

Procesamiento de imágenes: smoothing/blurring implementado a través del método averaging

Marco Valarezo, Maria Jose Vire, Lenin Zhingri

Facultad de Ingeniería, Universidad de Cuenca, Ecuador

marcof.valarezoy@ucuenca.edu.ec

mjose.vire@ucuenca.edu.ec

lenin.zhingrib@ucuenca.edu.ec

Resumen —Este informe detalla el proceso y resultados de la implementación de un algoritmo de suavizado de imágenes utilizando el método de averaging o promedio. El código es implementado en el lenguaje de programación Python.

- ❖ Evaluar la eficacia del proceso de convolución con el filtro de averaging en la reducción de ruido.
- ❖ Analizar el efecto de suavizado en los detalles de la imagen.

I. INTRODUCCIÓN

El procesamiento de imágenes es una disciplina que permite manipular y analizar datos visuales. Una de las técnicas fundamentales en esta área es el suavizado o desenfoque, que reduce el ruido en las imágenes. Este informe se centra en la técnica de promedio, un método que consiste en suavizar una imagen calculando la media de los píxeles en una vecindad determinada. Esta técnica es ampliamente utilizada para reducir el ruido, mejorar la calidad visual y simplificar la imagen para su posterior análisis.

II. OBJETIVOS

A. Objetivo general.

- ❖ Aplicar la técnica de convolución para reducir el ruido y suavizar detalles en imágenes utilizando el método de averaging.

B. Objetivos específicos.

- ❖ Implementar el efecto de smoothing/blurring utilizando el filtro de averaging.

III. FUNDAMENTO TEÓRICO

Procesamiento de Imágenes Digitales

El procesamiento de imágenes digitales es una disciplina que involucra la manipulación y mejora de imágenes mediante técnicas computacionales. Las imágenes digitales son representaciones bidimensionales de una escena que contienen una matriz de valores de intensidad de píxeles. Cada píxel, representado en coordenadas (x,y) , tiene un valor que refleja la intensidad luminosa en esa posición específica. Este campo ha cobrado gran relevancia en aplicaciones como la visión por computadora, el diagnóstico médico y el monitoreo de seguridad, donde la calidad de las imágenes es crucial (Gonzalez & Woods, 2018).

Ruido en Imágenes

El ruido en imágenes digitales es una distorsión no deseada que afecta la calidad de la imagen. Este puede originarse de varias fuentes, como condiciones de captura inadecuadas o la transmisión de la imagen. El ruido puede manifestarse de

diferentes formas, incluyendo ruido gaussiano, sal y pimienta, y ruido de impulso, cada uno afectando la imagen de manera particular. Para reducir los efectos del ruido y mejorar la calidad visual, se emplean técnicas de smoothing o suavizado (Jain, 1989).

Suavizado de imágenes

El suavizado de imágenes es una técnica utilizada para reducir el ruido y suavizar los detalles finos de una imagen. Esto se logra mediante el ajuste de los valores de los píxeles basándose en los valores de sus vecinos, generando una transición más uniforme y menos ruidosa entre los diferentes niveles de intensidad. Entre las técnicas de suavizado se encuentran el filtro de la mediana, el filtro gaussiano y el filtro de promedio (averaging), cada uno con aplicaciones específicas según el tipo de imagen y el tipo de ruido (Castleman, 1996).

Convolución en Procesamiento de imágenes

La convolución es una operación matemática clave en el procesamiento de imágenes y es fundamental para la aplicación de filtros. Consiste en deslizar una matriz pequeña (kernel) sobre cada píxel de la imagen, multiplicando los valores del kernel por los valores de la submatriz de píxeles correspondiente y sumando los resultados. Esto produce una imagen nueva con valores ajustados. Esta técnica permite modificar los valores de los píxeles en función de sus vecinos, aplicando así filtros que reducen el ruido o resaltan características específicas (Szeliski, 2010).

