SCC0251 + MAI5020 — Prof. Moacir Ponti

PAEs: Gustavo Evangelista Araújo and Leo Sampaio Ferraz Ribeiro

Assignment 2: Fourier Transform

Code the assignment by yourself. Ask if you need help. Plagiarism is not tolerated.

1 Introduction

1.1 Goal

The objective of this exercise is to deepen the students understanding of the frequency domain of images and how to use Fourier Transform and filtering techniques to match the expected results.

1.2 Task

In this assignment, your task is to match a set of given images using filters in the frequency domain. Read the instructions for each step. Students are required to use `python 3` and the libraries `numpy`, `imageio` and `scipy` to complete the task.

Follow the instructions carefully:

1. **Read** the parameters:
   a) **Input image** I (available in `./Dataset/in`);
   b) **Expected image** H (available in `./Dataset/gt`);
   c) **Filter index** $i \in$ [0,1,2,3,4,5,6,7,8];
   d) **Filter Parameters** respective to each index.
2. **Implement** the filters below:
   a) Ideal Low-pass - with radius r;
   b) Ideal High-pass - with radius r;
   c) Ideal Band-pass - with radius r_1 and r_2;
   d) Laplacian high-pass (edit: in a previous version we have wrongly asked for low-pass);
   e) Gaussian Low-pass - with σ_1 and σ_2;
   f) Butterworth low-pass - with D_0 and n;
   g) Butterworth high-pass - with D_0 and n;
   h) ~~Butterworth band-reject - with D_0, n_0, D_1 and n_1~~ Because of a confusion with the order of parameters, the test cases with this filter will not count towards your grades

    i) ~~Butterworth band-pass - with D_0, n_0, D_1 and n_1;~~ Because of a confusion with the order of parameters, the test cases with this filter will not count towards your grades

3. **Compare** the restored images $\hat{H}$ against expected images H using Root Mean Squared Error (RMSE);
4. **Output** the RMSE;
5. (Recommended) **Save** each restored image for further observations;

2 Dataset

Set12 is a collection of 12 grayscale images of different scenes that are widely used for evaluation of image denoising methods [2]. The size of each image is 256×256. However, the images will have varying levels of blur and periodic noises added to them or will extract edges, as seen in 2. Your algorithms should aim to produce an RMSE that closely matches the expected result, given the provided parameters.

