Procesamiento de Señales utilizando la Convolución: Filtro Gaussiano

Pablo Bermeo
Facultad de ingeniería
Universidad de Cuenca
Cuenca, Ecuador
pablo.bermeo@ucuenca.edu.ec

Sebastián Guazhima
Facultad de ingeniería
Universidad de Cuenca
Cuenca, Ecuador
sebastian.guazhima@ucuenca.edu.ec

Tyrone Novillo
Facultad de ingeniería
Universidad de Cuenca
Cuenca, Ecuador
tyrone.novillo@ucuenca.edu.ec

***Resumen*—This report presents an analysis of image processing utilizing the convolution property between an image and a Gaussian filter. To achieve this, a MATLAB script has been developed specifically for convolving a BMP image with a matrix representing the Gaussian filter. The primary objective is to determine the impulse response of the system, denoted as h(t), which filters images to produce a blurring effect. Additionally, the iterative operation of convolution between the kernel (Gaussian filter) and the image will be scrutinized to comprehend its impact.**

***Keywords*—** Image processing, Convolution, Gaussian filter, MATLAB, Blurring effect, Impulse response, Image filtering

I. Introducción

En el ámbito del procesamiento de imágenes, el análisis y la aplicación de propiedades como la convolución entre imágenes y filtros, como el filtro gaussiano, representan áreas fundamentales. En este contexto, se desarrolla un enfoque para comprender y manipular imágenes digitales mediante la convolución con filtros específicos. La exploración de la respuesta al impulso del sistema y la operación iterativa de convolución entre el kernel y la imagen, constituyen elementos cruciales para comprender la transformación y el procesamiento de las imágenes. Este enfoque permite no solo entender el efecto de filtrado de una imagen, sino también evaluar cómo esta operación influencia la representación visual de la información contenida en las imágenes.

II. Marco teórico

II-A. Sistemas lineales

Los sistemas, dentro del ámbito de la teoría de señales y sistemas, representan una base fundamental para comprender cómo se manipulan y transforman las señales. Estos elementos intentan modelar procesamiento de información representada en señales. Su estudio se centra en propiedades clave como la linealidad, que permite combinar y escalar entradas de manera predecible, y la invarianza en el tiempo, que garantiza que el comportamiento del sistema permanezca constante a lo largo del tiempo [1].

Los sistemas LTI son una clase fundamental de sistemas en el campo del procesamiento de señales y sistemas. Estos sistemas poseen dos propiedades cruciales: linealidad e invarianza en el tiempo. La linealidad implica que el sistema sigue el principio de superposición, donde la respuesta a una suma ponderada de señales de entrada es la misma suma ponderada de las respuestas individuales a cada señal. La invarianza en el tiempo indica que el comportamiento del sistema no cambia con el tiempo, es decir, si la entrada se desplaza en el tiempo, la salida se desplaza de la misma manera sin alterar su forma. Estas características hacen que los sistemas LTI sean fundamentales en el análisis y diseño de sistemas de procesamiento de señales [2].

La respuesta al impulso unitario, denotada como $h(t)$ en sistemas continuos y $h[n]$ en sistemas discretos, es la base fundamental para entender y caracterizar los sistemas LTI. En un sistema lineal, la respuesta al impulso unitario es la salida del sistema cuando se aplica un impulso unitario como entrada. Esta respuesta describe completamente el comportamiento del sistema para cualquier entrada posible, ya que cualquier señal de entrada puede ser considerada como la suma ponderada de versiones escaladas y desplazadas del impulso unitario tal como lo describe la propiedad de *shifting* de la ecuación 1 [3].

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k] \tag{1}$$

La figura 1 muestra un sistema LTI caracterizado por la respuesta al impulso unitario con una señal de entrada $x(t)$ lo que produce la salida $y(t)$

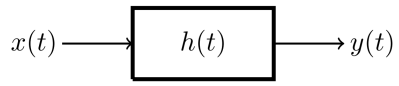

Figura 1: Diagrama de un sistema LTI básico con entrada $x(t)$, respuesta al impulso $h(t)$ y salida $y(t)$ [1].

