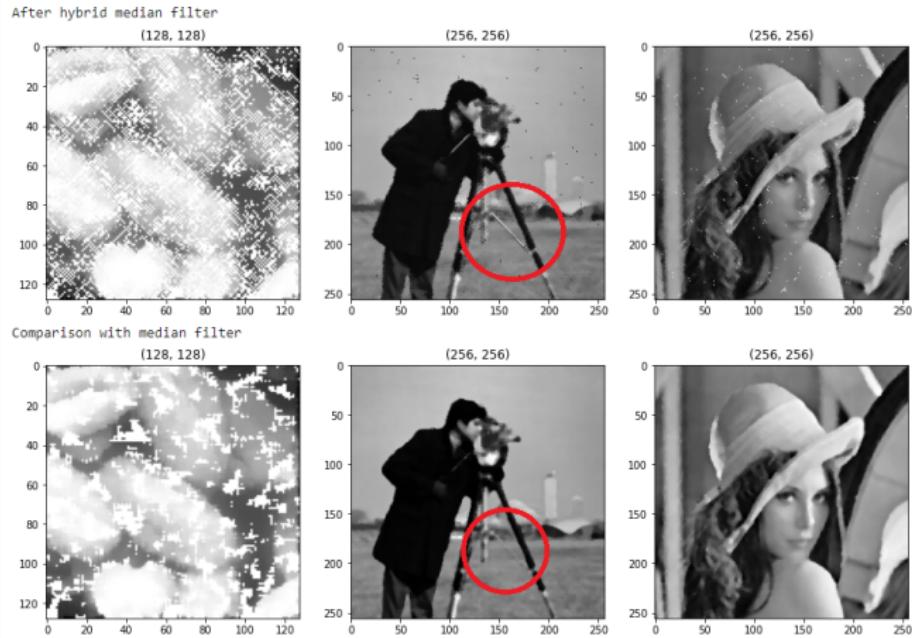
Uno de los principales **problemas** que se encontraban en el filtro de la media es el tratamiento que este hace de los **bordes**.

La mediana de la ventana de píxeles cercanos trata de manera igual a todos los píxeles. El **filtro de la mediana híbrido** tiene como objetivo presentar mejores características a la hora de tratar las esquinas de la ventana.



# Filtro de la mediana híbrido

Con la aplicación de este filtro, se consiguen recuperar de manera **más nítida** ciertas estructuras de la imagen original.



# Filtro de la mediana híbrido

Imagen Original



Filtro de la mediana híbrido

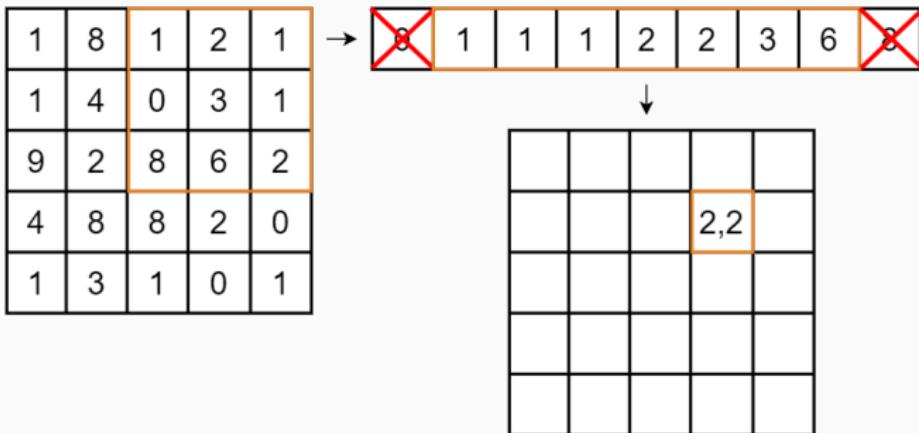


Kernel de 31x31

## Filtro de la media truncada

El **filtro de la media truncada** mejora el comportamiento del filtro de la mediana ante ruido gaussiano.

Esto se debe al realizar un **promedio** de los píxeles cercanos, pero sin tener en cuenta posibles **outliers**.



