



**Universidad de Guanajuato**  
**DIVISIÓN DE CIENCIAS E INGENIERÍAS**  
**CAMPUS LEÓN**

**Fundamentos de Procesamiento Digital de Imágenes**

Grupo:A

**Tarea 8 - Filtro High-Boost y Suavizado de  
Imágenes  
Unidad 1**

**FECHA DE ENTREGA: 14 DE OCTUBRE DE 2022**

**Alumno:**

Miguel Ángel Hernández Tapia

**Profesor:**  
Dr. Arturo González Vega

# Índice

<b>1. Introducción</b>	<b>3</b>
<b>2. Metodología</b>	<b>5</b>
<b>3. Resultados</b>	<b>5</b>
<b>4. Discusiones</b>	<b>6</b>
<b>5. Conclusión</b>	<b>7</b>
<b>A. Código</b>	<b>8</b>

# Fundamentos de Procesamiento Digital de Imágenes

## Tarea 8 - Filtro High-Boost y Suavizado de Imágenes

Miguel Ángel Hernández Tapia

14 de octubre de 2022

### Resumen

En este reporte se obtuvieron los parámetros para visualizar mejor un tumor en una mastografía dados 2 filtros High-Boost con kernels Gaussiano y Laplaciano. También se comparan 4 métodos para suavizar imágenes y mejorar la nitidez eliminando el ruido.

## 1. Introducción

Restando una suavizada versión de una imagen con la original es un proceso que ha sido usado desde 1930 para la industria de la imprenta y publicación para afinar imágenes. Este proceso se llama **unsharpen mask** o máscara sin afinar.

Este proceso consiste en

1. Suavizar la imagen original  $f(x, y)$ .
2. Crear una máscara restando la imagen original con la suavizada.

$$g_{mask}(x, y) = f(x, y) - \bar{f}(x, y)$$

3. Añadir la máscara al original con un escalamiento como

$$g(x, y) = f(x, y) - kg_{mask}(x, y)$$

El **suavizado**, también llamado **desenfoque**, es una operación de procesamiento de imágenes simple y de uso frecuente. Hay muchas razones para suavizar. En este documento nos centraremos en suavizar para reducir el ruido

**Filtro de caja normalizado.** Este filtro es el más simple de todos! Cada píxel de salida es la media de sus vecinos del núcleo (todos ellos contribuyen con pesos iguales). Su kernel es

$$\text{kernel} = \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Probablemente el filtro más útil (aunque no el más rápido). **El filtrado gaussiano** se realiza haciendo girar cada punto de la matriz de entrada con un núcleo gaussiano y luego sumándolos todos para producir la matriz de salida. Recordando que un gaussiano 2D se puede representar como:

$$G_0(x, y) = Ae^{-\frac{(x_x)^2}{2^2_x} + \frac{(y_y)^2}{2^2_y}}$$

**El filtro mediano** recorre cada elemento de la señal (en este caso la imagen) y reemplaza cada píxel con la mediana de sus píxeles vecinos (ubicados en un vecindario cuadrado alrededor del píxel evaluado).

A veces los filtros no solo disuelven el ruido, sino que también suavizan los bordes. Para evitar esto (al menos en cierta medida), podemos usar un **filtro bilateral**. [1][3][2]

## 2. Metodología

Este trabajo se desarrolló en Google Colab. En este ambiente se puede compilar el lenguaje Python con una basta cantidad de librerías ya incluidas sin necesidad de instalarlas. Se utilizaron las librerías cv2 (OpenCV), numpy y matplotlib.pyplot. La primera sirve para manejo de imágenes como objetos que tiene diversos métodos y funciones definidas. Numpy es una librería optimizada para operaciones de arreglos (vectores, matrices, etc.), muy útil para reducir y optimizar las operaciones con sus métodos. Por último, matplotlib.pyplot es una función para graficar, funciona muy similar al plot de Matlab.

Los pasos son:

1. Realice una función que se llame *IGBoost*, que reciba una imagen, el tamaño de kernel, la sigma y un parámetro de combinación, con estos datos la función debe realizar el **filtrado highboost** con el proceso de **unsharpen mask**.
2. Realice una función (*IGLaplacian*) que reciba una imagen y un parámetro de combinación, con estos dos datos la función hará un reforzamiento de bordes usando reforzamiento por **Laplaciano**
3. Aplique las dos funciones anteriores a la imagen *Fig0303(a).tif*, encuentre los parámetros apropiados que permita que en la imagen se refuerce el tumor (zona hiperintensa de la mamografía). Reporte valores e imágenes tratadas.
4. A la imagen *Fig0319Noise.tif* aplíquele **filtros de caja** y **gaussianos** con distintos parámetros para eliminar el ruido lo más posible y lograr que la imagen recuperada se parezca la imagen *Fig0319(a).tif*. Muestre los resultados obtenidos y los parámetros utilizados. Ahora haga un **filtrado de mediana** cambiando parámetros, reporte los valores y el resultado de este filtrado

