PRÁCTICA 4

"Simulación y Filtrado de Ruidos en Imágenes"

Objetivo: simulación y filtrado de ruidos en imágenes; evaluación de distintos tipos y opciones de filtrado.

Instrucciones Matlab:

- `randn`: obtiene una matriz cuyos valores son generados por una distribución gaussiana de media nula y varianza 1 ($N(0,1)$).

  Utilizando esta instrucción se puede obtener una matriz cuyos valores se ajusten a una distribución gaussiana de media μ y varianza σ^2 ($N(\mu,\sigma^2)$):

$$N(0,1) = \frac{N(\mu,\sigma^2) - \mu}{\sigma} \quad \rightarrow \quad N(\mu,\sigma^2) = \mu + \sigma N(0,1)$$

- `rand`: permite obtiene una matriz cuyos valores son generados por una distribución uniforme en el rango (0,1).

- `fspecial`: para crear máscaras de aplicación en filtrado lineal.

- `medfilt2`: para filtrar una imagen digital mediante filtro de la mediana.

- `ordfilt2`: para filtrar una imagen digital mediante cualquier filtro de orden.

- `stdfilt`: para obtener la desviación estándar en cada píxel de una matriz, calculada en una vencidad especificada por parámetro.

- `mean` ; `median`; `std`: para calcular la media, mediana y desviación estándar.

La práctica se ha dividido en tres partes:

1. **Primera parte:** simulación de ruidos tipo gaussiano y *sal y pimienta*.
2. **Segunda parte:** implementación de filtros gaussiano, mediana y adaptativo.
3. **Tercera parte:** evaluación de eficiencia de filtros gaussiano, mediana y adaptativo.

PRIMERA PARTE: Simulación de ruidos.

1. Lee la imagen "P4.tif", que es una imagen en escala de gris.

2. Corrompe la imagen anterior con ruido de tipo de sal y pimienta y de tipo gaussiano para generar, tal y como se describe a continuación:

   a. Sal y pimienta, con p = 0.9 y q = 0.95 (ver ecuación). La imagen corrompida con este ruido (imagen A) se puede calcular de la siguiente manera:

   $$A(i,j) = \begin{cases} I(i,j) & x < p \\ 0 & p \le x < q \\ 255 & q \le x < 1 \end{cases}$$ siendo I la imagen original sin corromper, x una variable aleatoria uniforme de rango (0,1), q-p el porcentaje de píxeles con ruido de tipo pimienta y 1-q el porcentaje de píxeles con ruido de tipo sal.

   b. Gaussiano de media nula y desviación típica 10.

Nota: *La matriz correspondiente a la imagen original es una matriz de elementos de tipo* `uint8`*. Tienes que realizar las operaciones en formato* `double` *para luego volver a convertir a tipo* `uint8`*, que es el tipo de datos de la imagen resultante.*

3. Visualiza las imágenes ruidosas A y B. Representa en un mismo gráfico la variación de los niveles de gris a lo largo de la línea horizontal central para la imagen original, para la imagen A con ruido de tipo sal y pimienta, y para la imagen B con ruido gaussiano. Para ello emplea la función `plot` (`hold on` para mantener la misma gráfica). Observa las distintas distribuciones del ruido.

Nota: *la sentencia* `title` *permite añadir títulos a las ventanas figure, y así poder identificarlas mejor.*

SEGUNDA PARTE: Implementación de filtros gaussiano, mediana y adaptativo.

Esta segunda parte de la práctica se debe realizar sobre la imagen corrompida con ruido sal y pimienta generada en la primera parte de la práctica

Aplica los siguientes filtros, visualiza las imágenes filtradas y discuta los resultados obtenidos:

a. Un filtro gaussiano con W = 5 σ, siendo σ la desviación típica del filtro y W el tamaño del filtro (considera un valor W=5). Para ello:

   - Crea la máscara del filtro gaussiano, implementada en Código Matlab (ver documentación teórica - Tema 3) y utilizando función `fspecial`. Comprueba que se obtienen los mismos resultados.
   - Utiliza la función `imfilter` para realizar la convolución.

b. Un filtro de la mediana considerando un entorno de vecindad de 5 x 5. Para ello, implementa la función (ver observación):

   ```
   Ifiltrada = Funcion_FiltroMediana (I, NumFilVent, NumColVent)
   ```

   Comprueba que se obtiene la misma imagen de salida que genera la función de Matlab `medfilt2`.

c. Un filtro adaptativo que actúe en un entorno de vecindad 5x5. Para ello, implementa la función (ver observación):

```
Ifiltrada = Funcion_FiltAdapt (I, NumFilVent, NumColVent, VarRuido)
```

Observación: En las funciones anteriores, `I` es la imagen de entrada, `NumFilVent` y `NumColVent` son el número de filas y columnas de la ventana de vecindad considerada, e `Ifiltrada es` la imagen de salida filtrada. En el caso de la función `Funcion_FiltAdapt`, `VarRuido` es la varianza del ruido presente en la imagen; para hacerla más eficiente, utilizar en su implementación las funciones `imfilter` y `stdfilt`. Por otra parte, se ha de considerar que las ventanas de vecindad tienen un número impar de filas y columnas. Además, se debe contemplar la asignación de un valor adecuado a todos aquellos puntos del entorno de vecindad que no existen en la imagen (para ello, puede utilizarse la función de matlab `padarray`).

TERCERA PARTE: Evaluación de eficiencia de filtros gaussiano, mediana y adaptativo.

Esta tercera parte de la práctica se debe realizar sobre la imagen "P4.tif"

1. A partir de la imagen "P4.tif", genera tres imágenes con ruido gaussiano de media nula y desviaciones típicas, por cada imagen generada, de 5, 10 y 35.

2. Filtra cada una de las imágenes ruidosas anteriores con filtros de tipo gaussiano, de tipo mediana y adaptativo, considerando tamaños 3x3 y 7x7 para cada filtro. Visualiza las distintas imágenes ruidosas y filtradas.

3. Evalúa la eficiencia de cada proceso de filtrado midiendo la relación señal-ruido (ISNR), definida como:

$$\mathrm{ISNR}_{\mathrm{dB}} = 10\log_{10}\frac{\sum_{i=1}^{M}\sum_{j=1}^{N}\left(\mathbf{I}(i,j)-\mathbf{G}(i,j)\right)^2}{\sum_{i=1}^{M}\sum_{j=1}^{N}\left(\mathbf{I}(i,j)-\mathbf{I_e}(i,j)\right)^2}$$

   donde I es la imagen original sin ruido, G la imagen ruidosaa, e I_e es la imagen filtrada.

4. A partir de los resultados obtenidos, realiza un informe de conclusiones.

Aplicación filtro temporal

5. Genera, a partir de la imagen inicial, 10 imágenes ruidosas con ruido blanco gaussiano y desviación típica 35. Visualiza una de estas imágenes.

6. Aplica un promediado a estas imágenes y observa la imagen resultante.

Nota: *las operaciones se deben realizar en tipo double.*