OPTI 536 – Introduction to Image Science
Spring, 2025

Homework 1

Assigned: Feb. 5, 2025
Due: Feb. 19, 2025 at 11:59 pm

This is the first homework that focuses solely on programming. It should be still possible to solve the problems with any programming language of your choice. However, **we strongly suggest to use MATLAB or Python for your solutions**. If you use Python, you will need the OpenCV package.

Feel free to discuss the homework with others in the class or the instructors. You may use any books or websites you find, but please give a brief citation for any references you use. You must write up your final solutions independently without direct copying from these sources or from solutions by other students. **Please be aware that we implemented advanced measures to check for duplicated code (also for code downloaded from the internet)!**

Please submit your solutions to the D2L website by 11:59pm on the due date. This time, your submission should consist of a **pdf that includes all requested plots and the answers to the questions** PLUS **one code file for each problem** (i.e., 1 pdf and 3 code files in total). Each code file should be a single file (e.g., **`*.m-file`**, **`*.py`** or **`*.ipynb`**-file) that automatically generates all requested plots/figures when executed. Again, make sure to include the requested plots and figures also in your "answer-pdf".

Please do not convert your pdf to grayscale this time (color is important) and do not lock/encrypt any of your submitted files.

Start with this homework EARLY, so that we have enough time to help you with any problems.

Problem 1. (30 points)

In this problem, we will play around with some simple image processing algorithms. We will first load an image of a dog and eventually overlay this image on top of a picture of a natural landscape.

Note: For this problem, you may need to use the MATLAB functions **`imread()`**, **`imresize()`**, **`imagesc()`**, **`conv2()`**, or their Python equivalents.

a. Load one of the two provided dog images. Both of these images have 4 channels (not 3). The 4th channel is the so-called alpha channel that contains the opacity of the image. In MATLAB, you must use **`[img, ~, alpha] = imread('YOURIMAGE.png')`**, otherwise the alpha channel is not loaded. Plot the RBG image of your dog and the respective alpha channel in two separate figures.

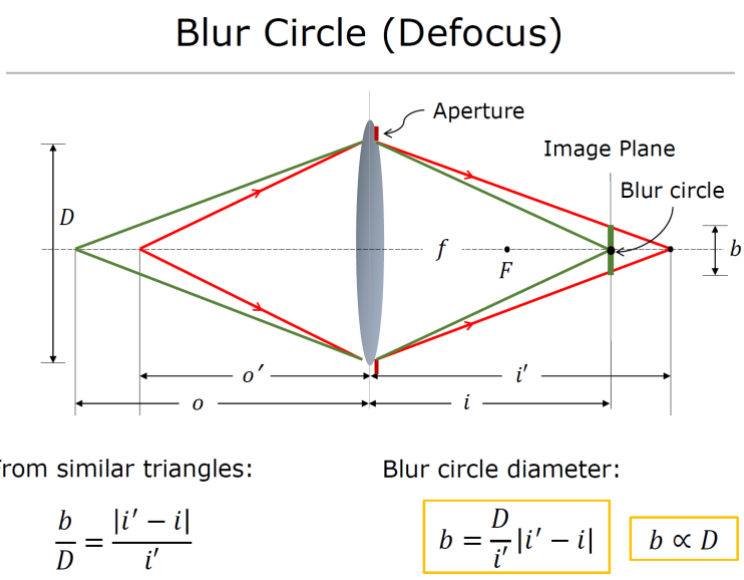
b. Load the background image '**yosemite.jpg**' as the background and plot the image.

c. Now we want to overlay images of the same size. For this, crop out a part of the background image that has the same size as your dog image (e.g., the upper left corner). Plot your choice of the cropped background image. Overlay your dog image on top of your cropped background image. Use the alpha channel to decide if a pixel in the overlayed image should be from the dog or the background. Plot your result.

d. Now we want to overlay images of arbitrary sizes. This time we will use the full background image. Use the **`imresize()`** function to scale the dog to an appropriate size, and then overlay the dog on the center of the background, similar to the image shown above.

