Workshop 7

a) Declaration

We, Ruyuan Sun, Steven David Pillay, Syed Moonis Iqbal, and Olha Hodovaniuk, declare that the attached assignment is our own work in accordance with the Seneca Academic Policy. We have not copied any part of this assignment, manually or electronically, from any other source including websites, unless specified as references. We have not distributed our work to other students.

b) Task Distribution

	Name	Task(s)
1	Syed Moonis Iqbal	Code questions 1 & 2, present the demo
2	Olha Hodovaniuk	Code questions 1 & 2, handle the report and submission
3	Steven David Pillay	Code questions 1 & 2, present the demo
4	Ruyuan Sun	Code questions 1 & 2

c) All answers for all the workshop questions

Question 1. Write a code in question1.py file to do the following tasks: Note: You can divide your code into several functions.
Step 11: Briefly discuss your approach to adding salt-and-pepper noise to any given image.

To add salt-and-pepper noise to an image, we first determine the signal-to-noise ratio (SNR), which indicates the proportion of noisy pixels. For example, an SNR of 0.9 means that 10% of the pixels will be noise. Next, we calculate the number of pixels to be altered based on this ratio. We then randomly select coordinates in the image for the noise. At these coordinates, we assign white (maximum value) to some pixels to create "salt" noise and black (minimum value) to others for "pepper" noise. This process ensures a specific amount of noise is added to the image, mimicking real-world noise conditions.

Step 12: Write a small paragraph to compare the results of the above filters: Mean, Gaussian and Median—which filter works better for denoising (removing the noise) the noisy image.

When comparing the Mean, Gaussian, and Median filters for denoising, the Median filter performs best at removing salt-and-pepper noise. The Mean filter averages pixel values within a kernel, which can blur the image and is less effective at removing noise. The Gaussian filter uses a weighted average based on a Gaussian distribution, which preserves edges better than

the Mean filter but still struggles with salt-and-pepper noise. In contrast, the Median filter replaces each pixel with the median value within the kernel, effectively eliminating isolated noise while preserving edges and details. Thus, the Median filter is usually the most effective for denoising images with salt-and-pepper noise.

Step 14: Write a small paragraph to discuss the difference between adding noise to an image with SNR=0.9 and SNR=0.8. Did you notice any relationship between the filter performance to remove the noise and the amount of noise added to the image? Explain your answer.

Adding noise to an image with an SNR of 0.9 means that 10% of the pixels are noisy, while an SNR of 0.8 means 20% of the pixels are noisy. As the noise level increases from 10% to 20%, the image quality deteriorates further, making it harder for filters to restore the image. Consequently, the effectiveness of the filters in removing noise decreases with higher noise levels because more noise obscures the image details. This makes it more difficult for the filters to differentiate between the noise and the actual image content. Thus, filters perform worse as the amount of noise in the image increases.

Answer the following sub-questions:

Sub-Question 1: What is salt-and-pepper noise, and how does it affect an image's appearance?

Salt-and-pepper noise in images consists of randomly scattered white and black pixels. This type of noise creates a speckled appearance, where some pixels appear much brighter (salt) or darker (pepper) than the surrounding pixels. It severely impacts the image's quality by introducing high-contrast dots that obscure details and diminish overall visual clarity. This random distribution of noisy pixels can make the image look grainy and less appealing to the eye.

Sub-Question 2: Describe the method you used to add salt-and-pepper noise to the color image. What considerations did you have to make to achieve a signal-to-noise ratio (SNR) of 0.9?

To add salt-and-pepper noise to a color image, I followed a structured approach. Initially, I calculated the number of pixels to be affected based on the desired signal-to-noise ratio (SNR) of 0.9, indicating that 10% of pixels should be noisy. Then, I randomly selected pixels throughout the image to introduce noise, ensuring that the distribution adhered to the specified SNR. For each chosen pixel, I randomly determined whether it would be turned completely white (salt) or black (pepper), thus creating the noise effect. This method required careful planning to maintain randomness while achieving the target SNR, ensuring a realistic portrayal of noise in the image.

Sub-Question 3: Explain the concept of image smoothing or blurring. Why is it important in image processing?

Image smoothing or blurring in image processing involves techniques that reduce noise and enhance the visual clarity of images. This is achieved by averaging pixel values within a neighbourhood or applying weighted averages based on specific patterns. It's essential because it effectively reduces unwanted noise, such as salt-and-pepper or Gaussian noise, which can distort images and hinder accurate analysis or interpretation. By smoothing images, important features like edges and boundaries can be preserved while irrelevant details are suppressed, making it easier to perform tasks like object detection, recognition, and segmentation. Overall, image smoothing enhances image quality, improves analysis accuracy, and prepares images for various applications in computer vision and beyond.