Filtro de Promedio (Averaging)

El filtro de promedio, también conocido como filtro de averaging, es un tipo de filtro lineal que emplea la convolución con un kernel en el que todos los valores son iguales. Este método consiste en calcular el promedio de los valores de los píxeles en una ventana específica (por ejemplo, un kernel de 3x3 o 5x5) alrededor de cada píxel y asignar ese valor promedio al píxel central. El filtro de promedio suaviza la imagen al reducir las variaciones abruptas en los valores de intensidad entre píxeles adyacentes. No obstante, como efecto secundario, también

tiende a desdibujar los bordes y detalles finos de la imagen (Jain, 1989; Gonzalez & Woods, 2018).

Supongamos que tenemos una imagen en escala de grises representada por una matriz de 5x5 con valores de píxeles que van de 0 a 255 (el rango común para imágenes en escala de grises) como en la Figura 1.

$$\begin{bmatrix} 34 & 30 & 28 & 30 & 34 \\ 34 & 30 & 28 & 30 & 34 \\ 34 & 30 & 28 & 30 & 34 \\ 34 & 30 & 28 & 30 & 34 \\ 34 & 30 & 28 & 30 & 34 \end{bmatrix}$$

Figura 1. Matriz de ejemplo.

Para aplicar un filtro de promedio, se usa un kernel de 3x3. Este kernel nos indica que vamos a considerar el píxel central y sus 8 vecinos. El valor de cada píxel en la imagen resultante será el promedio de los valores dentro de esta ventana de 3x3. Un kernel promedio 3x3 se ve como la Figura 2.

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Figura 2. Kernel de 3x3.

Para aplicar el filtro, se coloca el kernel 3x3 sobre cada píxel de la imagen. Luego se multiplica cada valor del kernel por el valor del píxel correspondiente en la ventana. Se suman todos los productos obtenidos y se divide por el total de elementos del kernel, que en este caso es 9. El valor resultante será el nuevo valor del píxel en la posición central de la ventana.

Para aplicar el filtro al píxel central (posición(3,3)) de la matriz de la Figura 1, se toman los valores en la ventana de 3x3.

$$\begin{bmatrix} 30 & 28 & 30 \\ 30 & 28 & 30 \\ 30 & 28 & 30 \end{bmatrix}$$

Figura 3. Valores en ventana 3×3 del píxel central de la matriz de la Figura 1.

Se calcula el promedio, para ellos se suman los valores de la submatriz:

$$30 + 28 + 30 + 30 + 28 + 30 + 28 + 30 = 264$$

Se divide por el número de elementos (9) para obtener el promedio:

$$\frac{264}{9} \approx 29$$

Entonces el nuevo valor del píxel central será 29. Este mismo proceso se repite para cada píxel de la matriz de la Figura 1, generando una versión suavizada de la imagen original.

Si se aplica el filtro a todos los píxeles de la matriz original de manera sistemática, se obtiene una imagen suavizada en la que los valores de los píxeles son los de la Figura 4.

$$\begin{bmatrix} 32 & 30 & 29 & 30 & 32 \\ 32 & 30 & 29 & 30 & 32 \\ 32 & 30 & 29 & 30 & 32 \\ 32 & 30 & 29 & 30 & 32 \\ 32 & 30 & 29 & 30 & 32 \end{bmatrix}$$

Figura 4. Resultado de aplicar Filtro de Promedio la matriz de la Figura 1.

Para imágenes en color (como las que están en formato RGB), el proceso de convolución con el filtro de promedio es muy similar al caso de imágenes en escala de grises, pero se realiza de manera independiente en cada uno de los tres canales de color: rojo (R), verde (G) y azul (B).

En el proceso de Convolución con el filtro de promedio en imágenes en Color se realiza una separación de canales. Una imagen en color está compuesta de tres canales: rojo (R), verde (G) y azul (B), que definen los valores de cada pixel en términos de intensidad de color en cada componente. Cada uno de estos canales es en sí mismo una

matriz de intensidad de píxeles, similar a una imagen en escala de grises. Por ejemplo, para una imagen de tamaño 5×5 , cada canal tendrá una matriz 5×5 .

