**Image Denoising Project and PSNR calculation Report**

**Introduction**

Image denoising is a crucial task in image processing and computer vision, aimed at removing noise from images while preserving important details. Noise in images can arise from various sources, such as sensor noise, compression artifacts, or transmission errors. This project focuses on implementing a simple yet effective image denoising algorithm using a mean filter in Python, without relying on high-level APIs. The mean filter, also known as the average filter, is a basic technique used to smooth images and reduce noise by averaging the pixel values within a neighborhood around each pixel.

**Objectives**

The objectives of this project are:

1. To develop a basic understanding of image denoising techniques.
2. To implement a mean filter for image denoising using NumPy.
3. To create a complete pipeline to read an image, apply denoising, and save/display the results.
4. To document the entire process in detail for educational and reference purposes.

**Project Architecture**

The architecture of the project can be divided into several key components:

1. **Image Reading and Preprocessing:**
   * Load the input image.
   * Convert the image to an appropriate format for processing (e.g., NumPy array).
2. **Denoising Algorithm:**
   * Implement the mean filter algorithm.
   * Apply the mean filter to the image.
3. **Postprocessing and Output:**
   * Convert the denoised image back to an appropriate format.
   * Save the denoised image.
   * Display the original and denoised images side by side for comparison.

**Techniques Implemented**

**Mean Filter:** The mean filter is one of the simplest image denoising techniques. It works by averaging the pixel values within a neighborhood (kernel) around each pixel. The mean filter smooths out noise by replacing each pixel with the average value of its neighbors.

Steps in Mean Filter Implementation:

1. **Kernel Definition:**
   * Define a kernel (a small matrix) filled with ones and normalize it by dividing by the total number of elements in the kernel.
2. **Padding the Image:**
   * Pad the image to handle the borders. This ensures that the filter can be applied to all pixels, including those at the edges of the image.
3. **Applying the Filter:**
   * Slide the kernel over the image.
   * For each position of the kernel, calculate the average of the pixel values within the kernel and assign this value to the corresponding pixel in the output image.

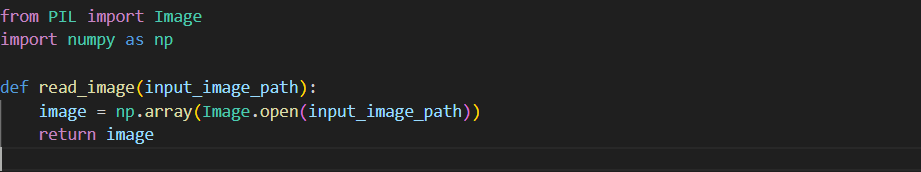
**Algorithm Implementation**

The mean filter was implemented using NumPy for efficient numerical computations. The key steps in the implementation are as follows:

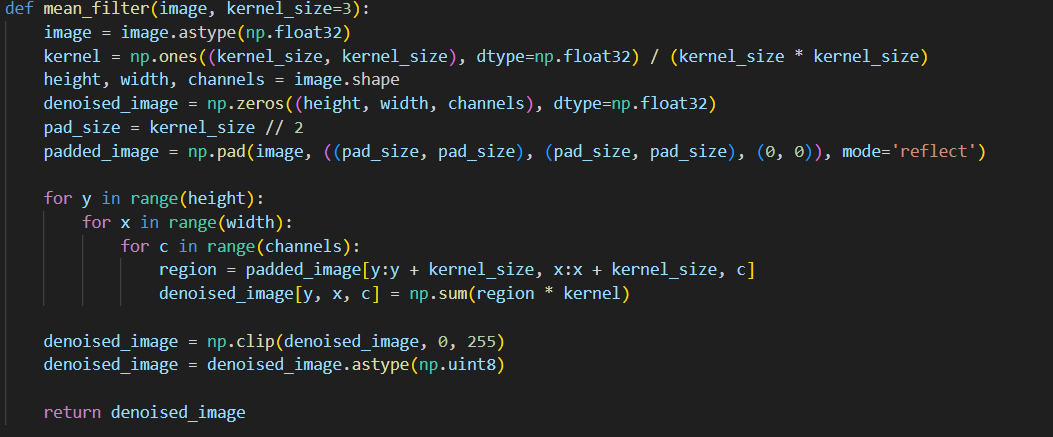
1. **Convert the Image to Float32:**
   * Converting the image to float32 format improves precision during calculations.
2. **Define the Kernel:**
   * Create a kernel filled with ones and normalize it.
3. **Pad the Image:**
   * Use the np.pad function to pad the image. Padding is done using the 'reflect' mode to handle the image borders.
4. **Apply the Kernel:**
   * Iterate over each pixel in the image.
   * For each pixel, extract the region corresponding to the kernel size.
   * Compute the average value of the region and assign it to the denoised image.
5. **Clip and Convert Back to uint8:**
   * Ensure that the pixel values in the denoised image are within the valid range (0 to 255).
   * Convert the denoised image back to uint8 format for saving and displaying.

