Universidad Nacional Autónoma de México

Facultad de Ciencias

Reporte. Práctica 5 Restauración de la imagen.

Profesora: Dra. María Elena Martínez Pérez
Ayudante: Miguel Angel Veloz Lucas
Alumno: Hernández Sánchez Oscar José

20 de Noviembre, 2023

1. Objetivos

- Implementar el filtro promedio aritmético, geométrico y adaptativo. Comparar su desempeño en presencia de ruido.
- Implementar el filtro mediana adaptativo y comparar su desempeño comparado con el filtro mediana simple, en presencia de ruido.
- De acuerdo al modelo de degradación que sufre una serie de imágenes, encontrar y aplicar el filtro de Wiener adecuado para la restauración óptima de cada imagen.

2. Introducción

Como en el realce de imágenes, la meta final de las técnicas de restauración es mejorar la imagen en un sentido predeterminado. A pesar de que existen áreas de sobrelape, el realce de una imagen es un proceso altamente subjetivo, mientras que la restauración de una imagen es parte de un proceso objetivo.

La restauración intenta reconstruir o recobrar una imagen que ha sido degradada utilizando conocimiento a priori del modelo de degradación y aplicando el proceso inverso a este para poder recobrar así la imagen original. Si H es lineal y es un proceso invariante a la posición, entonces la imagen degradada está dada, en el dominio espacial por:

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

donde $h(x, y)$ es la representación espacial de la función de degradación, el símbolo "$*$" indica convolución y $\eta(x, y)$ es ruido aditivo.

- Restauración en presencia de ruido (filtros espaciales).
  Las fuentes principales de ruido en las imágenes digitales son durante la adquisición (digitalización) y/o durante la transmisión. El desempeño de los sensores de imágenes es afectado por una variedad de factores, como son las condiciones ambientales durante la adquisición de la imagen y por la calidad de los elementos de sensor.

  Cuando la única fuente de degradación en una imagen es ruido, la ecuación anterior se reescribe:

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

Existen diversos filtros útiles para eliminar el ruido:

1. El **filtro promedio aritmético** calcula el valor promedio de la imagen corrupta $g(x,y)$ en el área S_{xy}. El valor de la imagen restaurada en el punto (x,y) es simplemente el promedio aritmético calculado en esa vecindad:

$$\hat{f}(x,y) = \frac{1}{mn} \sum_{(s,t)\in S_{xy}} g(s,t)$$

2. Una imagen restaurada utilizando un **filtro promedio geométrico** está dada por la expresión:

$$\hat{f}(x,y) = \left[\prod_{(s,t)\in S_{xy}} g(s,t) \right]^{\frac{1}{mn}}$$

   En este caso, cada píxel restaurado está dado por el producto de los píxeles en la subimagen (ventana), elevado a la potencia $1/mn$.

3. **Filtro adaptativo** cuyo comportamiento cambia según las características de la imagen dentro de la región del filtro definida por una ventana rectangular S_{xy} de tamaño $m \times n$.

   Una expresión adaptativa para obtener $\hat{f}(x,y)$ basada en los supuestos anteriores, puede escribirse como:

$$\hat{f}(x,y) = g(x,y) - \frac{\sigma_\eta^2}{\sigma_L^2}[g(x,y) - m_L]$$

   donde σ_η^2 es la varianza del ruido que corrompe a $f(x,y)$ para formar $g(x,y)$, y m_L y σ_L^2 son la media y la varianza locales de los píxeles en la vecindad S_{xy}, respectivamente.

   La única cantidad que se necesita conocer o estimar es la varianza del ruido general, σ_η^2. Los otros parámetros se calculan de los píxeles en S_{xy} en cada posición (x,y) en donde el filtro está centrado.