## Filtro de la media truncada

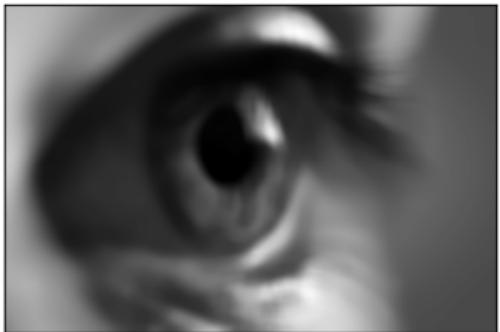
Este filtro es un híbrido entre el **filtro de tipo caja** y el **filtro de la mediana**, ya que promedia los valores del vecindario de un píxel pero al mismo tiempo elimina posibles **valores extremos**, eliminando así el **ruido espontáneo**.

# Filtro de la media truncada

Imagen Original



Filtro de la media truncada



Kernel de 31x31,  $\alpha = 0.3(30\%)$

## Filtrado bilateral

El **filtrado bilateral**[2] es un filtro de suavizado de imágenes cuyo objetivo es la **eliminación de ruido** preservando todo lo posible los **bordes de las estructuras**. El cálculo del filtro se realiza a través de la siguiente ecuación:

$$I^{\text{filtered}}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|) \quad (1)$$

donde:

$$W_p = \sum_{x_i \in \Omega} f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|) \quad (2)$$

## Filtrado bilateral

$$I^{\text{filtered}}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|) \quad (3)$$

donde:

$$W_p = \sum_{x_i \in \Omega} f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|) \quad (4)$$

- $I^{\text{filtered}}$  es la imagen filtrada;
- $I$  es la imagen de entrada original que se va a filtrar;
- $x$  son las coordenadas del píxel actual que se filtrará;
- $\Omega$  Es la ventana centrada en  $x$ , entonces  $x_i \in \Omega$  es otro píxel;
- $f_r$  es el kernel para suavizar las diferencias en intensidades (esta función puede ser una función gaussiana );
- $g_s$  es el núcleo espacial (o de dominio) para suavizar las diferencias en las coordenadas (esta función puede ser una función gaussiana).

## Filtrado bilateral

Existen dos problemas derivados del uso de filtrado bilateral:

- **Efecto escalera**: mesetas de intensidad que hacen que las imágenes parezcan dibujos animados.
- Inversión de degradado

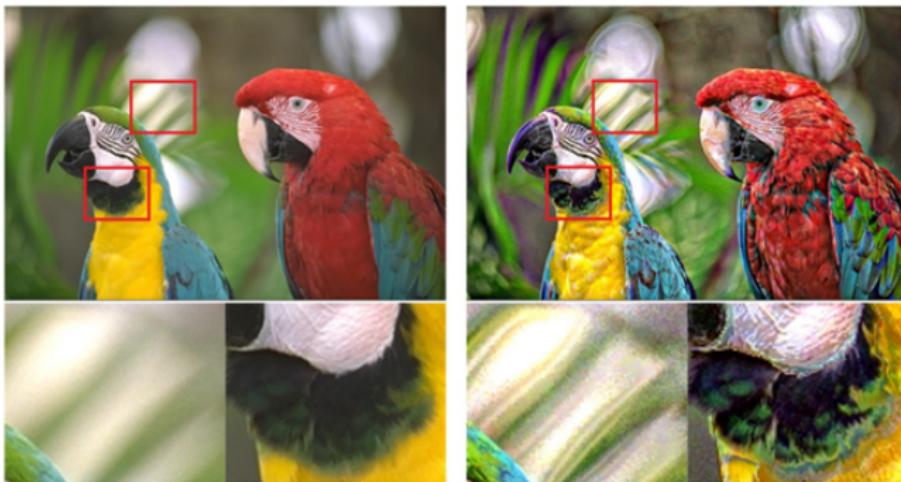


[3]

# Filtrado bilateral

Existen dos problemas derivados del uso de filtrado bilateral:

- Efecto escalera
- Inversión de degradado[4]: introducción de bordes falsos en la imagen.