El filtro de promedio se aplica de manera independiente en cada uno de estos canales. Esto significa que, al aplicar el filtro en el canal rojo, solo estamos afectando los valores de intensidad roja de cada píxel, y lo mismo se aplica para los canales verde y azul.

Por ejemplo, si se usa el kernel de la Figura 2 se calculará el promedio de los valores de los píxeles en una ventana 3×3 alrededor de cada píxel para cada canal.

Para calcular el nuevo valor de un píxel en la imagen de salida se toma el canal rojo y se aplica la convolución (el filtro de promedio) en cada píxel usando el kernel. Lo que implica sumar los valores de los 9 píxeles en la ventana de 3×3 , dividir la suma por 9, y asignar el valor promedio resultante al píxel central en el canal rojo de la imagen de salida. Se repite el mismo procedimiento para el canal verde y luego para el canal azul. El proceso para cada canal de un píxel podría lucir así:

- Valor del nuevo píxel rojo (R') = promedio de la ventana 3×3 de valores rojos.
- Valor del nuevo píxel verde (G') = promedio de la ventana 3×3 de valores verdes.
- Valor del nuevo píxel azul (B') = promedio de la ventana 3×3 de valores azules.

Después de aplicar el filtro de promedio a cada canal, los tres canales resultantes (R, G y B) se combinan para formar la imagen final suavizada. Cada píxel en la imagen final tendrá sus componentes de color suavizados, resultando en una imagen que presenta menos ruido y transiciones de color más uniformes.

Supongamos que un píxel central y sus vecinos tienen valores de la Figura 5,6 y 7 para cada canal.

$$\begin{bmatrix} 255 & 200 & 200 \\ 150 & 100 & 100 \\ 50 & 0 & 0 \end{bmatrix}$$

Figura 5. Píxel central y valores vecinos, Canal rojo (R)

255	220	210
160	120	110
60	20	10

Figura 6. Píxel central y valores vecinos, Canal verde (G)

240	180	160
140	100	80
60	20	10

Figura 7. Píxel central y valores vecinos, Canal azul (B)

Se aplica el filtro de promedio en cada canal por separado, tomando el promedio de los valores en cada 3×3 .

➤ Rojo (R'):

$$\frac{255+200+200+250+200+200+50+0+0}{9} = \frac{1055}{9} \approx 117$$

➤ Verde (G'):

$$\frac{255+220+210+160+120+110+60+20+10}{9} = \frac{1165}{9} \approx 129$$

➤ Azul (B'):

$$\frac{240+180+160+140+100+80+60+20+10}{9} = \frac{990}{9} \approx 110$$

Después de aplicar el promedio en cada canal, el nuevo valor del píxel central será:

$$(R', G', B') = (117, 119, 110)$$

Este proceso se repite para cada píxel de la imagen, generando una versión suavizada en color, donde cada canal ha sido procesado de manera independiente.

El Proceso de Convolución con el Filtro de Promedio suaviza las transiciones de color y reduce el ruido en la imagen, pero también tiende a desdibujar los bordes y detalles finos, como en las imágenes en escala de grises.

Para evitar efectos no deseados en los bordes, se suelen aplicar técnicas de "padding" o relleno, en las que los bordes de la imagen se expanden o se reflejan, permitiendo que el filtro se aplique uniformemente.