- El filtro mediana adaptativo también trabaja en una vecindad S_{xy}. Sin embargo, a diferencia de los demás filtros, el filtro mediana adaptativo cambia (incrementa) el tamaño de S_{xy} durante su operación, dependiendo en ciertas condiciones que veremos más adelante. Recuerde que la salida del filtro es un solo valor que se utiliza para reemplazar el valor del píxel en la posición (x, y), del punto central particular de la vecindad S_{xy}. El algoritmo del filtro mediana adaptativo trabaja en dos niveles, denotados nivel A y nivel B como sigue:

  Nivel A: $A1 = z_{\text{med}} - z_{\text{min}}$
  $A2 = z_{\text{med}} - z_{\text{max}}$

  Si $A1 > 0$ AND $A2 < 0$, ve al nivel B.
  Si no, incrementa el tamaño de la ventana.
  Si el tamaño de la ventana $\leq S_{\text{max}}$, repita el nivel A.
  Si no, la salida es z_{xy}.

  Nivel B: $B1 = z_{xy} - z_{\text{min}}$
  $B2 = z_{xy} - z_{\text{max}}$

  Si $B1 > 0$ AND $B2 < 0$, la salida es z_{xy}.
  Si no, la salida es z_{med}.

  donde z_{min}, z_{max} y z_{med} son el valor mínimo, máximo y mediana de los niveles de gris de S_{xy}, respectivamente. z_{xy} es el nivel de gris en las coordenadas (x, y) y S_{max} es el valor de tamaño máximo permitido para la ventana S_{xy}.

- **Filtro Wiener**. Un método que incorpora ambos, la función de degradación y las características estadísticas del ruido, en el proceso de restauración es el llamado filtro Wiener. El método consiste en considerar imagen y ruido como un proceso aleatorio, y el objetivo es encontrar un estimador $\hat{f}$ de la imagen no-corrupta f de tal manera que el error promedio al cuadrado entre ellas sea mínimo. Este error está dado por:

$$e^2 = E\left\{(f - \hat{f})^2\right\}$$

  donde $E\{\cdot\}$ es el valor esperado (la esperanza) del argumento. Se asume que el ruido y la imagen no están correlacionados, que una u otra tienen

media igual a cero, y que los niveles de gris en la estimación son una función lineal de los niveles de la imagen degradada.

Basados en estas condiciones, la función de error mínima está dada en el dominio de la frecuencia por:

$$\hat{F}(u,v) = \left[\frac{1}{H(u,v)}\frac{|H(u,v)|^2}{|H(u,v)|^2 + S_\eta(u,v)/S_f(u,v)}\right] G(u,v)$$

donde $H(u,v)$ es la función de degradación, $H^*(u,v)$ es el conjugado complejo de $H(u,v)$, $|H(u,v)|^2$ es $H^*(u,v)H(u,v)$, $S_\eta(u,v)$ es $|N(u,v)|^2$ y representa el espectro de potencia del ruido, $S_f(u,v)$ es $|F(u,v)|^2$ y representa el espectro de potencia de la imagen no degradada.

El filtro que consiste en los términos dentro de los corchetes también se conoce como **filtro de error promedio mínimo al cuadrado**, o **filtro de error de mínimos cuadrados**.

3. Desarrollo

Resuelve los problemas de la lista siguiente y describe tu solución en cada inciso. Los incisos en donde únicamente tengas que desplegar imágenes no requieren de ninguna descripción.

1. Utiliza la imagen del circuito, genera ruido gaussiano aditivo con media cero y desviación estándar de 0.04. Filtra la imagen primero con un filtro promedio aritmético de tamaño 3×3, y luego filtra con un filtro promedio geométrico del mismo tamaño. Compáralos.

2. Utiliza nuevamente la imagen circuito, genera ruido gaussiano aditivo de media cero y desviación estándar de 0.04. Filtra primero con un filtro promedio aritmético de tamaño 7×7. Filtra ahora con un filtro geométrico del mismo tamaño. Finalmente, filtra con un filtro adaptativo para reducción de ruido del mismo tamaño que los anteriores. Compáralos.