**Pipeline in Detail**

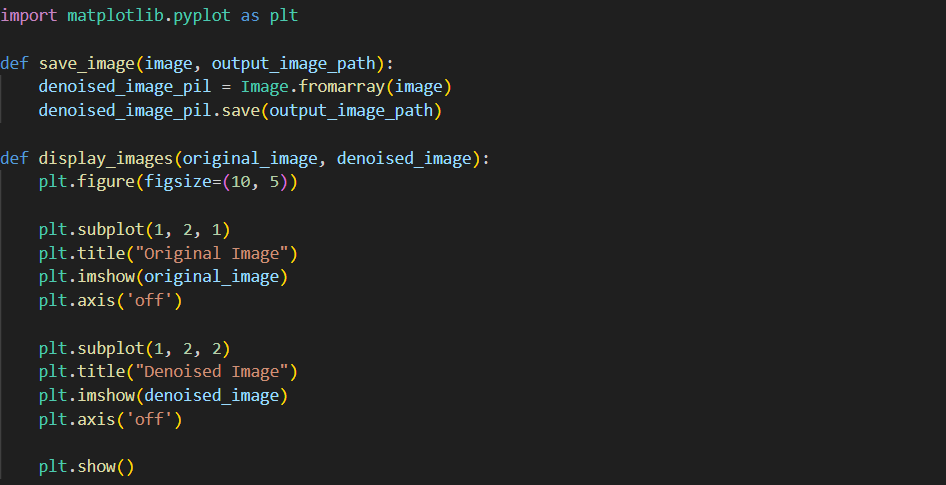
1. **Image Reading and Preprocessing:** The first step in the pipeline is to read the input image. The Python Imaging Library (PIL) is used for this purpose. The image is then



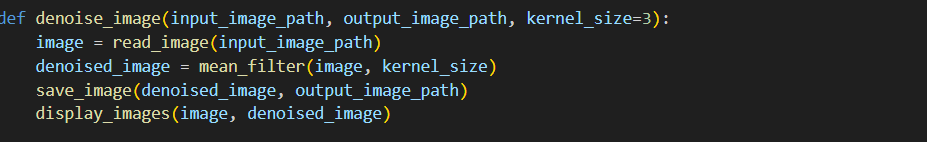
1. **Mean Filter Implementation:** The mean filter is implemented as a function that takes the image and kernel size as inputs and returns the denoised image.



**3)Postprocessing and Output:** The final steps involve saving the denoised image and displaying the results using Matplotlib.



**4) Full Pipeline Function:** Combining all the steps into a single function for ease of use.



**PSNR Calculation**

The PSNR (Peak Signal-to-Noise Ratio) is a metric used to measure the quality of a reconstructed or de-noised image compared to the original image. It is expressed in decibels (dB) and is defined as:

PSNR=20⋅log⁡10(MAXIMSE)\text{PSNR} = 20 \cdot \log\_{10} \left(\frac{\text{MAX}\_{I}}{\sqrt{\text{MSE}}}\right)PSNR=20⋅log10​(MSE​MAXI​​)

where:

* MAXI\text{MAX}\_{I}MAXI​ is the maximum possible pixel value of the image (usually 255 for 8-bit images).
* MSE\text{MSE}MSE (Mean Squared Error) is the average of the squares of the differences between the original and the de-noised images.

#### Results

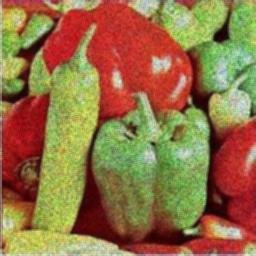
To test the implementation, an image named input.jpg was used. The mean filter with a kernel size of 3x3 was applied to the image. The denoised image was saved as output.jpg and displayed alongside the original image.

Original and Denoised Images: The following figures show the original and denoised images:

Original image:



Denoised image



### PSNR Value

The PSNR value for the de-noised image was approximately **28.49 dB**.

#### Conclusion

This project demonstrated the implementation of a simple image denoising algorithm using a mean filter. The algorithm was implemented from scratch using NumPy, and the entire pipeline from reading the image to saving and displaying the denoised image was documented. While the mean filter is a basic technique, it serves as a foundation for understanding more advanced denoising methods.

SUKHMANDEEP SINGH

22115148

3RD YEAR

Electrical engineering