Los sistemas lineales discretos tienen aplicaciones amplias y variadas en campos como el procesamiento digital de señales, telecomunicaciones, procesamiento de imágenes y sistemas de control. Por ejemplo, en procesamiento de imágenes, la convolución se utiliza extensamente para aplicar filtros a imágenes, como el filtrado gaussiano para el suavizado o el filtrado de paso alto para la detección de bordes. Además, en telecomunicaciones, los sistemas lineales discretos son esenciales en la modulación y demodulación de señales para la transmisión de datos digitales a través de canales de comunicación [4].

II-B. Convolución

La convolución en sistemas LTI discretos es una operación central que describe la relación entre la entrada y la salida de un sistema. Se utiliza para modelar cómo un sistema responde a diferentes señales de entrada a lo largo del tiempo. En este contexto, la convolución se realiza multiplicando cada valor de la secuencia de entrada por los valores correspondientes de la respuesta al impulso del sistema, desplazados en el tiempo, y sumando estos productos, es decir, se

vale de la propiedad de *shifting* (Ec. 1). Esta operación se expresa matemáticamente como:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k] \tag{2}$$

En sistemas LTI, la convolución permite entender y predecir la salida del sistema ante diferentes entradas, lo que resulta fundamental en el diseño y análisis de sistemas de control, filtrado de señales, y diversas aplicaciones en ingeniería y ciencias aplicadas [5].

II-C. Procesamiento de imagenes: Filtros espaciales

Explicar como funcionan los filtros espaciales y la convolución de dos dimensiones.

El empleo del procesamiento de imágenes ha ganado una relevancia notable tanto en ámbitos científicos como en el arte fotográfico. Esta herramienta posibilita el análisis de capturas digitales, abarcando desde fotografías de telescopios, microscopios, imágenes satelitales y mapeos topográficos, entre otros. La evolución del procesamiento digital ha permitido su implementación en aplicaciones que van desde la edición básica de imágenes hasta la extracción de datos significativos mediante diversas técnicas de procesamiento [6].

Entre los métodos utilizados en el procesamiento de imágenes se encuentra el filtrado, que permite obtener una versión modificada de una imagen fuente, seleccionando, agregando o suprimiendo información específica de esta. Los filtros espaciales, tales como los de paso bajo, paso alto o detección de bordes, son considerados operaciones donde el nuevo pixel calculado no depende únicamente del original sino de los pixeles situados a su alrededor [7] tal como lo indica la figura 2.

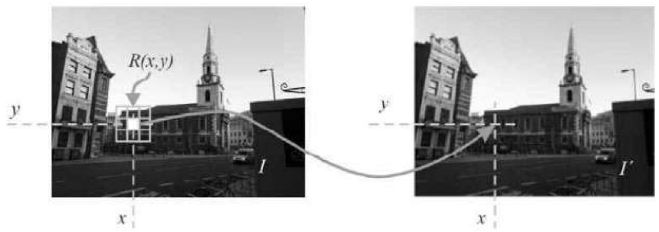

Figura 2: Principio de las operaciones de filtro espacial. Cada nuevo pixel se calcula a partir de una determinada región $R(x, y)$ de vecinos cuyas coordenadas centrales están relacionadas con las del pixel a calcular [2].

Una imagen se define como un arreglo de píxeles en dos o tres dimensiones, siendo en este caso un arreglo bidimensional de píxeles con distintas intensidades luminosas. Esta imagen puede ser representada como una función $f(x, y)$, donde x e y son variables espaciales, y su conjunto de píxeles se organiza en una matriz en dos dimensiones $I = f(x, y)$ [8].

El filtrado espacial se basa en la aplicación del método matemático de la convolución lineal de dos dimensiones. En este procedimiento, una matriz de filtro o máscara de convolución $H(i, j)$ se aplica a la imagen original $f(x, y)$. En otras palabras, los píxeles de la imagen contenidos en la región definida por la matriz se multiplican por los coeficientes de dicha matriz, sumando luego cada resultado de estas multiplicaciones para obtener una imagen resultante. En este contexto, la matriz es conocida como filtro, máscara, kernel o ventana [9].

El kernel es una matriz 2D con un tamaño menor a la imagen original y realiza un barrido a la misma. En cada posición al realizar el barrido se hace la convolución entre el kernel y una porción de la imagen [10].