3. Utiliza la imagen circuito, genera un ruido sal y pimienta aditivo con probabilidades $P_a = P_b = 0.25$. Filtra primero con un filtro mediana

de tamaño 7 × 7. Filtra ahora con un filtro mediana adaptativo con $S_{\max} = 7$. Compáralos.

4. Para la imagen lena con ruido aditivo de tipo gaussiano, encuentra el filtro de Wiener y restaura la imagen.

5. **E**ncuentra el filtro de Wiener y restaura una imagen lena que ha sido sometida a un proceso de pérdida de nitidez. La imagen con pérdida de nitidez se obtiene filtrando una imagen nítida y libre de ruido con un filtro paso bajo de tamaño 9x9 normalizado (filtro promedio ponderado).

6. Para una imagen lena a la que se le ha agregado ruido de tipo gaussiano y posteriormente ha perdido nitidez, encuentra el filtro de Wiener y restaura. Para obtener esta imagen degradada, primero se agrega el ruido de tipo gaussiano a la imagen original y luego se filtra con el mismo filtro paso bajo descrito en el punto 5.

7. Encuentra el filtro de Wiener y restaura una imagen lena que ha sido degradada por pérdida de nitidez y posteriormente se le ha agregado ruido. Para obtener esta imagen degradada se utiliza el filtro paso bajo descrito en el punto 5 y posteriormente se le agrega ruido.

4. Ejercicio 1

En el primer problema, se abordó la tarea de introducir y gestionar el ruido gaussiano en una imagen del circuito. Posteriormente, se aplicaron filtros promedio aritmético y geométrico para evaluar su efectividad en la reducción del ruido y comparar sus resultados.

4.1. Desarrollo

4.1.1. Carga de la Imagen

Se procedió a cargar la imagen en escala de grises del circuito utilizando la biblioteca `skimage`.

```
image_path = './ImagenesFiguras/circuitobn.png'
img = io.imread(image_path)
```

4.1.2. Añadir Ruido Gaussiano Aditivo

Se generó ruido gaussiano aditivo con media cero y desviación estándar de 0.04. Este ruido se sumó a la imagen original.

```
ruido_gauss = np.random.normal(0, 0.04, img.shape)
img_ruido = img + ruido_gauss
```

4.1.3. Filtro Promedio Aritmético

Se implementó una función para aplicar un filtro promedio aritmético de tamaño 3x3 a la imagen con ruido.

```
def filtro_promedio(img):
    # Implementación del filtro promedio aritmético
    # Vease en la seccion de Código
```

Luego, se aplicó este filtro a la imagen con ruido.

```
filtro_aritmetico = filtro_promedio(img_ruido)
```

4.1.4. Filtro Promedio Geométrico

Se implementó una función para aplicar un filtro promedio geométrico de tamaño 3x3 a la imagen con ruido.

```
def filtro_geom(img):
    # Implementación del filtro promedio geométrico
    # Vease en la seeción de Código
```

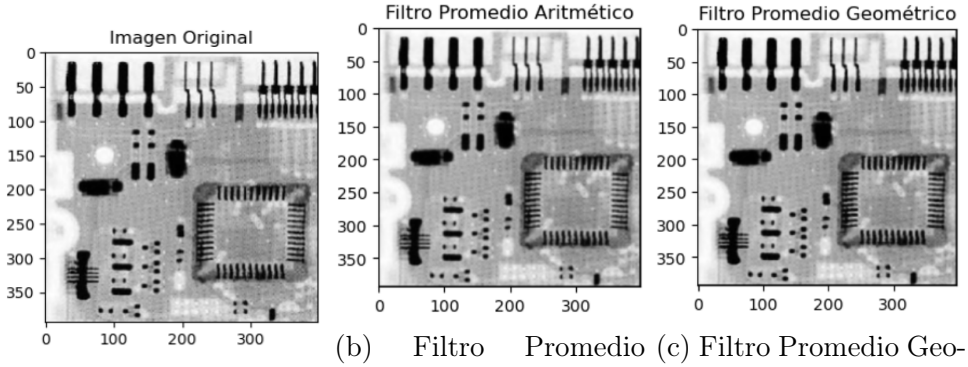
Luego, se aplicó este filtro a la imagen con ruido.

```
filtrado_geom = filtro_geom(img_ruido)
```