[5]

# Filtro bilateral

Imagen Original



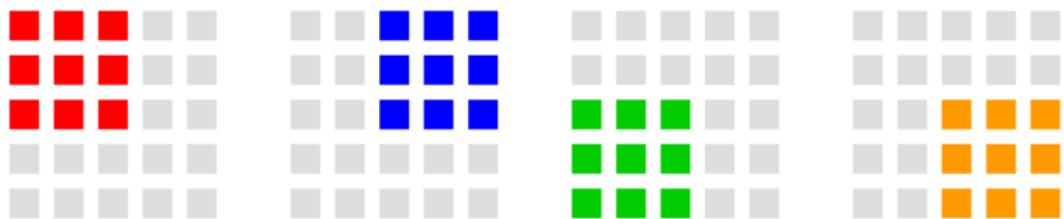
Filtro bilateral



Kernel de 31x31,  $\sigma = 75$

# Filtro de Kuwahara

Consiste en el desplazamiento de una ventana sobre la imagen.  
Dicha ventana se divide en **4 regiones**.



# Filtro de Kuwahara

1. Se toma una ventana de tamaño  $n \times n$ .
2. Dicha ventana se divide en 4 regiones de mismo tamaño a partir del píxel central.
3. Se calcula la **media y la varianza** de cada región.
4. Se toma la media de los valores de la región con menor varianza.
5. Dicha media es el nuevo valor del píxel central.

# Filtro de Kuwahara

Imagen Original



Filtro de Kuwahara



Resultado con filtro 31x31

# Filtrado Rolling-Guidance

El **filtro Rolling-Guidance**[6] tiene como objetivo la **eliminación de ruido** preservando las **estructuras importantes** de las imágenes.

Los autores del filtrado Rolling-Guidance reivindican que, en comparación con el filtrado bilateral, su propuesta consigue mejores resultados.



Source



Bilateral filter  
[6]



Rolling-Guidance filter

# Filtrado Rolling-Guidance

Los resultados mostrados consiguen **separar las estructuras de las imágenes a distintas escalas**.

Debido a la **sencilla** implementación del algoritmo, el filtrado se puede realizar en tiempo real.



Source



Bilateral filter  
[6]



Rolling-Guidance filter

# Filtrado Rolling-Guidance

Uno de los principales problemas observados con el **filtrado Gaussiano** es que las imágenes producidas están **difuminadas**, perdiendo así las estructuras importantes de la imagen.

Para evitar esto, el filtrado **Rolling-Guidance** se centra en recuperar los bordes perdidos tras aplicar un filtro Gaussiano a una imagen.

De esta manera se tiene un proceso **iterativo** basado en **dos pasos** diferenciados.

# Filtrado Rolling-Guidance

- **Paso 1: Eliminación de estructuras pequeñas**

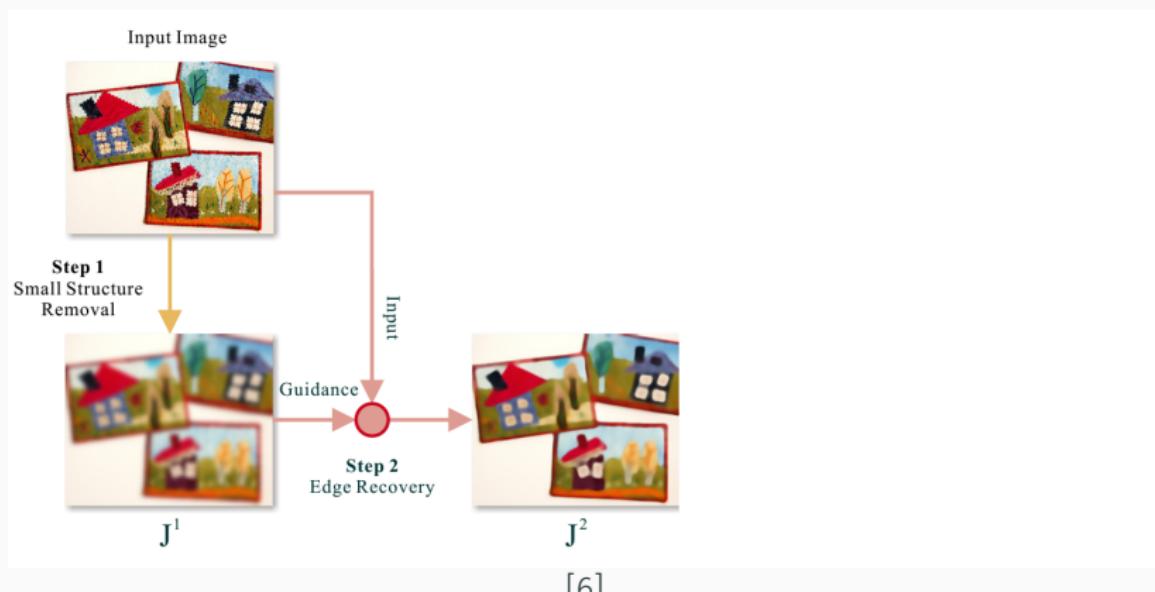
Para eliminar el posible **ruido** y **pequeñas estructuras** de las imágenes de entrada, primero se aplica un filtro Gaussiano a las imágenes.



