BİL 467/561 – Image Processing

Homework #3

Assigned on 03.10.2022 – Due on 10.10.2022

- Submit one Python script file. Do not submit Jupyter Notebooks.
- Make sure your submission file name is formatted as:
  FirstName_LastName_StudentID_HW#.py
  For example: Toygar_Akgun_123456789_HW3.py

1. [40 points] For this question, you will implement the median filter for 8-bit grayscale images as detailed in our lecture slides. **You are not allowed to use OpenCV's or any other image/signal processing API's built-in median filtering function for this part.**

   Once you implement your own median filtering function, you will apply it to the provided test image. **The provided test image (noisyImage.jpg) will be corrupted with salt-and-pepper noise and your median filter (with 5x5 window size) will be used to denoise it. Do NOT add additional noise to the provided test image. Just use it as it is.**

   You will then apply OpenCV's built-in median filter to the same test image. Your median filter output must match OpenCV's median filter output perfectly. If the difference between these two output images has any non-zero pixels, this is an instant fail, meaning you will get zero points for this question. **Hint: OpenCV's median filter uses BORDER_REPLICATE padding.**

2. [10 points] In academic and industrial image restoration/reconstruction algorithm development works, there is always a need to measure how well an algorithm performs. For this purpose, it is quite common to gather a set of test images (called originals, goldens or ground truth images) and corrupt these images (for example, by adding noise) with the degradation for which the restoration algorithm is being developed. These corrupted images then become the test set on which the algorithm is tested. **The key observation is this: Since these test images are obtained by corrupting the originals, the algorithm's outputs can be compared against the original images to see how well the algorithm works (how well it reconstructs).** This is how we measure the performance of restoration/reconstruction algorithms.

   Peak Signal to Noise Ratio (PSNR) is a metric (in decibels) that can be used to measure how good a reconstructed (for example, denoised – as you did in Question 1) image matches the original (uncorrupted, golden) image.

   The PSNR is defined as:

$$PSNR = 10\log_{10}\left(\frac{R^2}{MSE}\right) \quad \text{where } MSE = \frac{\sum_{M,N}[f(x,y) - g(x,y)\,]^2}{M * N}$$

   Here, Mean Squared Error (MSE) is the average (over all pixels) of the squared differences between two images. For 8-bit grayscale images, R is the maximum intensity value, that is 255.

In case of denoising, one image is the original (clean, ground truth) and the other image is the output of the denoising algorithm. **The higher the PSNR metric, the better.** That is, if the PSNR value of the output of a denoising algorithm, say algorithm_1, is higher than the PSNR value of the output of another denoising algorithm, say algorithm_2, then algorithm_1 is likely to be a better denoising algorithm. **Since there are multiple different definitions of PSNR with very minor differences, we will use OpenCV's PSNR calculation function (cv.PSNR).**

For this question, you will use OpenCV's built-in

1. 5x5 box filter,
2. zero mean, 7x7 Gaussian function,
3. 5x5 median

filters to denoise the noisy test image.

You will be provided with the clean version (lena_grayscale_hq.jpg) of the noisy test image to use as the ground truth. **Using OpenCV's PSNR calculation function,** compute PSNR values for these three denoised output images.

3. [30 points] For this question, you will modify your median filter implementation to make it center weighted as we discussed in the class. That is, instead of using each pixel value only once to compute the median, this modified version will count the center pixel 3 times (by replicating it 3 times in the sorted array), making your median filter "center weighted" (putting more weight on the center pixel).

   Once you are done, you should have five outputs:

   1. Your own median filter output
   2. OpenCV's box filter output
   3. OpenCV's Gaussian filter output
   4. OpenCV's median filter output
   5. Your own center weighted median filter output.

   You will be provided with the clean version (lena_grayscale_hq.jpg) of the noisy test image to use as the ground truth. **Using OpenCV's PSNR calculation function,** you will compute five PSNR values for these five denoised output images. Note that the PSNR values for (1) and (4) must be the same. You will display these outputs and print the corresponding PSNR value in the figure title in a clearly visible way.

   [<u>Observation</u>: What are the PSNR values? According to PSNR, which filter performs the best? Can you tell why? – You do not need to return any result/code for this observation]

4. [20 points] By looking at the definition of the PSNR metric, is PSNR a good (dependable) metric? Can we say that a lower PSNR value **always** means a **worse looking** (visually lower quality) picture? Find and code a very simple image transformation that breaks the PSNR metric. That is, when this transformation is applied to a well-reconstructed image (for example, the best denoising filter's output from the last question) its output should be visually very similar to the corresponding original (clean) image, but the corresponding PSNR value should be much lower.