# Assignment : Filtering and Histograms
# Signal and Image Processing

February 8, 2025

You can work on this assignment and submit your solution (report and code) as a GROUP. This assignment counts towards your grade and have to be submitted in order to pass the course. You must follow the report guidelines found in `guidelines.pdf`. The page limit for this assignment is **12 pages** including everything, i.e. illustrations and code snippets.

1. **Pixel-wise contrast enhancement**

   1.1. (1 point) Write your own Python function that implements the gamma transform of a gray scale image. Illustrate the function by applying it to an image of your choice. Try different values of gamma and notice how the image details become more or less visible.
   **Deliverables:** Include your function as a code snippet in the report, a figure illustrating your results, and a comment about the effect of changing gamma.

   1.2. (1 point) Apply the grayscale gamma-correction function from the previous question to the color image 'autumn.tif' by doing so on each of the RGB component separately.
   **Deliverables:** Include a code snippet in the report showing your solution and plot the original image and the result of this procedure side-by-side.

   1.3. (1 point) Apply the grayscale gamma-correction function from the previous question to the color image 'autumn.tif' by first converting to HSV color representation (see `skimage.color.rgb2hsv`) and apply the gamma correction to the v-channel, and finally convert back to RGB representation (see `skimage.color.hsv2rgb`).
   **Deliverables:** Plot the original image and the result of this procedure side-by-side. Compare the result with your solution to Question 1.2, which of the two approaches provide the best result?

2. **Reverb Convolution**

   2.1. (1 point) Load `laugh2.wav` into a numpy array (hint, you can use the soundfile package, see notes on absalon `soundfiles_example.py`). Plot the channels of the signal as different curves on the same plot.
   **Deliverables:** Include the plot in your report. Write a sentence describing what each curve represents. Write a sentence describing what the sample rate represents.

   2.2. (1 point) Reverb is commonly done in the audio industry to change the quality of a sound. In practice it is done by convolving a soundbyte with an impulse

signal. Choose one of the soundbytes in the `Sound samples` folder and an impulse from both the `Sound impulses/Claps` and `Sound impulses/Splashes` folders. Convolve the soundbyte with each of the impulses, play the output and plot the original signal and the response signals. For this it is recommended to use the `scipy.signal.convolve` function.
**Deliverables:** In 2-3 sentences describe what reverb is and the effect it had on the sound. Include the plots of original signal and the response signal in your report.

2.3. (1 points) **Deliverables:** The altered sound file is longer than the original. Explain why in 1-2 sentences.

## 3. Image filtering and enhancement

3.1. (1 point) Compare the effect of mean and median filtering on a noisy version of the image *eight.tif* for salt and pepper as well as gaussian noise. You can use the function `skimage.util.random_noise` for doing this.
**Deliverables:** Include images as illustration. What is the visual effect of increasing the kernel size $N$ ? Store and plot the different computational times obtained for N=1 to N=25 and each time for 100 executions (use Pythons `timeit` package). Comment on your results.

3.2. (1 point) Consider a Gaussian filter of fixed standard deviation $\sigma = 5$ and filter the image with increasing value of kernel size $N$.
**Deliverables:** Include images as illustration. What do you observe when $N$ is large enough and how can it be explained?

3.3. (1 point) Experiment with filters of increasing $\sigma$'s, choosing at each iteration $N = 3\sigma$ for the filter kernel size $N \times N$.
**Deliverables:** Include images as illustration and comment on the effect with respect to denoising versus sharpness of the image.

## 4. Histogram-based processing

4.1. (1 point) Implement a Python function that, for a given histogram of a grayscale image, computes its cumulative distribution function CDF (you may use the *numpy.cumsum* function and normalize the expression).
**Deliverables:** Include the code for your Python function as a code snippet in the report. Display the result of computing the CDF for the image *pout.tif* using your function. What do the regions of fast increase of the function correspond to? And what about its flat regions?

4.2. (1 point) Given an image $I$ and its CDF $C$, write a Python function that computes the floating-point image $C(I)$ such that the intensity at each pixel $(x, y)$ is $C(I(x, y))$.
**Deliverables:** Include the code for your Python function. Show the result of applying this function to the *pout.tif* image.

4.3. (1 point) The CDF is in general not invertible – why ? To overcome the problem of non-invertibility for histogram matching, we can consider instead a pseudo-inverse. For any CDF function $C(s)$ defined on the integer set $s \in \{0, .., 255\}$, we define its pseudo-inverse $C^{(-1)}$ as follows :

$$C^{(-1)}(l) = \min\{s \mid C(s) \geq l\} ,$$

where $l \in [0, 1]$. Write a Python function that computes the pseudo-inverse of any given CDF (you may use *numpy.min* or *numpy.where* function).
**Deliverables:** Include your answer to the first question and the code for your Python function as a code snippet.

4.4. (1 point) Assume we have two gray scale images $I_1$ and $I_2$ and the corresponding CDFs $C_1$ and $C_2$, then histogram matching can be described by the mapping

$$J(x, y) = C_2^{-1}\left(C_1(I_1(x, y))\right) \ .$$

Based on this equation and on the previous questions, implement your own Python function to perform histogram matching between two images.
**Deliverables:** Include the code for your Python function and show the two input and resulting images. Also plot and compare the cumulative histograms of the original image, the target and the one obtained by histogram matching.