# Filtrado Rolling-Guidance

- **Paso 2: Recuperación de bordes**

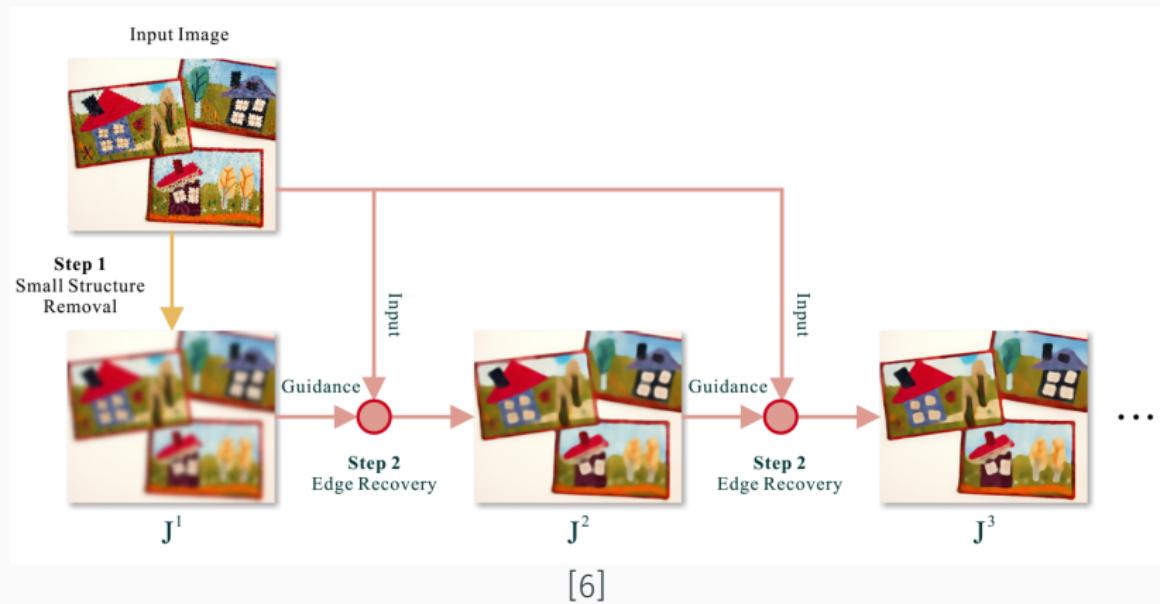
Durante este paso, se toma la imagen de entrada y se le añade la imagen obtenida en el paso anterior.



[6]

# Filtrado Rolling-Guidance

Este proceso se repite tanto como se desee durante un proceso **iterativo**.



## Detector de bordes de Canny

El **detector de bordes de Canny**[7] es un algoritmo para resaltar los bordes de una imagen.



[8]

# Detector de bordes de Canny

Este algoritmo está dividido en **5 pasos distintos**:

1. Reducción de ruido
2. Cálculo del gradiente
3. Non-maximum suppression
4. Doble umbralización
5. Resaltar bordes con Hysteresis

## 1. Reducción de ruido

Para eliminar el ruido de la imagen, se aplica un **filtrado gaussiano** de tamaño variable dependiendo de las dimensiones de la imagen (normalmente de 5x5).



[8]

## 2. Cálculo del gradiente

Una vez eliminado el ruido, se resaltan las estructuras más importantes haciendo uso del **filtrado de Sobel**.