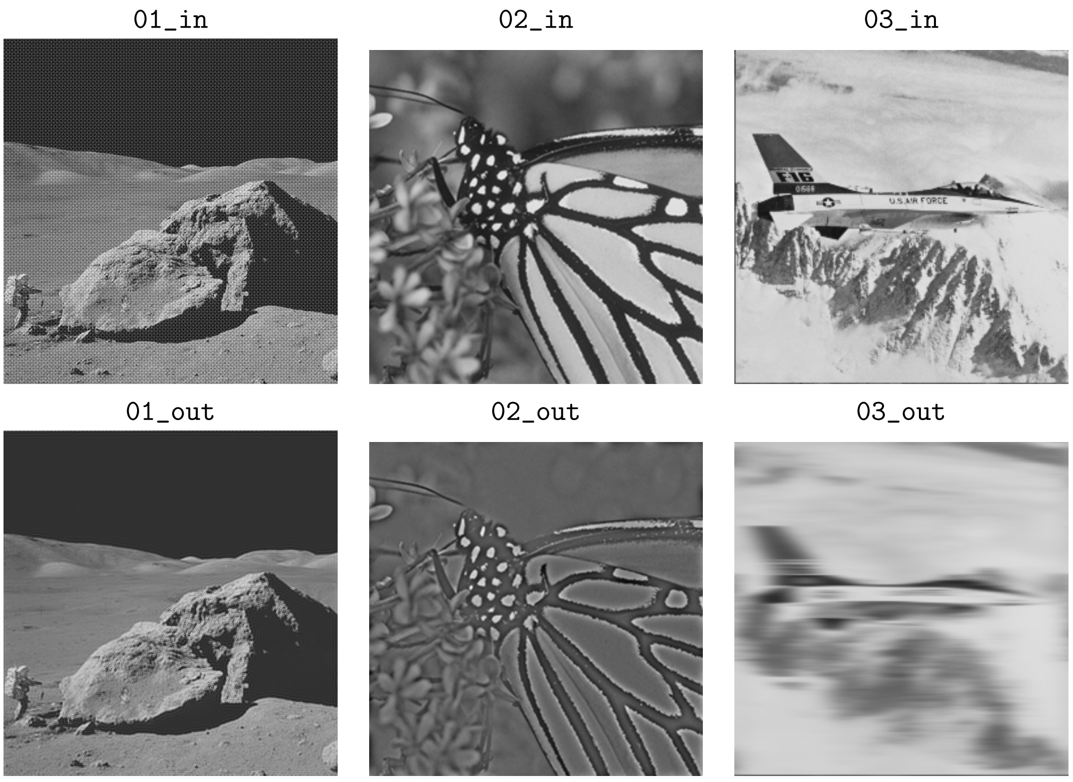


Figure 1: Examples of inputs and ideal outputs images from Set12 to compare

3 Filters

As previously described, the students are required to test various filter types on real grayscale images. This includes the ideal low-pass, ideal high-pass, and band filters that were covered in class. In addition, they need to learn about three other filter types: Laplacian, Gaussian, and Butterworth filters, which will be introduced.All filters in use follow the theory presented in the book "*Digital Image Processing - 3º edition*" [1].

Your algorithm will need to compute the 2D FFT of the input image `img` using `np.fft.fft2()`, a function takes an input array as its argument and returns the complex-valued 2D FFT of the input. The resulting complex-valued array is then shifted so that the zero-frequency component is at the center using `np.fft.fftshift()`. This is necessary because the FFT outputs the frequency components in a format where the zero frequency is at the origin. In summary, the image in the frequency domain F, will be expressed as:

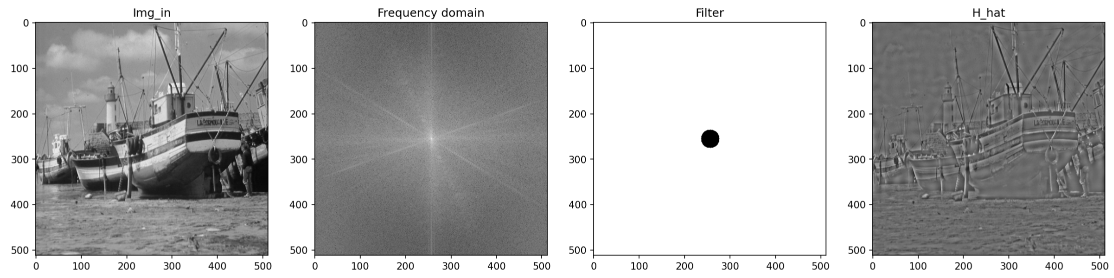
```
F = np.fft.fftshift(np.fft.fft2(img))
```

This operation will be needed in every filter function, **in order to initiate the filter shape** (P, Q) **and values**. It's important to define `dtype` as a float32, as shown below:

```
filter = np.zeros((P,Q), dtype=np.float32)
or
filter = np.ones((P,Q), dtype=np.float32)
```

It's important to note that each filter will compute the $H[u, v]$ values in the frequency domain. It's worth mentioning that not all filters presented below will be evaluated for the students' grades, only those specified in section 1.2. However, all variations of the filters are explained here to provide a complete understanding of the topic.

In summary, remember to do computations in the correct domain: **(1)** Get the FFT of the image F; **(2)** Do the filtering process in the frequency domain; **(3)** Apply the filter in F; **(4)** Get the inverse FFT of the image to return it back to the spatial domain $\hat{h}$.



3.1 Ideal Filters

In contrast to what was covered in class, the ideal filters used here are circular rather than square in shape. An ideal lowpass filter (ILPF) is a 2D filter that allows all frequencies within a circular region with a given radius from the center of the image to pass without attenuation, and blocks all frequencies outside this region. It is called "ideal" because it provides a perfect cutoff with no transition band or ripple in the passband.

The filter can be mathematically described as a binary function that is equal to 1 within the circular region and 0 outside. This filter can be used to remove high-frequency noise from an image while preserving its low-frequency content. It is specified by the functions:

$$D(u,v) = \sqrt{(u-\frac{P}{2})^2 + (v-\frac{Q}{2})^2} \tag{1}$$

Lowpass

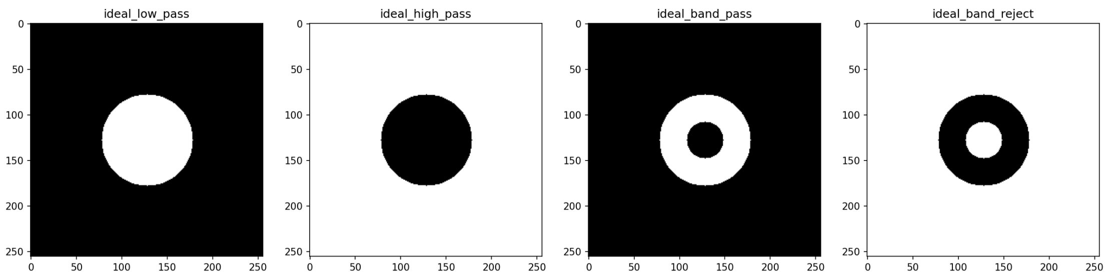
$$H(u,v) = \begin{cases} 1, & \text{if } D(u,v) <= D_0 \\ 0, & \text{if } D(u,v) > D_0 \end{cases} \tag{2}$$

Highpass

$$H(u,v) = \begin{cases} 0, & \text{if } D(u,v) <= D_0 \\ 1, & \text{if } D(u,v) > D_0 \end{cases} \tag{3}$$

Bandpass and Bandreject

Both Bandpass and Bandreject algorithms can be created from operations with lowpass and highpass filters. For example, assuming the radius $r_0 > r_1$, you could subtract a ILPF with radius r_0 from another ILPF with radius r_1 to get a bandreject filter.