4.1.5. Visualización de Resultados

Se utilizó la biblioteca `matplotlib` para visualizar la imagen original, la imagen con ruido, la imagen filtrada con el filtro promedio aritmético y la imagen filtrada con el filtro promedio geométrico.

Este primer problema permitió observar cómo afectan los filtros promedio aritmético y geométrico a una imagen que ha sido degradada por ruido gaussiano aditivo. La comparación visual de los resultados ofrece una evaluación de la eficacia de ambos filtros en la reducción del ruido.



(a) Imagen Original (b) Filtro Promedio Aritmético (c) Filtro Promedio Geométrico

Figura 1: Resultados del Problema 1.

5. Ejercicio 2

En el segundo problema, se abordó la tarea de introducir y gestionar el ruido gaussiano aditivo en una imagen del circuito. Posteriormente, se aplicaron filtros promedio aritmético, geométrico y adaptativo para evaluar su efectividad en la reducción del ruido y comparar sus resultados.

5.1. Desarrollo

5.1.1. Carga de la Imagen y Añadir Ruido Gaussiano Aditivo

Se procedió a cargar la imagen en escala de grises del circuito y a generar ruido gaussiano aditivo con media cero y desviación estándar de 0.04.

```
image_path = './ImagenesFiguras/circuitobn.png'
img = io.imread(image_path)
ruido_gauss = np.random.normal(0, 0.04, img.shape)
img_ruido = img + ruido_gauss
```

5.1.2. Filtro Promedio Aritmético de Tamaño 7x7

Se aplicó un filtro promedio aritmético de tamaño 7x7 a la imagen con ruido.

```
filtro_aritmetico_7x7 = filtro_promedio(img_ruido, size=7)
```

5.1.3. Filtro Promedio Geométrico de Tamaño 7x7

Se aplicó un filtro promedio geométrico de tamaño 7x7 a la imagen con ruido.

```
filtrado_geom_7x7 = filtro_geom(img_ruido, size=7)
```

5.1.4. Filtro Adaptativo de Tamaño 7x7

Se aplicó un filtro adaptativo para reducción de ruido de tamaño 7x7 a la imagen con ruido.

```
filtrado_adaptativo_7x7 = filtro_adaptativo(img_ruido, size=7)
```

5.1.5. Visualización de Resultados

Se utilizó la biblioteca `matplotlib` para visualizar la imagen original y las imágenes filtradas con los tres tipos de filtros.

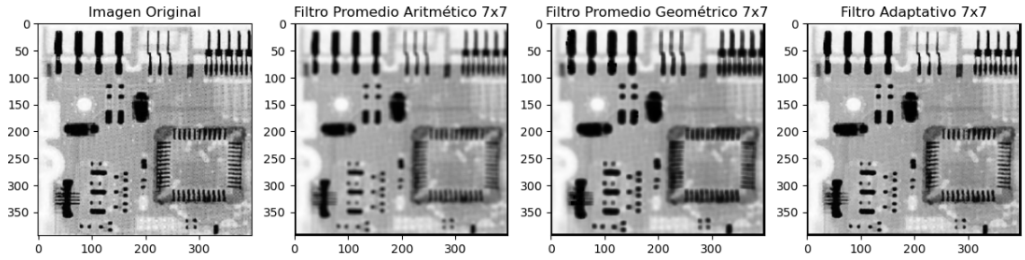


Figura 2: Resultados del Problema 2.

Este segundo problema permitió observar cómo afectan los filtros promedio aritmético, geométrico y adaptativo a una imagen que ha sido degradada por ruido gaussiano aditivo. La comparación visual de los resultados ofrece una evaluación preliminar de la eficacia de los tres filtros en la reducción del ruido.