Las ecuaciones 3 y 4 ilustran el proceso de aplicación del filtro mediante la convolución.

$$I = f * H \tag{3}$$

O su equivalente:

$$I(x, y) = \sum_{i=1}^{n} \sum_{j=1}^{m} f(x-i, y-j) \cdot H(i, j) \tag{4}$$

II-D. Efecto Blurring - Filtro Gaussiano

II-D1. Efecto Blurring: El efecto Blurring en el procesamiento de imágenes se utiliza para realizar mejoras a la misma, en el cual, la principal característica es realizar un difuminado o suavizado que reduce entre píxeles adyacentes su variación de intensidad (contrastes abruptos). Y los operadores para obtener este efecto es el filtro de la media y el filtro gaussiano [11].

- El filtro de la media es sensible a cambios entre píxeles cercanos, el cual crea intensidades de grises que indebidas.
- El filtro Gaussiano utiliza otro tipo de máscara y es separable, en el cual se puede dividir la convolución bidimensional en dos convoluciones unidimensionales, obteniendo resultados mas naturales. [12]

II-D2. Filtro Gaussiano: Este tipo de filtro se basa en la aplicación de la función Gaussiana (Ecuación 5) a una imagen usando la operación de convolución, esta función tiene la característica de ser bidimensional y discreta [13].

$$G_\sigma(i, j) = \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}\left(\frac{d[centro,(i,j)]}{\sigma^2}\right)} \tag{5}$$

Donde el filtro, tiene asociada a la gráfica de la figura 3 también llamada la campana de Gauss o curva de distribución normal.

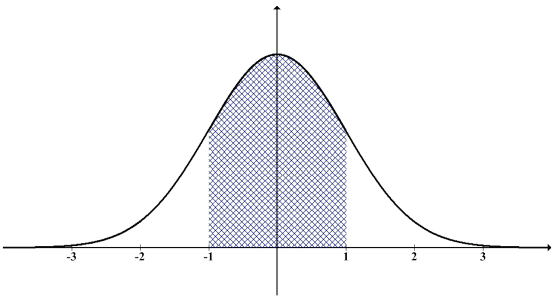

Figura 3: Gráfica del filtro Gaussiano [14]

La figura 4 muestra una matriz cuadrada de dimensión 15x15 la cual representa el filtro Gaussiano con una desviación estándar 3.

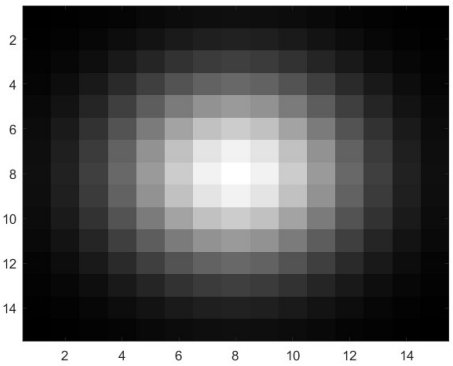

Figura 4: Mapa de calor que representa el kernel Gaussiano. Donde el blanco indica mayor ponderación y el negro la más baja.

El valor de cada $G(i, j)$ de la máscara responderá a la distribución normal en función de las distancias hacia el centro. Note que el

mayor valor de la máscara se ubica en el centro, debido a que $d[centro,(i,j)] = 0$ y por lo tanto el valor del mismo es el de la ecuación 7.

$$G_\sigma(centro) = \frac{1}{2\pi\sigma^2} \tag{6}$$

En cambio, los valores circundantes experimentan un factor exponencial decreciente. En este sentido se consigue una distribución normal que cumple la propiedad

$$\sum_{i=1}^{k}\sum_{j=1}^{k} G(i,j) = 1 \tag{7}$$

Es decir que las columnas del Kernel Gaussiano, cumplen con la propiedad de normalidad de la distribución normal.

III. Desarrollo

A continuación, se explica el proceso para implementar un script de Matlab el cual ejecuta la operación de convolución entre una imágen y el kernel correspondiente al filtro Gaussiano para lograr el efecto de *blurring* a la salida.

III-A. Generación del Kernel Gaussiano

El objetivo planteado en este punto corresponde a la formación de la respuesta al impulso que será convolucionada con la imagen de entrada.