Tamaño del Kernel

Cuando cambias el tamaño del kernel en un filtro de promedio (o cualquier filtro de suavizado en general), esto afecta el grado de suavizado en la imagen. Un kernel más grande abarca una región más amplia de píxeles alrededor del píxel central. Esto

significa que los valores de intensidad de color se promedian en una área más grande, generando una imagen con transiciones más suaves y un aspecto menos detallado. Con un kernel grande, el ruido se reduce más efectivamente porque las variaciones pequeñas entre píxeles se diluyen en el promedio. Sin embargo, también se pierden detalles finos, como bordes y texturas, ya que los valores de los píxeles se apllanan en áreas más grandes. Un kernel más pequeño toma en cuenta menos píxeles vecinos, por lo que el promedio es menos pronunciado y el efecto de suavizado es menor. Con un kernel pequeño, el suavizado afecta solo un pequeño grupo de píxeles alrededor del píxel central, lo que permite conservar más detalles y bordes en la imagen. Como se promedian menos píxeles, el filtro de promedio no logra reducir tanto el ruido en la imagen. Esto puede ser útil si solo deseas un suavizado leve y quieres preservar la nitidez general.

Aumentar el tamaño del kernel también aumenta el tiempo de procesamiento, ya que el método de convolución debe realizar más cálculos para cada píxel. Si se necesita una imagen muy suavizada (por ejemplo, en fondos o áreas donde el detalle no es importante), un kernel grande es ideal. Para preservar bordes o detalles en áreas importantes (como texto o contornos), se suele usar un kernel pequeño o técnicas alternativas.

IV. HERRAMIENTAS

- Pinetools.com (Herramienta para colocar el ruido a la imagen).
- Lenguaje de Programación Python 3.12.
- IDE: IntelliJ Community Edition 2023.2.2.

V. PROCEDIMIENTO EXPERIMENTAL Y ANÁLISIS

1. Primero, se obtuvo una imagen para la aplicación del método.



Figura 8. Imagen original.

2. Se utilizó la aplicación Pinetools para obtener la imagen con el ruido adecuado.



Figura 9. Imagen con una cantidad de 25 de ruido añadido.

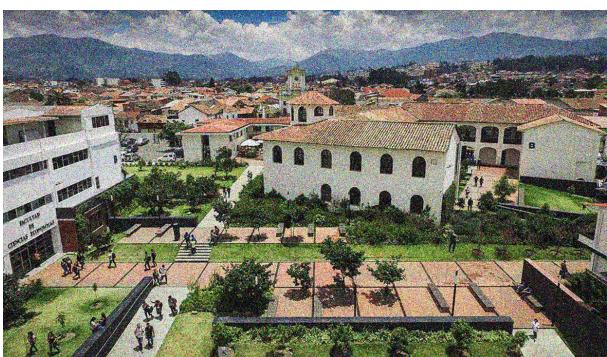


Figura 10. Imagen con una cantidad de 50 de ruido añadido.

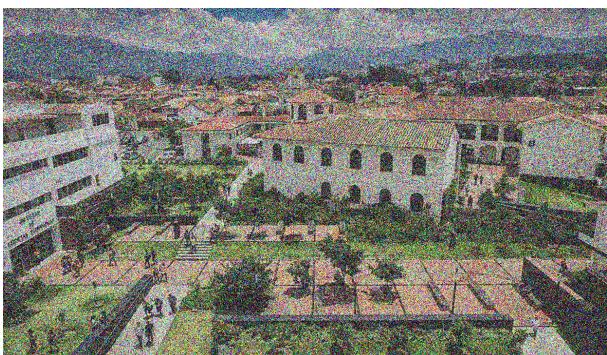


Figura 11. Imagen con una cantidad de 90 de ruido añadido.