e. We want to add some blur to the background to simulate the effect of portrait photos that appear naturally in DLSR cameras, or are simulated for similar effects in modern smartphones ("Bokeh-effect"). For this we need to convolve the background image with a gaussian filter kernel. You may use the MATLAB **`conv2()`** function or its equivalent in Python. Use a gaussian kernel with a standard deviation of your choice, but make sure that the blurring effect is visible. Overlay your dog on the filtered image and plot your result.

f. Finally, we will calculate a realistic value for the standard deviation of your gaussian kernel. The formulas to calculate a geometric blur circle b from geometrical optics are given below. D is the diameter of the aperture, i is the image distance of the object in focus, i' is the image distance of the blurred object. Assume the focal length of your camera is 50 mm, the aperture diameter is 10mm, the dog is located at 1.5m standoff and the background is at 1000m standoff from the camera. The sensor size of your camera is 35mm in the horizontal dimension.
Write down your calculation for b. Set $b = 2\sigma$ for your new gaussian blur kernel. Blur the background with the new kernel, overlay it with your dog, and plot the result.



g. **Fun and extra points (does not need to be answered to get full points for this HW):** Find another image with an alpha channel on the internet (or create one) and overlay two images of your choice. Be creative! Feel free to use gaussian blur or any other cool filter kernel (e.g., motion blur). Plot your result and tell us quickly what you did so that we can appreciate your creativity!

Problem 2. Grayscale image denoising using Bilateral filter (30 points)

In this problem, we want to denoise a grayscale image with a bilateral filter. You will implement your own bilateral filter using the formulas discussed in class. Eventually, you will compare your result with the built-in MATLAB / Python filters.

$$g[i,j] = \frac{1}{W_{sb}} \sum_m \sum_n f[m,n]\, n_{\sigma_s}[i-m, j-n]\, n_{\sigma_b}(|f[m,n] - f[i,j]|$$

where

$$n_{\sigma_s}[m,n] = \frac{1}{2\pi {\sigma_s}^2} e^{-\frac{1}{2}\left(\frac{m^2+n^2}{\sigma^2}\right)} \qquad n_{\sigma_b}[k] = \frac{1}{2\pi {\sigma_b}^2} e^{-\frac{1}{2}\left(\frac{k^2}{\sigma^2}\right)}$$

$$W_{sb} = \sum_m \sum_n n_{\sigma_s}[i-m, j-n] n_{\sigma_b}(|f[m,n] - f[i,j]|)$$

a) Load **NoisyGrayImage.png** image and plot it.

b) Explain the function of the two parameters of σ_b, σ_s in the bilateral filter. What are typical value ranges for these parameters, assuming that you have an 8-bit image? Explain your answer.

c) Implement your own bilateral filter function using the formula above (do not use any built in MATLAB / Python function for this). Go back to the lecture recording if you need to understand

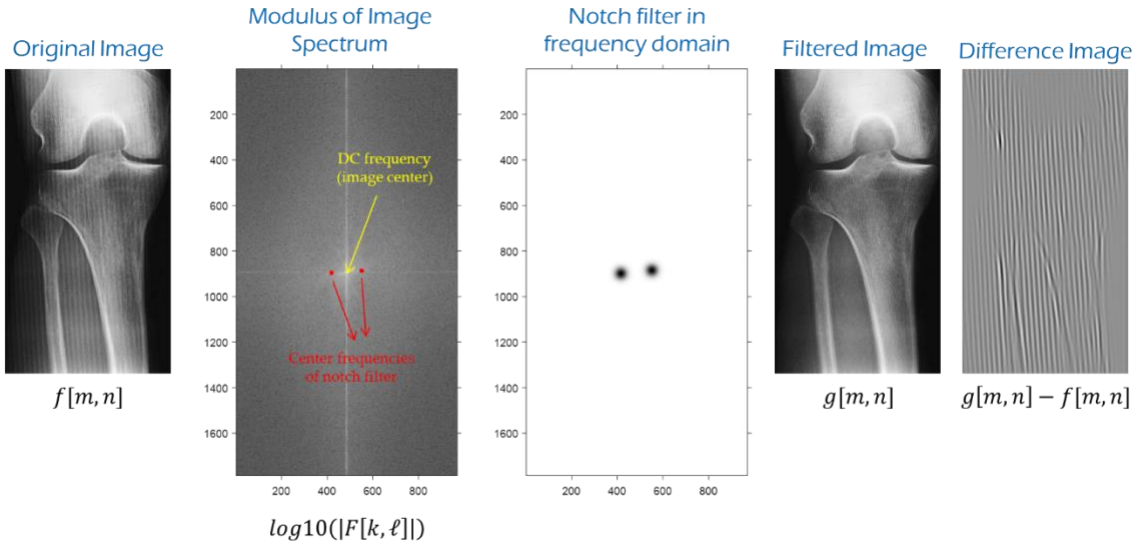
the individual parameters in the formula. Your filter function must have the following inputs: *input image, values for* σ_b *and* σ_s.
Apply your bilateral filter to **NoisyGrayImage.png** by using all four possible combinations of $\sigma_s = 3, 10$ and $\sigma_b = 0.2, 0.4$. Plot all four results and explain the differences you see and WHY you see these differences.