## 3. Resultados

1.- La función *IGBoost* es:

```
# HIGH-BOOST FILTER WITH GAUSSIAN KERNEL
def IGBoost(img, n, sigma, c):

    kernel = cv.getGaussianKernel(n, sigma)          # GAUSSIAN VECTOR
    window = np.outer(kernel, kernel.transpose())      # GAUSSIAN MATRIX
    img_fil = cv.filter2D(img, -1, window)

    mask = img - img_fil
    img_highboost = img + c*mask           # ADD PARAMETER WITH MASK

    return img_highboost
```

2.- La función *IGLaplacian* es:

```
# HIGH-BOOST FILTER WITH LAPLACIAN KERNEL
def IGLaplacian(img, alpha, c):

    kernel = Laplacian_kernel(3, alpha) # KERNEL OF 3X3
    mask = cv.filter2D(img, -1, kernel)
    img_fill = img + c*mask

    return img_fill
```

3.- Las imágenes resultantes de estas 2 funciones y la original se muestra en la Figura 1. Los parámetros utilizados en *IGBoost* fueron un kernel de  $25 \times 25$ , con  $\sigma = 5$  y  $k = 4$ . Y los parámetros para *IGLaplacian* fueron un  $\alpha = 0$  y  $k = -1$ .

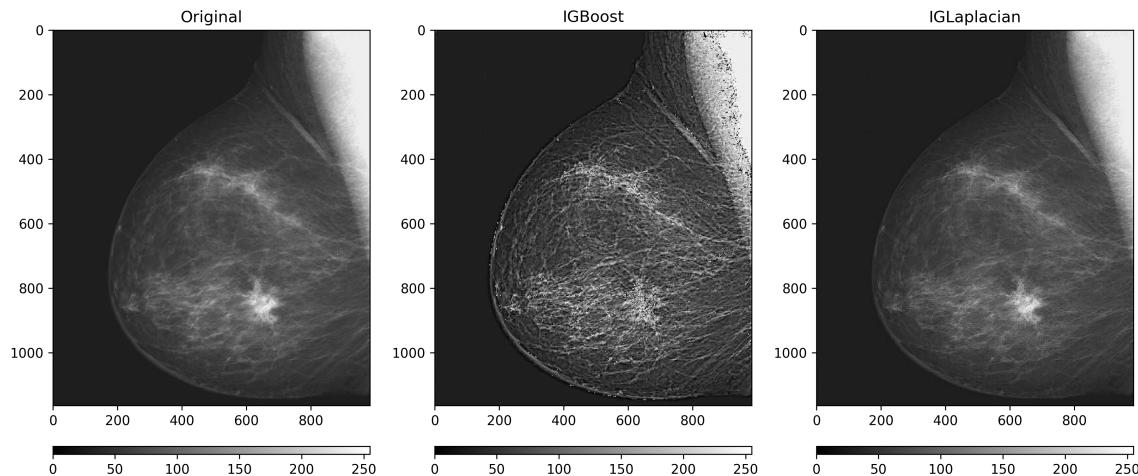


Figura 1: Comparación de resultados entre las funciones.

4.- Los resultados del suavizado con los kernels de caja y gaussianos se muestran en la Figura 2. Los resultados comparados con los kernels de filtro mediano y bilateral se muestran en la Figura 3

## 4. Discusiones

3.- La función *IGBoost* preserva los bordes de la imagen original y resalta más las líneas internas en la mastografía. La función *IGLaplacian* también conserva los bordes, elimina algunas partes con ruido y hace ligeramente más notorio el tumor. Con la primera función, casi no se distingue el tumor, ya que se resta el relleno y solo se distinguen los bordes. La mejor en estos casos fue el *IGLaplacian* aunque podría variar si se encuentran mejores parámetros para cada uno.