6. Ejercicio 3

En el tercer problema, se abordó la generación y gestión de ruido de sal y pimienta en una imagen del circuito. Se aplicó un filtro de mediana adaptativo para reducir el impacto del ruido y se compararon los resultados.

6.1. Desarrollo

6.1.1. Carga de la Imagen y Agregar Ruido de Sal y Pimienta

Se cargó la imagen en escala de grises del circuito y se generó ruido de sal y pimienta con probabilidades $P_a = P_b = 0.25$.

```
image_path = './ImagenesFiguras/circuitobn.png'
img = io.imread(image_path)

def ruido_sal_pimienta(img, pa, pb):
    # ... Vease en la sección Código
    return img_ruido

img_sal_pimienta = ruido_sal_pimienta(img, 0.25, 0.25)
```

6.1.2. Filtro de Mediana Adaptativo de Tamaño 7x7

Se aplicó un filtro de mediana adaptativo de tamaño 7x7 a la imagen con ruido de sal y pimienta.

```
def filtro_mediana_adaptativo(img, s_max=7):
    # ... Vease en la sección código
    return result

filtro_mediana_adaptativo_7x7 =
filtro_mediana_adaptativo(img_sal_pimienta, s_max=7)
```

6.1.3. Visualización de Resultados

Se utilizó la biblioteca `matplotlib` para visualizar la imagen original con ruido de sal y pimienta y la imagen filtrada con el filtro de mediana adaptativo.

El tercer problema permitió explorar el efecto del ruido de sal y pimienta en una imagen y evaluar cómo un filtro de mediana adaptativo puede ser eficaz para reducir este tipo de ruido. La comparación de las imágenes proporciona información sobre la capacidad del filtro para conservar los detalles de la imagen original mientras elimina el ruido.

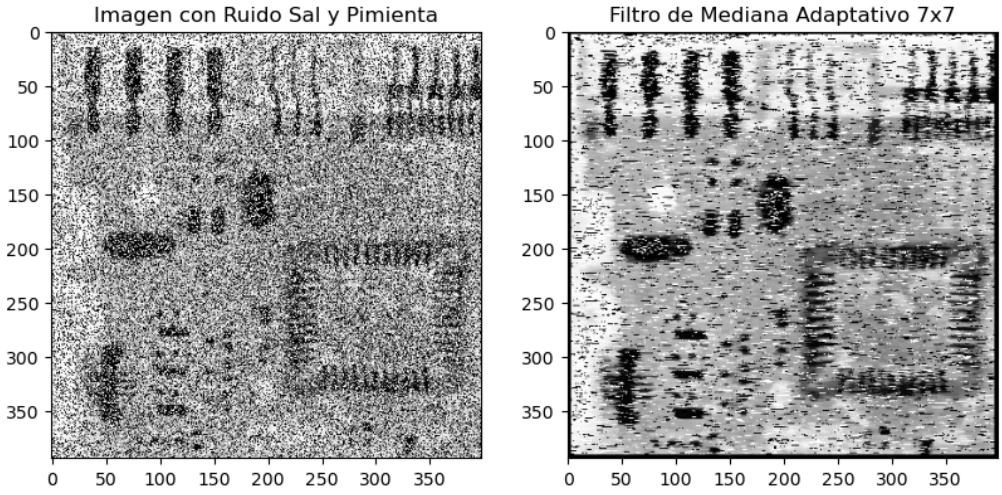


Figura 3: Resultados del Problema 3.

7. Ejercicio 4

El cuarto problema trata sobre la restauración de una imagen con ruido gaussiano utilizando el filtro de Wiener. Se aplicó el filtro a una imagen de Lena con ruido gaussiano y se compararon los resultados.