3. **Seleccionar Niveles de Kernel:** Para efectos prácticos, se trabajará con tres niveles de kernel: 3 y 9. Estos valores determinarán la intensidad del suavizado en cada caso.
4. **Preparar el Entorno:** Antes de ejecutar el proyecto en Python, asegúrate de tener instaladas las librerías necesarias:
 - **Pillow** (para manejar imágenes con el paquete PIL).
 - **Matplotlib** (para visualizar los resultados).
5. Si no se encuentran instaladas, puede hacerlo ejecutando:
`pip install pillow matplotlib`.
6. Al ejecutar el código, se importará la imagen con ruido añadido para analizar el efecto de cada nivel de kernel. La librería PIL (disponible a través de Pillow) será la encargada de importar la imagen.
7. El código, incluido en los anexos, aplica el método de suavizado por promediado (*averaging*) utilizando el tamaño de kernel seleccionado. Esto reducirá el ruido en la imagen según la intensidad definida por cada kernel.
8. Al finalizar la ejecución, el programa mostrará en pantalla dos imágenes:
 - La imagen original con ruido.
 - La imagen suavizada mediante el método de promediado con el tamaño de kernel correspondiente.
9. Las dos imágenes aparecerán una al lado de la otra, lo que permite observar el efecto del filtro de suavizado con claridad.

VI. RESULTADOS

- Imágenes procesadas con cantidad de ruido de 25:



Figura 12. Imagen procesada con tamaño de kernel 3.



Figura 13. Imagen procesada con tamaño de kernel 9.

- Imágenes procesadas con cantidad de ruido de 50:

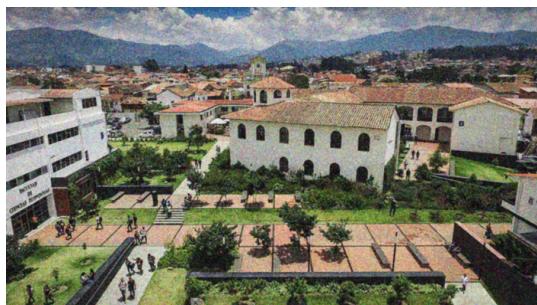


Figura 14. Imagen procesada con tamaño de kernel 3.

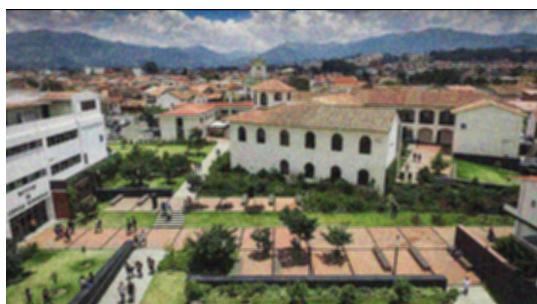


Figura 15. Imagen procesada con tamaño de kernel 9.

- Imágenes procesadas con cantidad de ruido de 90:

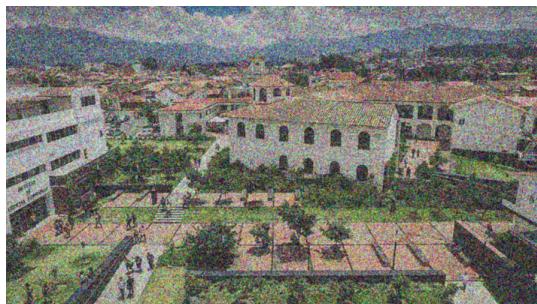


Figura 16. Imagen procesada con tamaño de kernel 3.



Figura 17. Imagen procesada con tamaño de kernel 9.

VII. CONCLUSIONES

Se puede concluir que la efectividad en la reducción de ruido depende de la intensidad de este mismo, en el smoothing/blurring mediante el averaging es eficaz si este ruido es bajo y su kernel es de 3 pero si esto aumenta la imagen que esperamos como resultado llega a perder detalles importantes y su calidad visual se degrada notablemente.

Una gran desventaja que este puede tener, no es bueno al preservar bordes y detalles, estos pueden terminar siendo difuminados.

