Práctica 2: Implementación de algoritmos

Grado en Inteligencia Artificial

La práctica consiste en implementar operaciones de procesado de imagen vistas en clase con el objetivo de comprender su funcionamiento y sus efectos sobre las imagenes de entrada.

- La práctica deberá realizarse de forma **individual**.
- Las soluciones propuestas se codificarán en Python 3.x con las bibliotecas de procesamiento **Numpy, SciPy, Scikit-image y Scikit-learn**.
- No está permitido el uso de versiones de Python 2.x.
- **No está permitido** el uso de funciones de tratamiento de imágenes existentes en los entornos de desarrollo en la implementación de las operaciones. En especial se prohíbe usar las funciones que implementan las mismas operaciones o cualquier parte sustancial de ellas.
- **Está permitido** el uso de cualquier librería para leer, escribir y visualizar imágenes, para transformar las imágenes al formato de entrada de las operaciones, o para cubrir cualquier otra necesidad en el código de pruebas. También está permitido usar cualquier función no relacionada con el procesado de imagen (estadística, ordenación, indexación de matrices, etc.) tanto en el código de prueba como en las operaciones implementadas.

1. Entrega y evaluación

- El alumnado **deberá entregar en el Campus Virtual (Moodle de la UDC), en un repositorio habilitado para tal fin**:
  - Los alumnos que presenten códigos con indicios de plagio (en cualquiera de las operaciones y con respecto a otras prácticas presentadas este curso o en cursos anteriores) obtendrán una calificación de Suspenso (calificado con 0.0) en la práctica, independientemente de la nota que pudiera merecer la calidad de la práctica.
- La **evaluación de la práctica se realizará mediante defensa ante el profesor de prácticas**.
  - En la defensa se facilitarán imágenes de prueba para varias de las funciones implementadas. El alumnado debe ser capaz de leer las imágenes de entrada, adaptarlas a la especificación de las funciones, usarlas como entrada y visualizar correctamente la salida obtenida. El resultado obtenido debe ser correcto, así como tener capacidad para analizar y comprender por qué es así o no.

2. Operaciones a implementar

2.1. Filtrado espacial: suavizado

- Implementar una función que calcule un **kernel Gaussiano unidimensional** con σ dado.

```
kernel = gaussianFilterSpatial1D (sigma)
```

`sigma:` Parámetro σ de entrada.

`kernel:` Vector `1xN` con el kernel de salida, teniendo en cuenta que:

- El centro $x = 0$ de la Gaussiana está en la posición $\lfloor N/2 \rfloor + 1$.
- N se calcula a partir de σ como $N = 2\lceil 3\sigma \rceil + 1$.

- Implementar una función que permita realizar un **suavizado Gaussiano bidimensional** usando un filtro $N \times N$ de parámetro σ, donde N se calcula igual que en la función anterior.

```
outImage = ApplyGaussianFilterSpatial (inImage, sigma)
```

`inImage, outImage, sigma:` ...

Nota. Como el filtro Gaussiano es lineal y separable implementamos este suavizado simplemente convolucionando la imagen, primero, con un kernel Gaussiano unidimensional $1 \times N$ y, luego, convolucionando el resultado con el kernel transpuesto $N \times 1$.

- Implementar una función que calcule un filtro gaussiano en frecuencia, de tamaño $N \times M$ y un σ dado.

```
outFilter = gaussianFilterFrec (inImage, sigma)
```

`outFilter,` $N \times M$`, sigma:` ...

- Implementar una función que aplique el filtro gaussiano en frecuencia anterior sobre una imagen, con el filtro del tamaño de la imagen y un σ dado.

```
outFilter = ApplyGaussianFilterFrec (inImage, sigma)
```

`outFilter, sigma:` ...