7.1. Desarrollo

7.1.1. Carga de la Imagen y Agregar Ruido Gaussiano

Se cargó la imagen en escala de grises de Lena y se le agregó ruido gaussiano aditivo.

```
image_path = './ImagenesFiguras/lenagbn.png'
img = io.imread(image_path, as_gray=True)

snr = 25  # Relación señal-ruido
noise = np.random.normal(0, 1/snr, img.shape)
img_noisy = img + noise
```

7.1.2. Definición del Kernel Gaussiano

Se definió un kernel gaussiano para simular el desenfoque.

```
kernel_size = 5
sigma = 1.0
gaussian_kernel = gaussian_kernel(kernel_size, sigma)
```

7.1.3. Aplicación del Filtro de Wiener

Se aplicó el filtro de Wiener para restaurar la imagen con ruido gaussiano.

```
restored_img = wiener_filter(img_noisy, gaussian_kernel, snr)
```

7.1.4. Visualización de Resultados

Se utilizó la biblioteca `matplotlib` para visualizar la imagen original con ruido gaussiano, la imagen restaurada con el filtro de Wiener y el kernel gaussiano.

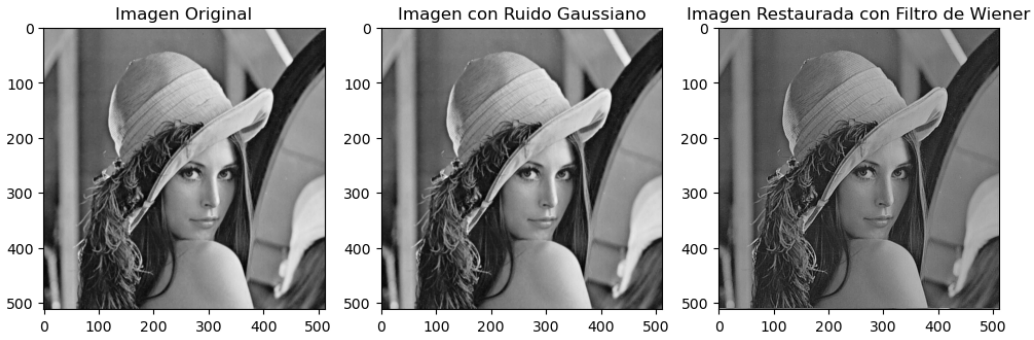


Figura 4: Resultados del Problema 4.

El cuarto problema permitió explorar el uso del filtro de Wiener para restaurar una imagen con ruido gaussiano. La comparación visual de la imagen original, la imagen con ruido y la imagen restaurada proporciona información sobre la efectividad del filtro en la reducción del ruido y la mejora de la calidad de la imagen.

8. Ejercicio 5

El quinto problema se centra en la restauración de una imagen que ha sufrido pérdida de nitidez. Se aplicó el filtro de Wiener a una imagen de Lena degradada por un filtro paso bajo y se compararon los resultados.

8.1. Desarrollo

8.1.1. Carga de la Imagen y Simulación de Pérdida de Nitidez

Se cargó la imagen en escala de grises de Lena y se simuló la pérdida de nitidez aplicando un filtro paso bajo de tamaño 9x9.

```
image_path = './ImagenesFiguras/lenagbn.png'
img_original = io.imread(image_path, as_gray=True)

kernel_size = 9
blur_kernel = weighted_average_filter(kernel_size)
img_blurred = wiener_filter(img_original, blur_kernel, 1)
```

8.1.2. Aplicación del Filtro de Wiener para Restaurar la Imagen

Se aplicó el filtro de Wiener para restaurar la imagen de Lena que había sufrido pérdida de nitidez.

```
snr = 25  # Relación señal-ruido
restored_img = wiener_filter(img_blurred, blur_kernel, snr)
```

8.1.3. Visualización de Resultados

Se utilizó la biblioteca `matplotlib` para visualizar la imagen original, la imagen degradada con pérdida de nitidez y la imagen restaurada con el filtro de Wiener.

Figura 5: Resultados del Problema 5.