VIII. RECOMENDACIONES

- **Usar imágenes con bajo ruido y pocos detalles:** Este método es muy bueno a la hora de trabajar con imágenes que contienen un bajo nivel de ruido y si dichas imágenes no contienen muchos detalles y bordes a la vez, así se garantiza tener una resultado aceptable y efectivo.
- **Evitar imágenes con alto ruido o con detalles importantes:** No es tan recomendable usar este método si vamos a trabajar con imágenes con un ruido alto ya que no está enfocado en determinar bordes y detalles importantes de la imagen demostrando que este no es un método óptimo para cualquier tipo de imagen y de ruido que esta contenga .
- **Ajuste del tamaño del kernel:** El kernel pequeño de 3 x 3 es adecuado para suavizados ligeros, mientras que el kernel mayor de 5 x 5 es

útil para reducir ruido más intenso, aunque a costa de un mayor desenfoque en los detalles

IX. REFERENCIAS BIBLIOGRÁFICAS

- [1] Castelman, K. R. (1996). *Digital Image Processing*. Prentice Hall.
 - [2] Gonzalez, R. C., & Woods, R. E. (2018). *Digital Image Processing* (4a ed.). Pearson.
 - [3] Jain, A. K. (1989). *Fundamentals of Digital Image Processing*. Prentice Hall.
 - [4] Russ, J. C. (2007). *The Image Processing Handbook* (5a ed.). CRC Press.
 - [5] Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer.

X. ANEXO

```
1 from PIL import Image
2 import matplotlib.pyplot as plt
3
4
5     1 usage
6
7 def load_image(filepath):
8     # Carga la imagen desde el archivo dado y la convierte a RGB.
9     return Image.open(filepath).convert("RGB")
10
11
12     1 usage
13
14 def apply_averaging_blur(image, kernel_size):
15     # Aplica el método de suavizado de promediado en la imagen a color con un kernel de tamaño definido.
16     width, height = image.size
17     pixels = image.load()
18
19
20     # Crear una nueva imagen para almacenar el resultado
21     blurred_image = Image.new(mode="RGB", size=(width, height))
22     blurred_pixels = blurred_image.load()
23
24
25     # Calcular el rango del kernel
26     offset = kernel_size // 2
27
28
29     # Recorrer cada pixel de la imagen, evitando los bordes según el tamaño del kernel
30     for x in range(offset, width - offset):
31         for y in range(offset, height - offset):
32             # Inicializar sumas para cada canal de color
33             total_r, total_g, total_b = 0, 0, 0
34             count = 0
35
36             # Sumar los pixeles dentro del kernel
37             for i in range(-offset, offset + 1):
38                 for j in range(-offset, offset + 1):
39                     if i == 0 and j == 0:
40                         continue
41                     else:
42                         total_r += pixels[x + i, y + j][0]
43                         total_g += pixels[x + i, y + j][1]
44                         total_b += pixels[x + i, y + j][2]
45                         count += 1
46
47             # Calcular el promedio y asignar al pixel resultante
48             blurred_pixels[x, y] = (total_r // count, total_g // count, total_b // count)
49
50
51     # Mostrar la imagen resultante
52     plt.imshow(blurred_image)
53     plt.show()
```

Figura 17. Código en Python

```

26
27         for i in range(-offset, offset + 1):
28             for j in range(-offset, offset + 1):
29                 r, g, b = pixels[x + i, y + j]
30                 total_r += r
31                 total_g += g
32                 total_b += b
33                 count += 1
34
35             # Calcular el promedio para cada canal
36             blurred_pixels[x, y] = (total_r // count, total_g // count, total_b // count)
37
38
39     return blurred_image
40
41
42 filepath = "C:/Users/USUARIO/Downloads/image90.png"
43 kernel_size = 9
44 image = load_image(filepath)
45 # Aplicar el filtro de suavizado con el tamaño de kernel especificado
46 blurred_image = apply_averaging_blur(image, kernel_size)
47
48
49 # Mostrar la imagen original con ruido y la procesada
50 fig, axes = plt.subplots( nroows= 1, ncols= 2, figsize=(10, 5))
51 axes[0].imshow(image)
52 axes[0].set_title("Imagen Original con Ruido")
53 axes[0].axis("off")
54 axes[1].imshow(blurred_image)
55 axes[1].set_title("Imagen Suavizada")
56 axes[1].axis("off")
57
58 plt.show()

```

Figura 17.1 Código en Python.