[8]

## 2. Cálculo del gradiente

El filtrado de Sobel se realiza sobre el **eje X** y el **eje Y**. El resultado de ambos filtrados se usa para obtener el gradiente de la imagen a través de la siguiente ecuación:

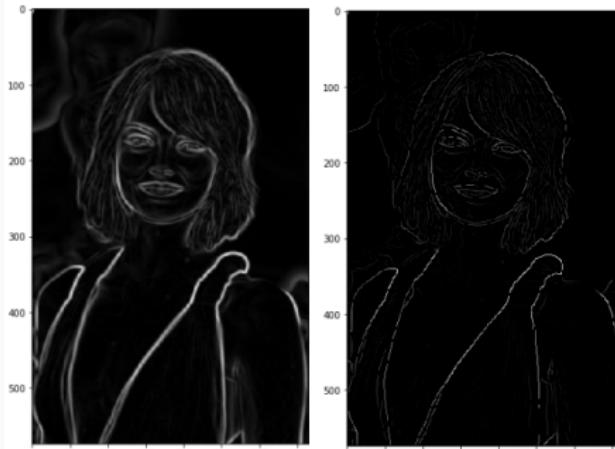
$$|G| = \sqrt{I_x^2 + I_y^2} \quad (5)$$

$$\theta(x, y) = \arctan \left( \frac{I_y}{I_x} \right) \quad (6)$$

## 3. Non-maximum suppression

Uno de los principales problemas del paso anterior era que los bordes que aparecen **no son uniformes**.

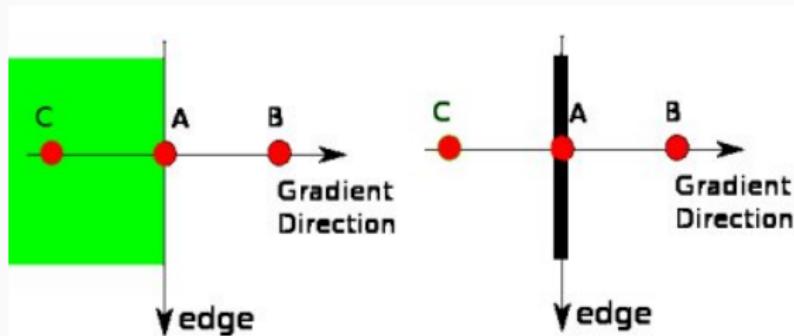
Para crear líneas uniformes se **eliminan los píxeles** que no pertenezcan a un borde.



## 3. Non-maximum suppression

Se eliminan los píxeles que no pertenecen a un borde, a través de detectar qué píxeles **suponen** un máximo en la dirección de su gradiente.

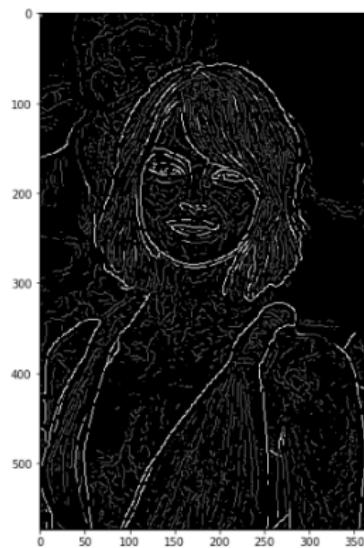
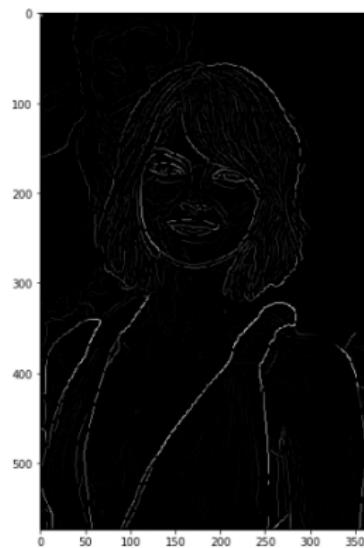
La dirección del gradiente es la **normal** al borde, para calcular el máximo se **compara** cada píxel con sus vecinos en esa dirección.