La respuesta al impulso se refiere a la máscara Gaussiana, que responde a dos parámetros esenciales, denominados,`k` y `sigma`. El valor `k` define la dimensión de la máscara, que es cuadrada e impar (a fin de que tenga un centro definido). Luego, `sigma` asigna el valor de la desviación estándar de la ecuación 5.

Partiendo de los conceptos nombrados en el marco teórico, es posible generar una respuesta la máscara Gaussiana a través del siguiente proceso:

1. Definir los valores para `k` y `sigma`.
2. A continuación se considera la distancia cuadrática de cada elemento de la matriz con respecto al centro. El resultado de este proceso se almacena en `CenterDist`. Posterior a este resultado, se aplica una distribución normal a partir de la ecuación 5. Cada resultado se almacena en todo `GaussKernel(i,j)` Una explicación gráfica del comportamiento cíclico del procedimiento, se encuentra en la Figura 5.

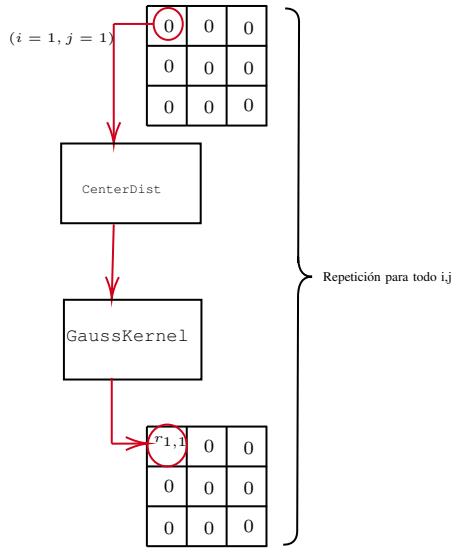

Figura 5: Descripción gráfica del bucle generador del Kernel.

III-B. Lectura de imagen

Previo a la operación de convolución bidimensional es necesario preparar la imágen a aplicar el filtro. Para ello, primero se debe tratar con imágenes con extensión .bmp (mapa de bits). Dicha extensión es un archivo de imagen sin compresión que almacena datos de imagen pixel por pixel. Para aplicar un filtro Gaussiano mediante convolución, es necesario representar la imagen como una matriz de valores de pixeles. Es importante considerar que la convolución implica "deslizar una ventana" (filtro Gaussiano) sobre la imagen y hacer multiplicaciones y sumas entre los elementos de una matriz de imagen y el kernel. Sin embargo, otros formatos de archivo de imagen como JPEG o PNG pueden ser igualmente utilizados para aplicar filtros y convoluciones, pero podrían requerir procesos adicionales para interpretar la información de la imagen debido a su compresión o estructuras de archivo más complejas.

También se optó por trabajar con imgénes en escala de grises debido a que simplifican el proceso de convolución. Las imágenes en escala de grises son matrices bidimensionales donde cada elemento representa la intensidad de un pixel mas no un array de los colores que componen a dicho pixel. Esto no afecta al análisis del efecto del filtro sobre la imagen debido a que el filtro busca suavizar la imagen, un proceso que prioriza la intensidad de los pixeles que los colores.

III-C. Convolución de la imagen con el kernel Gaussiano

III-C1. Problema de los bordes: Antes de considerar el proceso de convolución discreta entre la máscara Gaussiana y la imagen de entrada, es necesario hablar sobre el comportamiento de la máscara en los bordes.

Tal como se visualiza en la Figura 6, la máscara excede la cantidad de píxeles de la imagen en los bordes. Las soluciones a este problema contemplan la ignorancia de los bordes, o la asignación de valores a constantes a la cantidad de píxeles requeridos.

La solución adoptada en este trabajo corresponde a la asignación de valores constantes. Específicamente, se han asignado valores de 0.

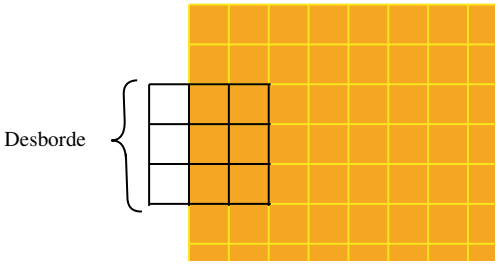

Figura 6: Problema de desborde al aplicar una máscara.