El quinto problema proporcionó el uso del filtro de Wiener en la restauración de imágenes que han experimentado pérdida de nitidez. La comparación de la imagen original, la imagen degradada y la imagen restaurada destaca la eficacia del filtro en la mejora de la nitidez y la reducción del ruido.

9. Problemas 6 y 7

Estos problemas abordan la restauración de una imagen de Lena que ha sido afectada por ruido gaussiano y pérdida de nitidez. En el caso del Problema 6, el ruido gaussiano se agrega a la imagen original antes de aplicar el filtro paso bajo para simular la pérdida de nitidez. En el Problema 7, la pérdida de nitidez y el ruido gaussiano se aplican secuencialmente.

A continuación se muestra el código en Python utilizado para simular estos procesos y los resultados obtenidos:

```
#(Código para cargar la imagen, agregar ruido gaussiano, simular
pérdida de nitidez y aplicar filtro de Wiener)

# Visualizar las imágenes
plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)
plt.imshow(img_original, cmap='gray')
plt.title('Imagen Original')

plt.subplot(1, 3, 2)
plt.imshow(img_noisy, cmap='gray')
plt.title('Imagen Degradada con Pérdida de Nitidez y Ruido Gaussiano')

plt.subplot(1, 3, 3)
plt.imshow(restored_img, cmap='gray')
plt.title('Imagen Restaurada con Filtro de Wiener')

# ... Vease en la sección Código
```

En estos problemas, se busca restaurar la imagen de Lena que ha experimentado ruido gaussiano y pérdida de nitidez. Se utiliza el filtro de Wiener para mejorar la calidad de la imagen degradada.

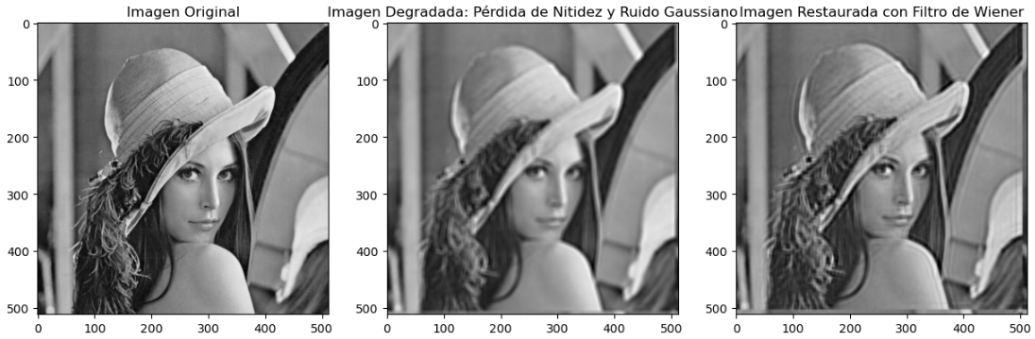


Figura 6: Resultados del Problema 6.

Figura 7: Resultados del Problema 7.

10. Código

El código fuente y los recursos utilizados en esta práctica se adjuntan en la entrega de esta práctica en un archivo comprimido llamado `Practica5_Hernandez_Sanchez_Oscar_Jose.zip`.

Además tanto el código como los recursos se encuentran disponibles también en el siguiente repositorio de Github:

`https://github.com/oscarhs123/Practica-5-PDI.git`

Tanto en el repositorio como en el archivo comprimido `Practica5_Hernandez_Sanchez_Oscar_Jose.zip`, se encuentra un archivo de Jupyter Notebook llamado `Practica5_Hernandez_Sanchez_Oscar.ipynb`, que contiene todos los pasos detallados de esta práctica en forma de código

ejecutable. Además, hay una carpeta llamada `ImagenesFiguras` que contiene las imágenes utilizadas durante la práctica.