[9]

## 4. Doble umbralización

Durante este paso se **resaltan** los bordes de la imagen, identificando qué píxeles son los que **pertenecen** a dichos bordes.



[8]

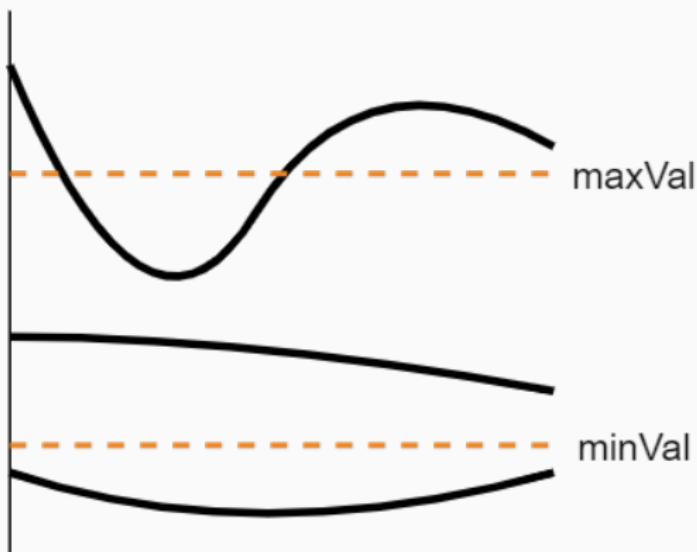
## 4. Doble umbralización

Se distinguen 3 tipos de píxeles:

- Píxeles *fuertes*: Pertenece a un borde debido a tener una intensidad elevada.
- Píxeles *débiles*: Su intensidad no es suficientemente elevada para pertenecer a un borde, pero tampoco lo suficientemente débil como para no pertenecer.
- Píxeles *no relevantes*: No tienen intensidad suficiente como para pertenecer al borde.

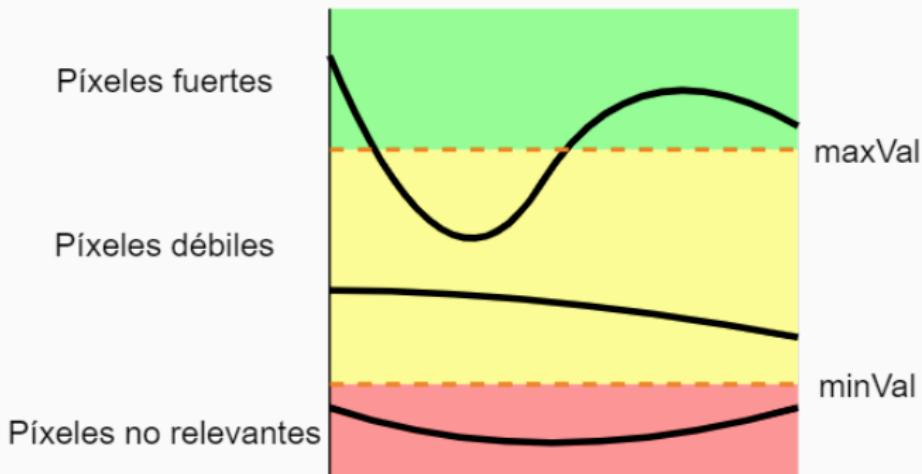
## 4. Doble umbralización

Se definen los límites superior e inferior de intensidad para realizar la clasificación de píxeles.