III-D. Operación de convolución

La codificación de la operación de convolución se compone a través de la evaluación del Kernel o máscara Gaussiana sobre cada uno de los píxeles de la entrada. Luego de que la máscara se ubica sobre un píxel, los valores limitados por la vecindad cubierta, se multiplican uno a uno. Esta multiplicación se realiza en función de su posición. Luego, los valores de la matriz interna resultante de la multiplicación, se suman.

Este proceso tiene un comportamiento cíclico y opera para todos los píxeles de la imagen.

Tal como se muestra en la Figura 7, el proceso codificado para este propósito elige para i,j que tienen como límite las dimensiones de la imagen y genera una matriz temporal. La última matriz temporal se multiplica con la máscara Gaussiana para luego ser sumada en todos sus valores. El valor resultante se vuelve a introducir en el píxel seleccionado inicialmente.

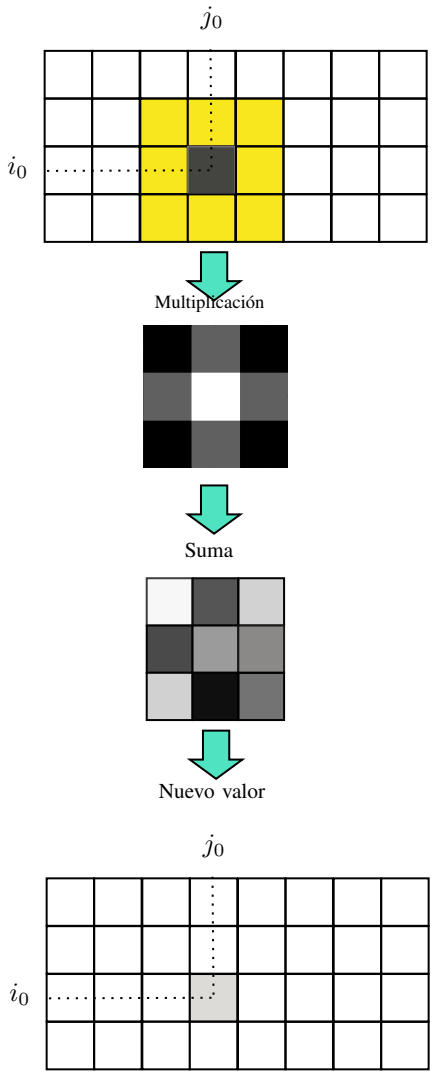

Figura 7: Descripción gráfica del proceso de convolución para cada par i,j

IV. Resultados

Figura 8: Imagen a aplicar el filtro Gausiano [15]

Figura 9: Resultado de la operación de convolución con filtro Gaussiano de tamaño 15x15 y sigma 1.

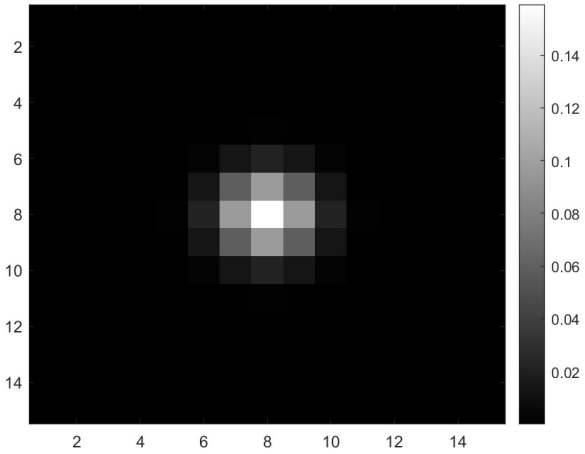

Figura 10: Respuesta al impulso - Kernel - Filtro Gaussiano de tamaño 15x15 y sigma 1.

