Práctica 2
Filtrado de una imagen
Reducción de ruido

Robótica y Visión por Computador
Ing. Telecomunicaciones

Universidad Miguel Hernández

Objetivo de la práctica:

El objetivo de la práctica es realizar un estudio comparativo de diversos algoritmos de filtrado para la reducción del ruido presente en una imagen. Las imágenes utilizadas son en escala de grises de 256 filas por 256 columnas, y se encuentran almacenadas en .jpg.

Se define como ruido toda aquella variación del nivel de gris que sufre un pixel no debida a la aportación lumínica de la escena.

Los ruidos se catalogan en:

- **Correlados**: cuando la variación del nivel de gris depende de la posición espacial del pixel. Un ruido correlado que se presenta frecuentemente en la adquisición de imágenes con integración de campo es el desplazamiento entre filas pares e impares en la imagen.

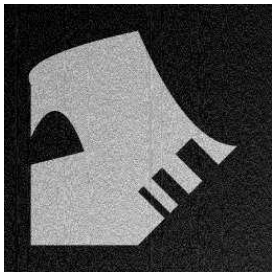
Imagen inicial

Imagen con filas pares desplazadas 10 píxels.

- **No correlados**: cuando la variación del nivel de gris no depende de la posición espacial del pixel. Los ruidos no correlados más frecuentes son:

    - El **ruido gaussiano**. La distribución del ruido se asemeja a una distribución gaussiana de una determinada media y varianza. En las siguientes imágenes se muestran dos ejemplos generados a partir de la anterior imagen inicial y afectados por distintas magnitudes de ruido gaussiano.

Ruido gussiano (M=0, s=5)



Ruido gaussiano (M=0, s=25).

- El **ruido aleatorio** o, también denominado, "*sal y pimienta*". El número de pixels afectados por el ruido y su intensidad son las variables aleatorias independientes entre sí. Las siguientes imágenes muestran dos ejemplos de ruido aleatorio a partir de la imagen inicial.

Cuando se quiere evaluar el grado de degradación de una imagen o el resultado de un algoritmo, es corriente emplear índices de error. Entre ellos, cabe destacar los índices cuadráticos de error entre dos imágenes. Así, se definen:

Error cuadrático total:

$$E_{CT} = \frac{1}{N \cdot M} \cdot \sum_{y=0}^{N-1} \sum_{x=0}^{M-1} \left[f(x,y) - f_r(x,y) \right]^2$$

siendo:

- f(x,y) la imagen a analizar.
- f_r(x,y) la imagen de base.

En ocasiones, se expresa en tantos por ciento.

Error cuadrático normalizado:

$$E_{CN} = \frac{\sum_{y=0}^{N-1} \sum_{x=0}^{M-1} \left[f(x,y) - f_r(x,y) \right]^2}{\sum_{y=0}^{N-1} \sum_{x=0}^{M-1} \left[f_r(x,y) \right]^2}$$

que también se suele expresar en tantos por ciento.

<u>Realización de la práctica:</u>

1. En primer lugar se debe abrir el workspace creado en la práctica 0 o bien generar uno nuevo tal y como se explicó en dicha práctica.
2. Añadimos un nuevo proyecto al workspace, que llamaremos **practica2**
   a. Si estamos trabajando sobre el workspace anterior, los proyectos `cv.dsp, cvaux.dsp` y `highgui.dsp` ya están añadidos, y lo único que debemos hacer es establecer las dependencias y los settings del nuevo proyecto creado.
   b. Si hemos creado un proyecto nuevo, se deben añadir los proyectos `cv.dsp, cvaux.dsp` y `highgui.dsp` tal y como se explicó en la práctica 0 y luego establecer la configuración adecuada.

3. Desde el explorador de Windows, creamos un directorio `\images` dentro de la carpeta `\practica2` y colocamos las imágenes de la práctica. Las imágenes utilizadas en esta práctica son las siguientes:
    a. *Inicial.jpg*. Imagen sintética sin ruido
    b. *Ruido_1.jpg*. Imagen afectada con ruido gaussiano de sigma baja.
    c. *Ruido_2.jpg*. Imagen afectada con ruido gaussiano de sigma alta.
    d. *Ruido_3.jpg*. Imagen afectada con ruido aleatorio en pocos pixeles.
    e. *Ruido_4.jpg*. Imagen afectada con ruido aleatorio en muchos pixeles.
    f. *Ruido_5.jpg*. Imagen afectada con ruido aleatorio y ruido gaussiano.
    g. *Ruido_6.jpg, ruido_7.jpg, ruido_8.jpg, ruido_9.jpg*. Imágenes afectadas con ruido gaussiano de la misma sigma.
4. Escribir el código que realice las siguientes acciones:
    a. Abrir las imágenes *inicial.jpg, ruido_x.jpg*
    b. Declarar una imagen resultado *(res)*
    c. Aplicar los diferentes filtros existentes en OpenCV a las imágenes con ruido (*ruido_1.jpg a ruido_5.jpg*). Para ello se usará la función `cvSmooth`, cuya prototipo se muestra a continuación. (ver ayuda en la referencia de cv)

```
void cvSmooth( const CvArr* src, CvArr* dst, int
smoothtype=CV_GAUSSIAN,int param1=3, int param2=0, double
param3=0 );
```

    d. Para cada filtro se debe obtener el error cuadrático total y el total normalizado, mostrándolos en la pantalla de consola.
    e. Utilizar las imágenes *ruido_6.jpg a ruido_9.jpg* para aplicar un filtro temporal basado en la adición de las 4 imágenes. Obtener los errores para este filtro.
    f. Liberar memoria.