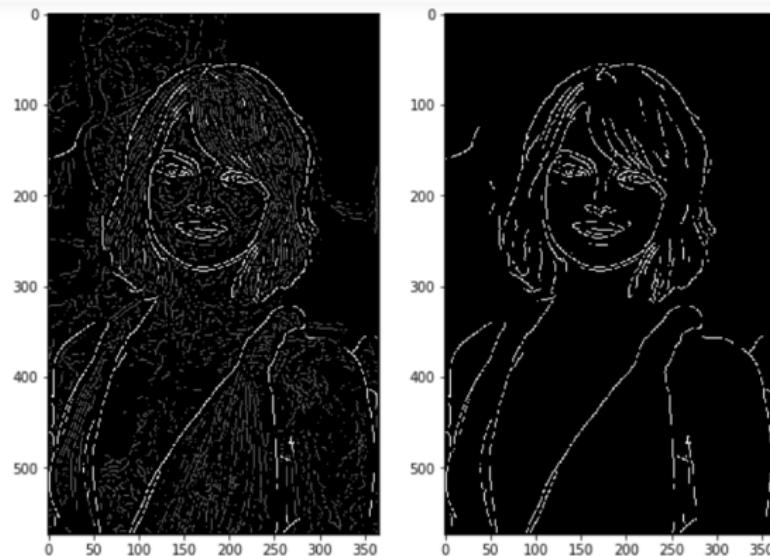
## 4. Doble umbralización

Se definen los límites superior e inferior de intensidad para realizar la clasificación de píxeles.



## 5. Resaltar bordes con Hysteresis

Por último, se realiza una **clasificación final** de qué píxeles pertenecen a un borde y cuáles no.

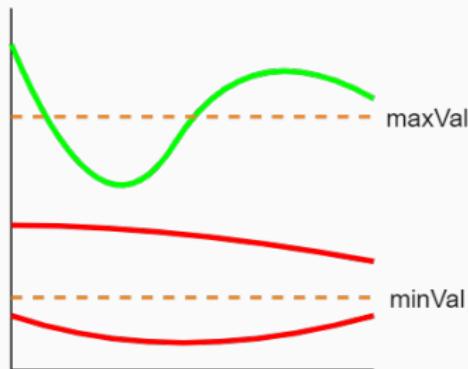


[8]

## 5. Resaltar bordes con Hysteresis

Los píxeles **débiles** detectados en el paso anterior se identifican como **fuertes** o no relevantes:

- Si **cerca del píxel débil** existe algún **píxel fuerte**, este se clasifica como **fuerte**.
- Por otro lado, si **no hay píxeles fuertes** alrededor, este se clasifica como **no relevante**.

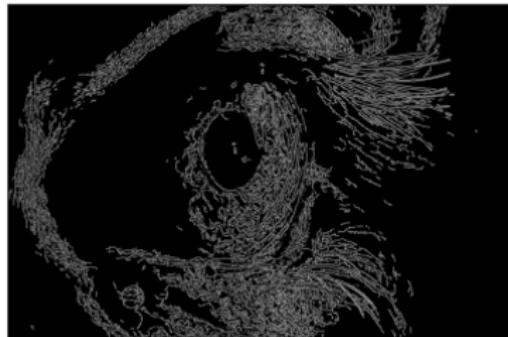


# Detector de bordes de Canny

Imagen Original



Detector de bordes de Canny



Kernel con min 10 y max 20

# Referencias i

---

- [1] mas\_bejo (Stack Overflow).  
**Median filter image.**  
[Online; accessed August, 2022].
- [2] Carlo Tomasi and Roberto Manduchi.  
**Bilateral filtering for gray and color images.**  
In *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*, pages 839–846. IEEE, 1998.
- [3] Antoni Buades, Bartomeu Coll, and J-M Morel.  
**The staircasing effect in neighborhood filters and its solution.**  
*IEEE transactions on Image Processing*, 15(6):1499–1505, 2006.

## Referencias ii

- [4] Kaiming He, Jian Sun, and Xiaoou Tang.  
**Guided image filtering.**  
*IEEE transactions on pattern analysis and machine intelligence*,  
35(6):1397–1409, 2012.
- [5] Wei Liu, Wei Xu, Xiaogang Chen, Xiaolin Huang, Chunhua Shen,  
and Jie Yang.  
**Edge-preserving piecewise linear image smoothing using  
piecewise constant filters.**  
*arXiv preprint arXiv:1801.06928*, 2018.
- [6] Qi Zhang, Xiaoyong Shen, Li Xu, and Jiaya Jia.  
**Rolling guidance filter.**  
In *European conference on computer vision*, pages 815–830.  
Springer, 2014.

## Referencias iii

[7] John Canny.

**A computational approach to edge detection.**

*IEEE Transactions on pattern analysis and machine intelligence*,  
(6):679–698, 1986.

[8] Sofiane Sahir (Towards Data Science).

**Canny edge detection results images.**

[Online; accessed August, 2022].

[9] OpenCV.

**Canny edge detection scheme images.**

[Online; accessed August, 2022].