April 9, 2025

Project 1 - Bilateral Filter, BM3D, and ADMM-PnP

General information:

In this work we consider 8 test images: cameraman, house, peppers, Lena, Barbara, boat, hill, and couple. They can be downloaded from moodle.
Each ground truth image should be considered in its original size (i.e., do not use resize) and after loading, its intensity values should be divided by 255, so the range is within [0,1].

Methods will be quantitatively evaluated via the PSNR, given in this case by $PSNR = 10\,log_{10}\left(\frac{r^2}{MSE}\right) = 10\,log_{10}\left(\frac{1}{MSE}\right)$, where $MSE = \frac{1}{n}\sum_{i,j}\left(\hat{x}(i,j) - x_{gt}(i,j)\right)^2$.
Image processing methods typically require hyper-parameters tuning, e.g., aiming for maximizing PSNR. Note that it is not valid to tune hyper-parameters differently per image, because in "real life" the ground truth image is unknown. On the other hand, it is considered valid to optimize via the PSNR a **single** tuning configuration that is shared by a (sufficiently large) test set. Typically, good tuning for 5-10 diverse images will perform well also on larger test sets.

- The project report must be printed in Word or LaTeX, and submitted electronically as a PDF file.
- Include all your code within the document as an appendix, with syntax coloring and indents, as it appears in the Python or Matlab editor.
- Wrap up the report and the code files inside a single ZIP file with name that includes your ID.

Q1. Denoising problem

Here, the goal is to output $\hat{x}$, an estimate of x_{gt}, given $y = x_{gt} + e$ where x_{gt} is the original image (after division by 255) and $e \sim \mathcal{N}(0, \sigma_e^2 I)$.

Implement the bilateral filter (BF):
`bilateral_filter(img, sigma_s, sigma_r)`

Write a script `runme_denoising` where you apply the BF denoiser on the test set for Gaussian noise of level $\sigma_e = 0.1$. Fix the random seed to 0 before generating the noise (this will make the results reproducible).
Report in a table the PSNR of y (input PSNR) as well as the PSNR of $\hat{x}$ per image and the average PSNR across images. What values of σ_s and σ_r gave the best results?

Compare the results with those of a BM3D denoiser, which incorporates non-local repetition of patches with their sparsity under certain transformations.

April 9, 2025

You can install/download the BM3D denoiser from
https://pypi.org/project/bm3d/
or
https://webpages.tuni.fi/foi/GCF-BM3D/index.html

Provide several visual examples $(x_{gt}, y, \hat{x})$.

Q2. Deblurring problem

Here, the goal is to output $\hat{x}$, an estimate of x_{gt}, given $y = x_{gt} * k + e$ where x_{gt} is the original image, $e \sim \mathcal{N}(0, \sigma_e^2 I)$, and k is a blur kernel. We assume that the 2D convolution is **cyclic** (in practice, we can assume it and ignore the boundary pixels of the reconstruction). Specifically, we consider a setting where $\sigma_e = 0.01$ (use random seed of 0) and $k(i,j) = \frac{1}{1+i^2+j^2}$, $i,j = -7, \ldots, 7$, followed by scaling such that $\sum_{i,j} k(i,j) = 1$.

Q2a.

Write in your document the scheme of ADMM-PnP for deblurring. Provide a closed-form expression for the data-fidelity update that utilizes DFT ($\mathcal{F}$ and $\mathcal{F}^{-1}$) for efficient implementation.

Q2b.

Write a script `runme_deblurring` where you apply the ADMM-PnP on the test set with the BM3D denoiser and with the BF denoiser (introduce a flag to select between them).
The following code can faciliate your FFT2 based implementation:
https://github.com/tirer-lab/DDPG/blob/fcd17382d6b2d084b4bd2686c531b61e392cc1a9/functions/fft_operators.py#L222

Tune the number of iterations, the ADMM penalty parameter ρ and the denoiser's noise level σ (associated with $\sqrt{\beta/\rho}$ in the slides). For the BF tuning σ translates to tuning σ_s and σ_r. Better denoiser is expected to lead to better deblurring results. So with BM3D you will get better results, and you are not required to exhaustively optimize the tuning for BF.

Report in a table the PSNR of y (input PSNR) as well as the PSNR of $\hat{x}$ per image and the average PSNR across images. What hyper-parameters values gave the best results?

Provide several visual examples $(x_{gt}, y, \hat{x})$.

April 9, 2025

Q2c.

Can you improve the results by modifying the baseline algorithm (e.g., allow more flexibility in the hyper-parameters)?
Explain any modification and present (in table, as before) improved results with the BM3D denoiser.

Q3. Inpainting problem

Here, the goal is to output $\hat{x}$, an estimate of x_{gt}, given $y = x_{gt}(M)$ where x_{gt} is the original image, and M is a random set containing indices of 20% of the pixels of the image (use random seed of 0 before uniformly drawing the indices). That is, 80% of the pixels are not observed.

Q3a.

Write in your document the scheme of ADMM-PnP for inpainting. Provide a closed-form expression for the data-fidelity update that allows efficient implementation.

Q3b.

Write a script `runme_inpainting` where you apply the ADMM-PnP on the test set with BM3D denoiser. You may use any of the advancements found in Q2c.
As an initialization, you can use median inpainting:
https://github.com/tirer-lab/CM4IR/blob/main/functions/median_inpainting.py
https://github.com/tomtirer/IDBP/blob/master/utilities/median_inpainting.m
or any other prior-free initialization and include it in your code.

Report in a table the PSNR of y (input PSNR) as well as the PSNR of $\hat{x}$ per image and the average PSNR across images. What hyper-parameters configuration gave the best results?

Provide several visual examples ($x_{gt}, y, \hat{x}$).