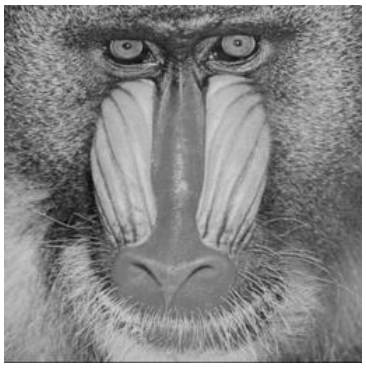
Bishop's University
Department of Computer Science
CS462-CS562 – Image Processing & Mathematical Modeling of Image
Winter 2024 – HW2

The following images can be useful for testing purposes: Lena, Mandrill, and Boat. They are provided to you.

Lina



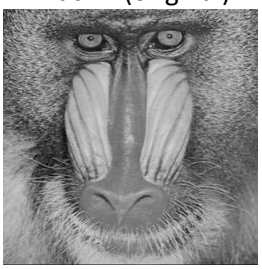
Mandrill

Boat

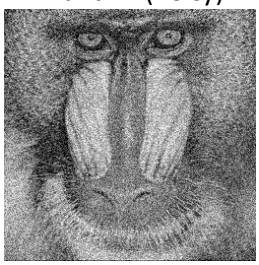
1. Develop a Python function, **load_image()**, to load an image. This function should accept an input parameter that specifies the image filename and returns a 2D array representing the image. If the input image is in color, the function should convert it to grayscale. When loading an image, convert it to float format and scale it between 0.0 and 1.0 by dividing by 255 if it was in the range [0, 255].
2. Construct a Python function, **add_Gnoise()**, to add Gaussian noise to an input grayscale image. The function should have three input parameters:
    a. The grayscale input image.
    b. The **mean** of the Gaussian noise, which is always 0.
    c. The standard deviation (**std**) of the noise, allowing adjustment of the noise level.
3. Develop a Python function, **add_SPnoise()**, to add Salt and Pepper noise to an image.
4. Construct a Python function, **filter_image()**, capable of filtering a noisy image using a selection of filters such as Gaussian, Median, and Bilateral filters. This function should accept the name of the desired filter as an input parameter, enabling flexible filter selection based on user preference.
5. Create a Python function, **psnr()**, to compute the Peak Signal-to-Noise Ratio (PSNR) of the filtered image.
6. Call **load_image()** with different image names such as Lena, Mandrill, and Boat.

7. Call **add_Gnoise()** with different levels of noise, such as 5, 10, 15, and 20.
8. Display the original images and their corresponding noisy images. Something like this.

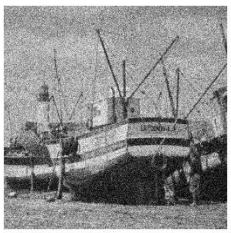
Madrill (original)



Mandrill (noisy)



9. Utilize the function `filter_image()` to apply the three mentioned filters to the noisy images.
10. Display the Boat noisy image with different levels of noise, and the filtered images as follows (only some images are shown here):

Lena (std=5)	Lena (std=10)	Lena (std=15)	Lena (std=15)
.....		.. ....	.. ....
Media F (PSNR=26)	Media F (PSNR=25)	Media F (PSNR=?)	Media F (PSNR=?)
...		....	....
Gaussian F (PSNR=?)	Gaussian F (PSNR=?)	Gaussian F (PSNR=?)	Gaussian F (PSNR=?)
....		....	....
Bilateral F (PSNR=?)	Bilateral F (PSNR=?)	Bilateral F (PSNR=?)	Bilateral F (PSNR=?)

....  ....

11. Redo 10 for other cases.
12. Call **add_SPnoise()**.
13. Redo questions 8, 9, and 10 and 11.
14. Develop a Python function called `smooth_image()` to apply Gaussian smoothing to the original Mandrill image using a kernel size parameter to control the degree of smoothing.
15. Construct a Python function named **laplace_sharpening()** to sharpen the smoothed Mandrill image by applying Laplace sharpening, with a kernel size parameter determining the extent of sharpening.
16. Present various smoothed and sharpened images, showcasing different parameter values.