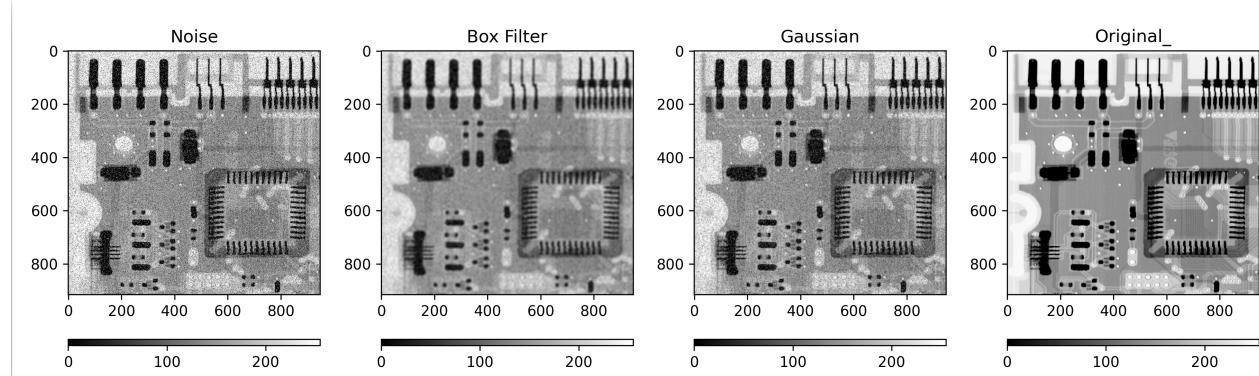


Figura 2: Comparaciones Filtro Caja y Gaussiano.

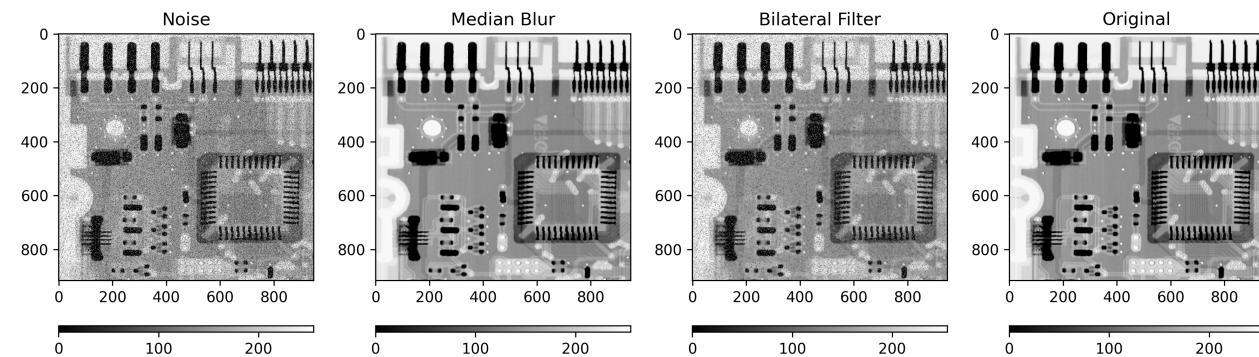


Figura 3: Comparaciones Filtro Mediano y Bilateral.

4.- La Figura 2 muestra la primera hilera de comparaciones. La idea es mostrar el ruido, luego 2 filtros de suavizado y al final el original. Notamos que el filtro de caja (Box Filter) reduce significativamente el ruido pero los bordes se difuminan. El filtro gaussiano también suaviza muy bien la imagen y los bordes se distinguen mejor que el filtro de caja. En la Figura 3 se muestra el suavizado mediano y bilateral. El suavizado bilateral preserva muy bien los bordes y suaviza mejor que el filtro de caja y gaussiano. El mejor resultado se logró con el filtro mediano, ya que suaviza y define los bordes, eliminando prácticamente todo el ruido y delimitando los detalles de forma muy precisa.

## 5. Conclusión

En conclusión, el proceso de máscara sin afinar es muy útil para distinguir los bordes de una imagen para luego hacer un análisis más preciso al respecto. Tanto la función que usa el kernel gaussiano con el laplaciano, preservan los bordes de la imagen original y variando los parámetros tendremos otros resultados. Si el kernel gaussiano es más grande y con más  $\gamma$ , más bordes se van a resaltar y viceversa. Con

$\times 3$  pero se puede ir variando el valor de  $\alpha$ . El coeficiente es qué tanto de la máscara vamos a retirar, así que también es algo en tomar en cuenta.

Los filtros de suavizado son muy útiles para mejorar una imagen, dependiendo si queremos preservar los bordes o reducir una imagen podrá ser más útil un filtro que otro. El filtro mediano es el que regresa un mejor resultado junto con el bilateral quien pierde un poco el brillo original. Los filtros de caja y gaussiano suavizan la imagen pero no preservan bien los bordes, el gaussiano es ligeramente mejor por sus bordes resultantes. Dependiendo el caso, será mejor el filtro mediano o bilateral en general para eliminar ruido.

## A. Código

El código diseñado para esta práctica se encuentra en un repositorio de Github para su visualización y con la opción de abrir en Google Colab.

## Referencias

- [1] Chung, B. W. (2017). *Getting Started with Processing and OpenCV*, pages 1–37. Apress, Berkeley, CA.
- [2] Solem, J. (2012). *Programming Computer Vision with Python: Tools and algorithms for analyzing images*, pages 73–273. O'Reilly Media.
- [3] Woods, G. . (2018). *Digital Image Processing*, pages 1–200. Pearson, New York, NY.