

CS 4391: Introduction to Computer Vision Homework 2

Yapeng Tian
The University of Texas at Dallas

Download the homework2_programming.zip file from eLearning, Course Homepage, Assignments, Homework 2. Finish the following programming problems and submit your scripts to eLearning. You can zip all the data and files for submission. TA will run your scripts to verify them.

Install the Python packages needed by

- `pip install -r requirement.txt`

Here are some useful resources:

- Python basics <https://pythonbasics.org/>
- Numpy <https://numpy.org/doc/stable/user/basics.html>
- OpenCV https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html

Problem 1

(1 points) Linear filtering with a mean/box filter.

Implement the `linear_local_filtering()` function in `p1_linear_filtering.py`. The function takes an image and a 7×7 box filter as input and outputs a filtered image.

After your implementation, run the `p1_linear_filtering.py` in Python to verify it. The filtered image will be saved in “results/im_box.png”. Figure 1c shows a generated image using the box filter.

Problem 2

(1 points) Linear filtering with a Gaussian filter.

Implement the `gauss_kernel_generator()` function in `p1_linear_filtering.py`. The function takes a kernel size: $2n + 1$ and a variance σ_s^2 as input and outputs a $(2n + 1) \times (2n + 1)$ Gaussian kernel, where $n = 3$ and $\sigma_s^2 = 15$ in the script.

After your implementation, run the `p1_linear_filtering.py` in Python to verify it. The filtered image will be saved in “results/im_gaussian.png”. Figure 1d shows a generated image using the filter.

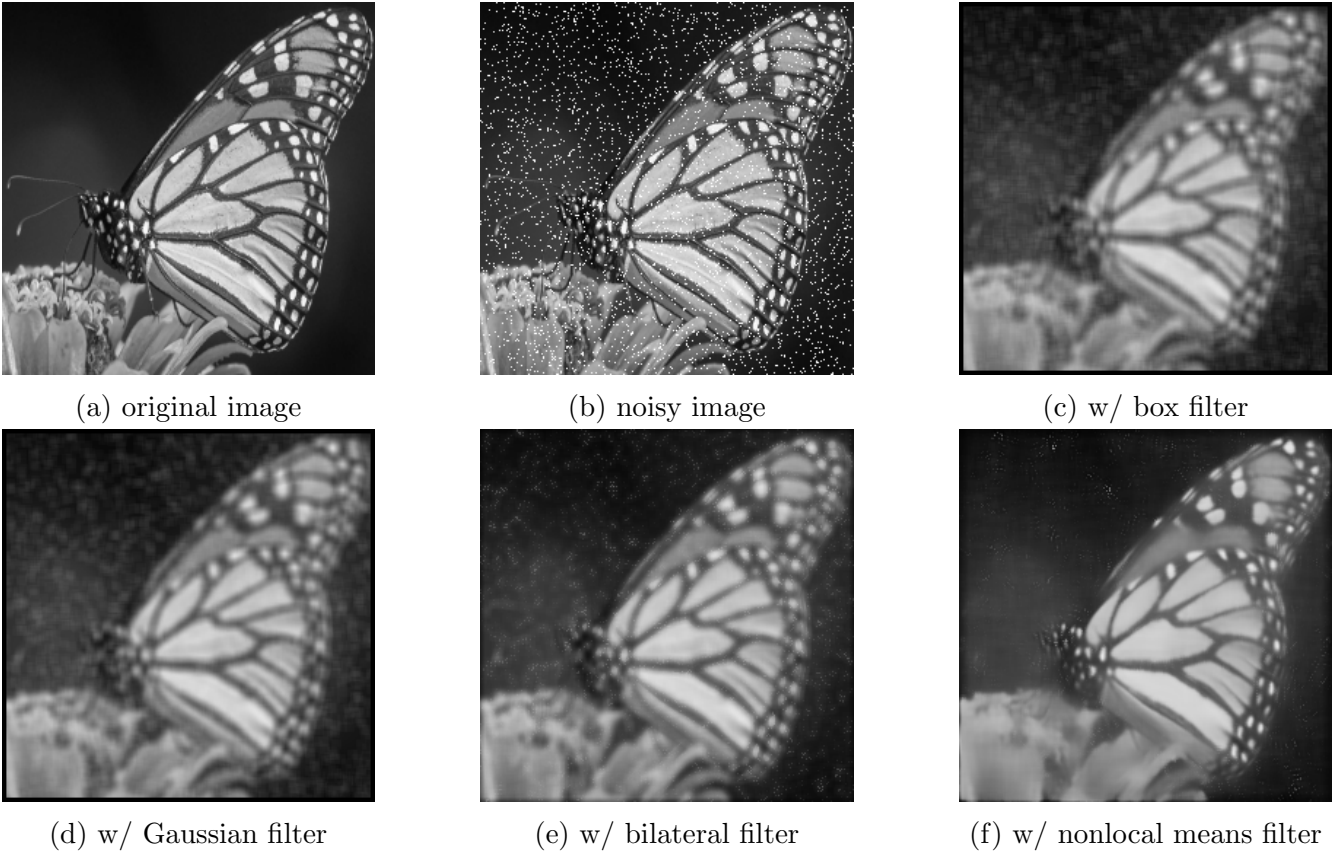


Figure 1: Clean, Noisy, and filtered images using different filters.

Problem 3

(2 points) Nonlinear filtering with a bilateral filter.

Implement the `bilateral_filtering()` function in `p2_bilateral_filtering.py`. The function takes an image, a spatial variance σ_s^2 , an intensity range variance σ_r^2 , and a kernel size $(2n + 1) \times (2n + 1)$ as input, and outputs a filtered image, where $n = 3$, $\sigma_s^2 = 30$, and $\sigma_r^2 = 0.5$ in the script.

After your implementation, run the `p2_bilateral_filtering.py` in Python to verify it. The filtered image will be saved in “results/im_bilateral.png”. Note that you need to use zero-padding to handle the black border issue in your code. Figure 1e shows a generated image using the filter.

Problem 4

(2 points) Nonlinear filtering with a non-local means filter.

Implement the `nlm_filtering()` function in `p3_nlm_filtering.py`. The function takes an image, an intensity range variance σ_r^2 , a patch size $(2n + 1) \times (2n + 1)$, and a search window size $(2N + 1) \times (2N + 1)$ as input, and outputs a filtered image, where $n = 2$, $\sigma_r^2 = 1$, and $N = 7$ in the script.

After your implementation, run the `p3_nlm_filtering.py` in Python to verify it. The filtered image will be saved in “results/im_nlm.png”. Note that you need to use zero-padding to handle the black border issue in your code. Figure 1f shows a generated image using the filter.