d) Validate your results by comparing them with the MATLAB `imbilatfilt(I,degreeOfSmoothing,spatialSigma)` function using the same choice of parameters you picked above. Plot a 2x5 figure (row x column, columns: original image plus 4 possible combinations of given σ_s and σ_b) for a nice comparison and make sure to label your figures accordingly.
Important note: Please be aware that the MATLAB function input `'DegreeOfSmoothing'` corresponds to the variance of the intensity kernel, meaning that you need to square your respective standard deviation σ_b for a correct input.

e) **Fun and extra points (does not need to be answered to get full points for this HW):** Take one of your pairs for σ_b and σ_s. Leave σ_s as is and set σ_b to a very large value ($\sigma_b > 255$). Apply the resulting bilateral filter again to **NoisyGrayImage.png** and plot your result. Now apply a standard MATLAB / Python gaussian filter to **NoisyGrayImage.png**, using your above choice of σ_s as standard deviation. Plot the result and compare it with the image from above. What do you see? Why?

Problem 3. Suppressing Moire by notch filtering in frequency domain (40 points)

The included image **Moire_Example2.png** shows an image of a robotic arm with periodic noise, called Moire. The noise resembles a pattern of closely spaced diagonal lines. The image inset right clearly illustrates the pattern. Your task is to implement a notch filter in the Fourier domain that removes this noise, i.e., reproduce the related slide that has been shown in the lecture (see below) with the provided image.



a) Load the **Moire_Example.png** image and plot it. Take a 2D Fourier transform and plot your result (use logarithmic plot to see something). Explain what you see, in particular the origin of the distinct peaks in the Fourier spectrum.

b) Use the MATLAB **`findpeaks()`** function (or the equivalent in Python) to find the exact position of the peaks which are not the DC peak. Plot the found position of the peaks in your Fourier Transform image.

c) The frequency response of each of the two required notch filters is derived from a Gaussian as follows: $H[k,\ell] = 1 - \exp\left(\frac{-1}{2D_0^2}[(k-k_0)^2 + (\ell-\ell_0)^2]\right)$ where k_0, ℓ_0 are the center frequencies of the notch filter corresponding to the row, column coordinates of the evaluated peaks in the Fourier plane of your image. The final notch filter is obtained as the product of the individual notch filters (you should use element to element multiplication). Now, set D_0 to 25pix and plot your final notch filter on a grid that equals the size of your Fourier transformed image (similar to the example picture in middle).

d) Apply the notch filter from above on the Fourier spectrum and plot the result. Take the inverse Fourier transform of your filtered Fourier Spectrum and plot the result. What do you see? Subtract your result from the original image and plot the difference image.

e) **Fun and extra points (does not need to be answered to get full points for this HW):** Is D_0 = 25 pixel a good choice for this image? Play around with different values of D_0, and plot and explain your results. Examples to explore: What happens if you pick a very large D_0, much larger than k_0 or ℓ_0? What happens if you shift around your notch filter in the Fourier spectrum?