Assignment 2

Digital Image Processing - M25

Deadline: 19th Sept, 2025 11:59PM

General Instructions

1. Follow the specified repository structure.
2. Make sure you run your Jupyter notebook before submitting to save all outputs. If you are submitting python scripts make sure to save all the outputs images.
3. Answer any questions asked in a markdown cell in your notebook.
4. Label your plots appropriately.
5. For each part of the assignment, you should implement both the filter and the convolution operation completely from scratch. Using OpenCV or any other built-in image filtering/convolution functions is not allowed.

---

1 Laplacian Sharpening [15 marks]

Unsharp masking and Laplacian sharpening are two common techniques for enhancing image sharpness. Some of you have noticed that they are similar. You're not wrong. Your job is to mathematically prove their relation

The general formulas are:

- **Unsharp Masking:** $g(x, y) = f(x, y) + k \cdot (f(x, y) - f_{\text{blurred}}(x, y))$
- **Laplacian Sharpening:** $g(x, y) = f(x, y) - c \cdot \nabla^2 f(x, y)$

where $f(x, y)$ is the original image, $g(x, y)$ is the sharpened image, k and c are constants, and $\nabla^2 f(x, y)$ is the Laplacian of the image.

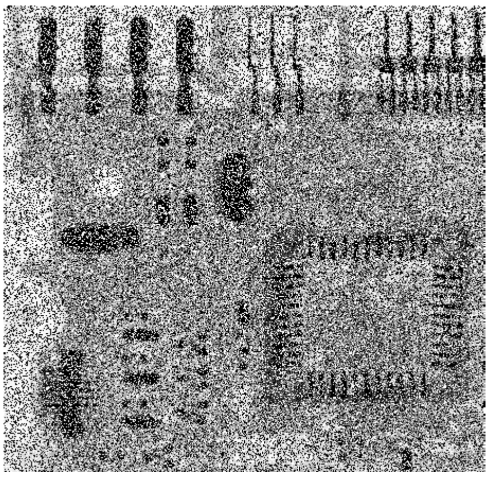
Task

Prove that **Laplacian sharpening is a specific instance of unsharp masking**.

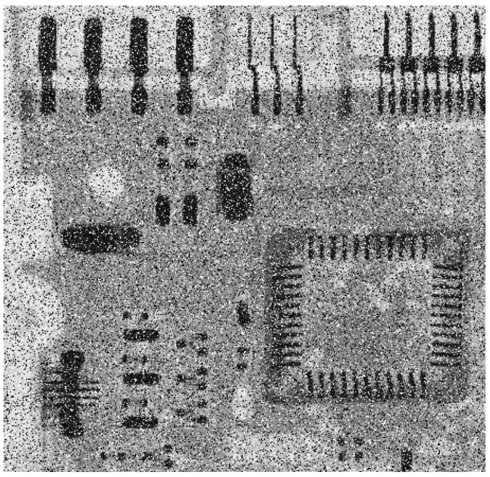
Hint: Start by considering how a blurred version of an image, $f_{blurred}(x, y)$, can be approximated using the Laplacian operator. The discrete Laplacian can be defined by a kernel that computes the difference between a pixel and the average of its neighbors.

2 Image Denoising [20 marks]

You are provided with two noisy images of a USAF 1951 resolution test chart that has been corrupted with noise. Your task is to implement and evaluate a suitable filter to remove this noise.



(a)



(b)

Figure 1: USAF 1951 resolution test chart images.

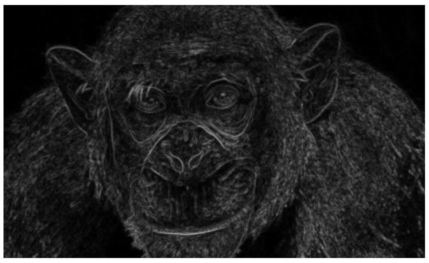
a) Implement a function of suitable filter for denoising.

b) Experiment with different kernel sizes(atleast 3) and explain how varying kernel sizes affect the final output.

c) Explain your choice of filtering and discuss any other alternatives

3 Bilateral Filtering for Edge-Preserving Smoothing [40 marks]

(a)



(b)

The Bilateral filter is a powerful edge-preserving smoothing filter whose behavior is controlled by two key parameters. In this section, you will first implement the filter, then analyze its parameters, and finally, compare its performance against the standard Gaussian filter.

a) **Implementation [15 marks]:** Write a function from scratch to implement a **Bilateral filter**. The function should take the input image, kernel size, σ_{space} (spatial standard deviation), and σ_{range} (intensity/range standard deviation) as parameters.

b) **Parameter Analysis [10 marks]:**

- Apply your bilateral filter to this image using varying parameters σ_{space} and σ_{range}
- Discuss the results. Explain the distinct roles of σ_{space} and σ_{range}. How does each parameter affect the blurring of textures versus the preservation of sharp edges?

c) **Comparison with Gaussian Filter [15 marks]:** Now, compare your tuned filter with a standard Gaussian filter.

- From your analysis in part (b), select one set of (σ_{space}, σ_{range}) that provides a good balance of smoothing and edge preservation.
- Apply your chosen **Bilateral filter** and a standard **Gaussian filter** to the original image.
- Apply a **Sobel edge detector** to the original image, the Gaussian-filtered image, and the Bilateral-filtered image.
- Compare the plots of both the filtered outputs and their outputs
- Briefly explain why the Bilateral filter is considered "edge-preserving" while the Gaussian filter is not.

4 DFT and FFT [25 marks]

a) Write a function `dft1` that calculates the DFT of a 1D array of arbitrary size. Use this function to decode the frequencies in the given audio file `1.wav` and plot the **frequency vs. magnitude** graph.

b) Write a function that calculates the 2D-DFT of a 2D array.

c) Write a function `fft1` that computes the DFT of a 1D array using the FFT algorithm. (You may pad the array with extra values to make its length a power of 2.)

d) Write a function `fft2` that computes the 2D-DFT of a 2D array using the FFT algorithm.

e) Compute the DFT of arrays of different lengths: $[128, 256, 512, 1024]$ using `fft1`. Plot the **time taken vs. array length** and comment on the observed differences.

f) Compute the 2D-DFT and 2D-FFT of the following image signals and display the outputs:

$$I_1 = 0.5(1 + \sin x), \quad I_2 = 0.5(1 + \sin x \sin y), \quad I_3 = 0.5(1 + \sin(x^2 + y^2))$$