AULA PRÁTICA 2. *PRACTICAL CLASS 2.*

1. CONVOLUÇÃO. *CONVOLUTION.*

1.1 Carregar e visualizar a imagem ***marilyn.tif***. Construir um filtro 3×3 arbitrário e executar a convolução desse filtro com a imagem. *Upload and preview image **marilyn.tif**. Build an arbitrary 3×3 filter and convolve that filter with the image.*

```
conv = ndimage.convolve(F.astype(float), filtro_k, mode='constant', cval=0.0)
```

NOTA: no python importar a ferramenta *ndimage* da biblioteca *scipy*. *NOTE: in python import ndimage tool from scipy library.*

1.2 Visualizar o resultado do ponto anterior, numa janela que contenha as imagens inicial e resultante. *Display the result of the previous point in a window containing the initial and resulting images.*

2. FILTROS PASSA-BAIXA E PASSA-ALTA. *LOW-PASS and HIGH-PASS FILTERS.*

2.1. Carregar e ver as imagens ***noisy_gauss.tif*** (ruído gaussiano) e ***noisy_sp.tif*** (ruído impulsivo). A estas imagens aplicar os filtros de suavização da média, gauss e mediana, com *kernels* de diversas dimensões. Qual dos filtros exibe resultados mais aceitáveis em cada um dos casos? *Upload and view images **noisy_gauss.tif** (gaussian noise) and **noisy_sp.tif** (impulsive noise). To these images apply the smoothing filters of the mean, gauss and median, with kernels of diverse dimensions. Which of the filters shows the most acceptable results in each case?*

```
#%% Filtro da média
mean_denoised = ndimage.convolve(F, h, mode='constant', cval=0.0)

#%% Filtro de Gauss
gauss_denoised = ndimage.filters.gaussian_filter(F, 1)

#%% Filtro da Mediana
median_denoised = ndimage.filters.median_filter(F, 3)
```

2.2. À imagem **marilyn.tif**, aplique os operadores de gradiente de Roberts, Prewitt e Sobel (com *kernels* de 3×3 segundo as direcções horizontal e vertical) e visualize o resultado. *Apply the Roberts, Prewitt and Sobel gradient operators to the image **marilyn.tif** (with 3 × 3 kernels in the horizontal and vertical directions) and view the result.*

```
# Filtro de Sobel
from scipy import ndimage
sx = ndimage.sobel(F, axis=0, mode='constant')
sy = ndimage.sobel(F, axis=1, mode='constant')
sob = np.hypot(sx, sy)

# Filtro de Prewitt
sx = ndimage.prewitt(F, axis=0, mode='constant')
sy = ndimage.prewitt(F, axis=1, mode='constant')
prw = np.hypot(sx, sy)
```

3. UNSHARP.

Executar a operação de *uncharp* sobre a imagem **marylin_unsharp.tif** e ver o resultado. *Run uncharp on **marylin_unsharp.tif** image and see the result.*

4. FILTRAGEM DE RUÍDO *BANDING*. *FILTERING OF BANDING NOISE.*

Executar as operações seguintes para filtrar o ruído não aleatório da imagem **Stripping_Noise.tif**. *Execute the following operations to filter the non-random noise of the **Stripping_Noise.tif** image.*

4.1. Aplicar filtro passa-baixa da média aritmética. Aplicar filtro passa-alta determinado a partir do filtro anterior. Somar as duas imagens resultantes. O que se obtém? *Apply low pass filter of arithmetic mean. Apply high pass filter determined from previous filter. Add the two resulting images. What do you get?*

4.2. Filtrar ruído banding. *Banding noise filtering.*

4.2.1. Aplicar filtro passa-baixa (PB1) com um *kernel* geometricamente semelhante ao ruído a eliminar e contendo este. Obter a imagem passa-alta (PA1) correspondente. *Apply low pass filter (PB1) with a kernel geometrically like the noise to be eliminated and containing this. Obtain the corresponding high-pass filtered image (PA1).*

4.2.2. Aplicar filtro passa-baixa (PB2) com um elemento estruturante geometricamente semelhante ao ruído a eliminar e incluído neste. Obter a imagem passa-alta (PA2) correspondente. *Apply low pass filter (PB2) with a structuring element geometrically like the noise to be eliminated and included in it. Obtain the corresponding high-pass filtered image (PA2).*

4.2.3. Somar o filtro passa-baixa (PB1) com o filtro passa-alta (PA2) e verificar o resultado da imagem resultante. *Add the low pass filter (PB1) with the high pass filter (PA2) and check the result of the resulting image.*