Figura 11: Resultado de la operación de convolución con filtro Gaussiano de tamaño 15x15 y sigma 2

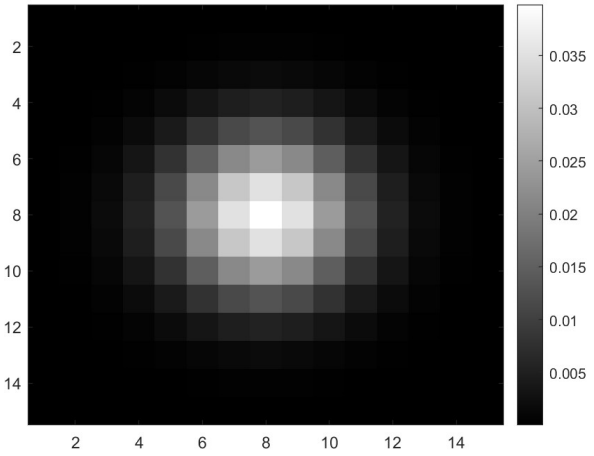

Figura 12: Respuesta al impulso - Kernel - Filtro Gaussiano de tamaño 15x15 y sigma 2.

Figura 13: Resultado de la operación de convolución con filtro Gaussiano de tamaño 15x15 y sigma 4

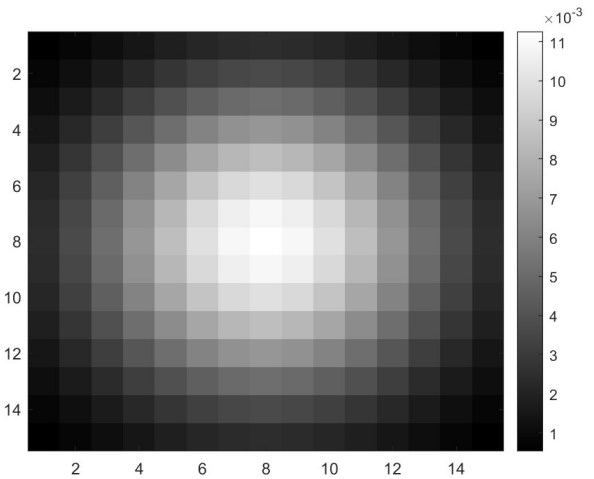

Figura 14: Respuesta al impulso - Kernel - Filtro Gaussiano de tamaño 15x15 y sigma 4.

V. Conclusiones

El kernel Gaussiano, al ser un conjunto de valores centrados alrededor del punto central del kernel, pondera los píxeles cercanos más que los píxeles más alejados. Esto significa que los píxeles vecinos tienen un impacto mayor en el resultado de la convolución que los píxeles más distantes. Esto crea un efecto de suavizado al combinar suavemente los valores de los píxeles adyacentes, disminuyendo así las transiciones bruscas entre áreas de alta y baja intensidad en la imagen.

El suavizado se logra al promediar los valores de los píxeles de la imagen original con los valores del filtro Gaussiano en una vecindad determinada. El resultado es una imagen suavizada donde las transiciones abruptas se atenúan, lo que produce un efecto de desenfoque controlado y una reducción en el ruido presente en la imagen original.

La operación de convolución bidimensional con el filtro gaussiano ha permitido suavizar las imágenes, reduciendo el ruido y realzando características de forma efectiva. Se ha observado que al variar el tamaño del kernel y la desviación estándar, el efecto del filtro gaussiano cambia significativamente en la imagen resultante.

La variación del kernel ha mostrado un impacto directo en la intensidad del suavizado. Al ajustar la desviación estándar, se ha evidenciado que valores mayores han generado un suavizado más extendido, mientras que valores menores han mantenido bordes más definidos.

Estas conclusiones destacan la importancia de elegir cuidadosamente el tamaño del kernel y la desviación estándar al aplicar el filtro Gaussiano. En conjunto, el filtro Gaussiano aplicado mediante la operación de convolución permite procesar imágenes para el tratamiento de ruido y la mejora de características visuales.

Referencias

[1] A. V. Oppenheim and A. S. Willsky, *Signals and Systems*, 1st ed. Prentice-Hall, 1996.

[2] C. .Erik and D. Margarita, *Tratamiento de Imagenes con MATLAB*, 1st ed., AlfaOmega, Ed., 2017.

[3] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, 1st ed. Prentice Hall, 1996.

[4] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 1st ed. Pearson Education, 2017.

[5] B. P. Lathi and Z. Ding, *Modern Digital and Analog Communication Systems*, 3rd ed. Oxford University Press, 2009.