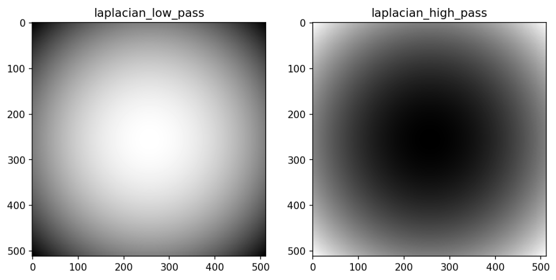
3.2 Laplacian

The Laplacian operator is a second-order derivative that is used in image processing to enhance edges and details. The Laplacian filter can be implemented as a high-pass filter to enhance edges or as a low-pass filter to smooth an image. The equation 4 is the frequency response of a Laplacian low-pass filter in the frequency domain.

$$H[u,v] = -4\pi^2((u-\frac{P}{2})^2 + (v-\frac{Q}{2})^2) \tag{4}$$

The filter is defined by the function H(u,v), where u and v are the spatial frequency variables, and P and Q are the dimensions of the image in the x and y directions, respectively.

The Laplacian filter is a high-frequency filter that attenuates high-frequency components in the image. The equation achieves this by applying a negative value to H(u,v) for higher frequency components. The squared distance between the frequency components (u,v) and the center of the frequency domain is calculated and multiplied by a factor of $-4\pi^2$. This factor determines the rate of attenuation for higher frequency components. The farther away the frequency component is from the center, the greater the attenuation.



3.3 Gaussian

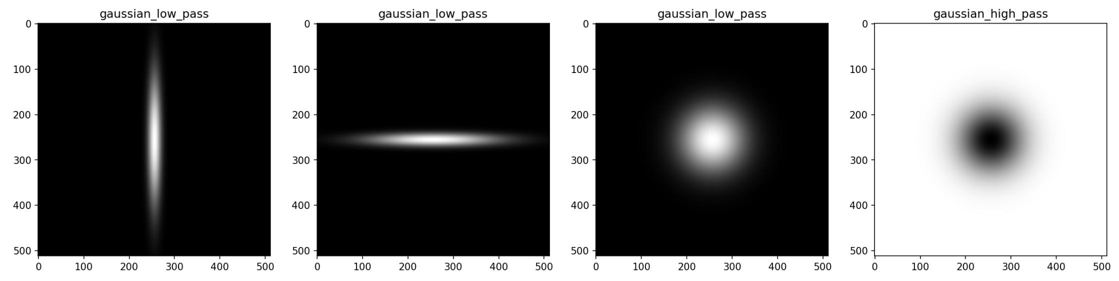
A Gaussian low-pass filter is a type of filter used in image processing to suppress high-frequency information in an image, providing a smooth alternative to the ideal low-pass filters previously presented; the effect is effectively a "blur" and is the filter most commonly used in user interface elements.

The equations 5 and 6 represents this filter. The standard deviation, σ, determines the width of the Gaussian curve, with a smaller value of σ resulting in a sharper filter.

$$x = (\frac{(u - \frac{P}{2})^2}{2\sigma_r^2} + \frac{(v - \frac{Q}{2})^2}{2\sigma_c^2}) \tag{5}$$

$$H(u, v) = e^{-x} \tag{6}$$

The frequency domain representation of the Gaussian filter is expressed in terms of the frequency coordinates in the row and column directions, denoted by $u - P/2$ and $v - Q/2$, respectively. The filter response at each frequency coordinate is determined by evaluating the Gaussian function at that coordinate. The exponent in the equation is the squared distance of the frequency coordinate from the center of the filter, divided by twice the squared standard deviation in the corresponding direction. Note that you can subtract the filter from a fully blank filter to get a high-pass version.