A continuación, se proporciona una descripción de los archivos y recursos disponibles en el repositorio y en el `.zip`:

- `Practica5_Hernandez_Sanchez_Oscar.ipynb`: Este archivo de Jupyter Notebook contiene los pasos de la práctica implementados en código Python. Cada paso está explicado y acompañado de los resultados obtenidos.
- `ImagenesFiguras/`: Esta carpeta contiene las imágenes utilizadas en la práctica. Estas imágenes se emplean para realizar las manipulaciones y demostraciones necesarias en los diferentes pasos.

Para acceder al código, se debe tener instalado `jupyter-notebook` en nuestra computadora, además de las bibliotecas `skimage`, `matplotlib`, `numpy` y `random` para su correcta ejecución.

Una vez instalados, el código se puede visualizar en el kernel gráfico de Jupyter.

1. **Instalación de Jupyter Notebook:**

   Para instalar Jupyter Notebook, puedes utilizar el administrador de paquetes de Python, pip, apt o dnf. Ejemplo:

   ```
   pip install jupyter
   ```

   Esto instala Jupyter Notebook en nuestro sistema.

2. **Instalación de Bibliotecas:**

   Además de Jupyter Notebook, necesita instalar las bibliotecas `skimage`, `matplotlib`, `numpy` y `random` para ejecutar el código del archivo.

   La biblioteca `random` es parte de la biblioteca estándar de Python, por lo que no se necesita instalar por separado.

3. **Ejecución del Jupyter Notebook:**

   Una vez instalado Jupyter Notebook y las bibliotecas necesarias, se debe abrir una terminal, navegar al directorio donde se encuentra el archivo `Practica5_Hernandez_Sanchez_Oscar.ipynb` y ejecutar el siguiente comando:

```
jupyter-notebook
```

Esto abrirá una ventana del navegador con el entorno de Jupyter Notebook. Desde allí, seleccionaremos el archivo de la práctica y lo abriremos para ver y ejecutar el código.

11. Conclusiones

En el desarrollo de los problemas planteados, se han aplicado diversas técnicas de procesamiento de imágenes para abordar situaciones específicas. A continuación, se presentan algunas conclusiones derivadas de cada problema:

1. En el **Problema 1**, se observó la diferencia entre el filtro promedio aritmético y geométrico al enfrentarse al ruido gaussiano aditivo. La comparación visual permitió entender cómo cada filtro afecta la imagen resultante.

2. En el **Problema 2**, se exploraron filtros aritméticos, geométricos y adaptativos para reducción de ruido gaussiano. La comparación destacó las fortalezas y debilidades de cada enfoque en términos de preservación de detalles y suavización.

3. El **Problema 3** introdujo el ruido sal y pimienta, y se aplicaron filtros de mediana y mediana adaptativa. La adaptabilidad del segundo filtro mostró ser efectiva al manejar este tipo de ruido.

4. En el **Problema 4**, se empleó el filtro de Wiener para restaurar una imagen con ruido gaussiano aditivo. Este enfoque demostró ser eficaz al reducir el impacto del ruido y mejorar la calidad de la imagen.

5. El **Problema 5** abordó la pérdida de nitidez utilizando un filtro paso bajo de tamaño 9x9. La restauración con el filtro de Wiener permitió recuperar la nitidez de la imagen original.

6. En el **Problema 6**, se combinaron ruido gaussiano y pérdida de nitidez. La aplicación secuencial del ruido y el filtro paso bajo, seguida por el filtro de Wiener, proporcionó una restauración efectiva.

7. Finalmente, el **Problema 7** integró pérdida de nitidez y ruido gaussiano. La restauración con el filtro de Wiener después de aplicar el filtro paso bajo y agregar ruido demostró ser un enfoque sólido.

Estas conclusiones reflejan la aplicación práctica de diversos conceptos de procesamiento de imágenes y resaltan la importancia de seleccionar y combinar adecuadamente las técnicas según el tipo de degradación presente en las imágenes.

12. Referencias

1. Gonzalez, R., Woods, R., *Digital Image Processing*, Prentice Hall, 2008.