[6] T. Acharya and A. K. Ray, *Image Processing Principles and Applications*. [Online]. Available: www.thietbiysinh.com.vn

[7] F. Palomares, J. M. M. in Science ..., and undefined 2016, "Aplicación de la convolución de matrices al filtrado de imágenes," *polipapers.upv.es*. [Online]. Available: https://polipapers.upv.es/index.php/MSEL/article/view/4524

[8] J. Elizondo, L. M. M. U. A. de, and undefined 2005, "Fundamentos de procesamiento de imágenes," *academia.edu*. [Online]. Available: https://www.academia.edu/download/35915704/FundamentosDeProcesamientoDeImagenes.pdf

[9] D. Peña and F. Balt, "Técnicas de filtrado por mascara de convolucion y segmentación de color para procesamiento digital de imágenes." *academia.edu*. [Online]. Available: https://www.academia.edu/download/49410760/Tecnicas_de_filtrado_por_convolucion_y_segmentacion_de_color_para_procesamiento_digital_de_imagenes.pdf

[10] 3.3 - el filtro o kernel — codificando bits. [Online]. Available: https://www.codificandobits.com/curso/fundamentos-deep-learning-python/redes-convolucionales-3-filtro-o-kernel/

[11] Q. Simbeni, "Procesamiento digital de imágenes," 7 2015.

[12] S/N, "Tema 3: Filtros 1." [Online]. Available: https://alojamientos.us.es/gtocoma/pid/tema3-1.pdf

[13] f. given i=P, given=Pablo. Filtros y convoluciones (procesamiento de imxe1;genes) (pxe1;gina 2). [Online]. Available: https://www.monografias.com/trabajos108/filtros-y-convoluciones/filtros-y-convoluciones2

[14] g. given i=A. Distribución normal. [Online]. Available: https://estamatica.net/distribucion-normal/

[15] "Filesamples - bmp file format," https://filesamples.com/formats/bmp, accedido: 10 de noviembre del 2023.

VI. Anexos

VI-A. Github

La dirección https://github.com/thyron001/filtro_gaussiano corresponde al repositorio que contiene los ficheros utilizados en este reporte como el script y las imágenes de prueba.

VI-B. Script de Matlab

Kernel Gaussiano | Lectura de imagen | Convolución | Resultados

```
%=========================================================
%                  Definición del Kernel Gaussiano
%=========================================================

%Desviación estándar
sigma = 1;

%Dimensión del Filtro Gaussiano
k = 3;

GaussKernel = zeros(k,k);
center = (k+1)/2;

for i = 1:k

    for j = 1:k

        CenterDist = (i-center)^2 + (j-center)^2;
        GaussKernel(i,j) = exp( -1*( CenterDist ) / ( 2*(sigma^2) ) );
    end
end

%Normalizacion
GaussKernel = GaussKernel/(2*pi*(sigma)^2);
```

Kernel Gaussiano | Lectura de imagen | Convolución | Resultados

```
%=========================================================
%      Lectura de Imagen y obtención de valores de filas y columnas
%=========================================================

img = imread("ex1.bmp");
%I = img;
I = rgb2gray(img);
```

Kernel Gaussiano | Lectura de imagen | Convolución | Resultados

```
%=========================================================================
%            Aplicación del Flitro Gaussiano al objeto "img"
%=========================================================================

%Asignar un valor de 0 a los valores de los bordes
[M,N] = size(I);
Im = padarray(I,[(k-1)/2 (k-1)/2 ],0);
salida = zeros(M,N);

for i = 1:M

    for j = 1:N

        temporal = Im( i:i+k-1, j:j+k-1);
        temporal = double(temporal);
        conv = temporal.*GaussKernel;
        salida(i,j) = sum(conv(:));

    end

end
```

Kernel Gaussiano | Lectura de imagen | Convolución | Resultados

```
%=========================================================================
%                        Muestra de resultados
%=========================================================================

figure(1);

set(gcf,'Position',get(0,'ScreenSize'));
subplot(121),imshow(Im),title('Original');
subplot(122),imshow(salida,[0, 255]),title('Salida');


figure(2);

imagesc(GaussKernel);
colormap(gray(256));

figure(3)
surf(GaussKernel)
colormap(gray)
shading interp
```