3.4 Butterworth

The transfer function of a Butterworth lowpass filter (BLPF) of order n, and with cutoff frequency at a distance D_0 from the origin, is defined as:

$$H[u,v] = \frac{1}{1 + [D(u,v)/D_0]^{2n}} \tag{7}$$

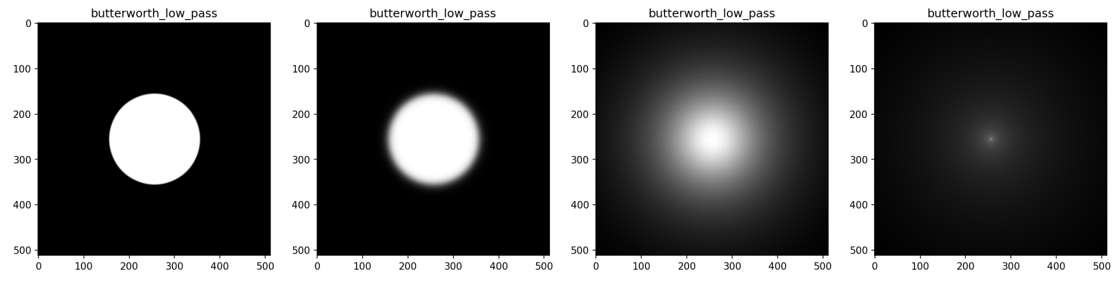
and for $D(u,v)$, we have:

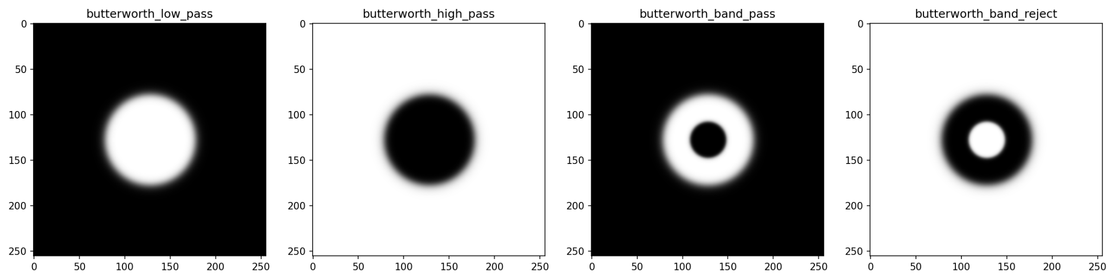
$$D(u,v) = [(u - \frac{P}{2})^2 + (v - \frac{Q}{2})^2]^{1/2} \tag{8}$$

where u and v are the spatial frequency variables, and P and Q are the dimensions of the image in the x and y directions, respectively. The Butterworth filter approaches the ideal filter in higher order values. For lower order values, the Butterworth filter is more like a Gaussian filter. Thus, the Butterworth filter may be viewed as providing a transition between two extremes.

One of the principal applications of bandreject filtering is for noise removal in applications where the general location of the noise component(s) in the frequency domain is approximately known. A good example is an image corrupted by additive periodic noise that can be approximated as two-dimensional sinusoidal functions.

Note that the Butterworth highpass filter (BHPF) can be obtained from a simple operation with the BLPF, as were possible with the previous filters. The same goes for the bandpass and bandreject algorithms.





4 Input and Output

The following parameters will be input to your program in the following order through `stdin`, as usual for run.codes:

input	**output**
Dataset/in/01.png Dataset/out/01.png 4 20 1	0.1037

5 Comparing against expected

Your program must compare the restored image against expected h. This comparison must use the root mean squared error (RMSE). Print this error in the screen, rounding to 4 decimal places. Because the RMSE values will be higher in this assignment, convert the matrices to `np.int32` before computing to avoid under and overflow.

$$RMSE = \sqrt{\frac{\sum_i \sum_j (g(i,j) - f(i,j))^2}{n \cdot m}}$$

6 Grading

Your work will be graded as:

$$\frac{R + F1 + F2 + F3 + F4}{5} - P$$

where each value ranges from $0 - 10$, R is the grade from run-codes-local, A is the grade for each new filter implemented (Ideal, Laplacian, Gaussian and Butterworth Filters, respectively). P goes up to 1.0 and is a possible penalty for failing to follow the rules from the previous section.

7 Submission

Submit your source code to e-disciplinas (only the `.py` file). You can check for correctness by downloading the test cases from e-disciplinas and testing with run-codes-local, which will be used by the PAEs to grade your work.

1. **Use your USP number as the filename for your code.**
2. **Include a header**. Use a header with name, USP number, course code, year/semester and the title of the assignment. A penalty on the evaluation will be applied if your code is missing the header.
3. **Comment your code**. For any computation that is not obvious from function names and variables, add a comment explaining.
4. **Organize your code in programming functions**. Use one function for each filter method.

References

[1] R. C. Gonzales and P. Wintz. Digital image processing. Addison-Wesley Longman Publishing Co., Inc., 1987.

[2] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. IEEE Transactions on Image Processing, 26(7):3142–3155, 2017.