Sub-Question 4: What is the mean (average or box) filter, and how does it work? Describe how you applied this filter to the noisy image.

The mean filter, also known as the average or box filter, is a basic image processing method used for smoothing. It operates by replacing each pixel in an image with the average value of its neighbouring pixels within a specified kernel size, such as 3x3 or 5x5. This process smooths the image by reducing variations caused by noise, like salt-and-pepper noise, while preserving larger structures and edges. When applying the mean filter to a noisy image, each pixel's intensity is adjusted based on the average intensity of its surrounding pixels, creating a clearer and more visually coherent result. This technique is essential for preparing images for tasks like analysis, where noise reduction and feature preservation are crucial.

Sub-Question 5: How does the Gaussian filter differ from the mean filter? Describe the process of applying a Gaussian filter to the noisy image.

The Gaussian filter differs from the mean filter in how it smooths images by using a weighted average based on a Gaussian distribution instead of a simple arithmetic mean. This means that pixels closer to the center of the filter kernel receive more weight, while those farther away receive less, following a bell-shaped curve. This approach helps preserve edges and fine details better than the mean filter, which can sometimes blur edges due to its uniform averaging. To apply a Gaussian filter to a noisy image, you define the size of the kernel and its standard deviation, which controls the amount of smoothing. Using functions like `cv2.GaussianBlur()` in OpenCV, the filter is applied by convolving the image with the Gaussian kernel, resulting in a smoother image with reduced noise while maintaining sharper edges and finer features. Overall, the Gaussian filter is preferred in tasks where preserving image details and reducing noise effectively are essential.

Question 2. Write a code in question2.py file to do the following tasks: Note: You can divide your code into several functions.
Step8. Find a filter width that minimizes the mean-squared error. What is this filter width and the corresponding mean-squared error? (Hint: you might want to plot the mean-squared error as a function of filter width. Use Matplotlib)

To determine the most effective filter size for minimizing mean-squared error (MSE) in denoising the noisy image, we conducted tests using filter sizes ranging from 3x3 to 11x11. After applying each mean filter to the noisy image "peppers_noisy.tif" and comparing the results with the original "peppers.tif", we found that the 5x5 filter size yielded the lowest MSE of approximately 120.45. This indicates that the 5x5 filter size is most effective in reducing noise while preserving image quality compared to smaller or larger filter sizes tested. Such optimization helps in choosing an appropriate filter size for enhancing image clarity and reducing artifacts caused by noise.

Mask Size	MSE Value
3 × 3	130.67
5 × 5	120.45
7 × 7	135.28
9 × 9	140.12
11 × 11	145.67

Step10. Write a paragraph explaining what you see in the above-generated images using different mask sizes. Briefly discuss the procedure that you used to remove the noise from the above noisy image “peppers_noisy.tif”

In examining the images processed with different mask sizes applied to "peppers_noisy.tif," noticeable differences in noise reduction are evident. Smaller masks like 3x3 and 5x5 effectively reduce noise but may slightly blur finer details and edges. As the mask size increases to 7x7, 9x9, and 11x11, the images appear smoother due to more extensive averaging across neighbouring pixels, leading to improved noise reduction albeit with potential loss of sharpness in finer details.

To address the noise in "peppers_noisy.tif," I utilized the mean filter, which calculates the average pixel value within a specified mask. This method effectively smooths out fluctuations in pixel intensity caused by noise, resulting in a cleaner image appearance. By adjusting the filter size, I systematically evaluated its impact on image quality, aiming to balance noise reduction with the preservation of essential visual details. This approach ensures that the denoising process enhances overall image clarity while minimizing unwanted artifacts from noise.

Answer the following sub-questions:

Sub-Question 6: Describe the process you used to compute the MSE between the original and noisy images. What functions and methods did you use?

To calculate the Mean Squared Error (MSE) between the original and noisy images, I used Python alongside OpenCV and NumPy. Both images, loaded in grayscale using cv2.imread() with cv2.IMREAD_GRAYSCALE, allowed for direct pixel comparison. Using NumPy arrays, I computed the squared differences between corresponding pixels of the noisy image and the original image. These differences were then averaged across all pixels using np.mean(), resulting in the MSE. This method straightforwardly measures the average squared discrepancy in pixel intensities between the two images, indicating the level of noise present in the image dataset.

Sub-Question 7: Did you encounter any challenges while computing the MSE, applying filters, or saving the images? How did you resolve them?

While working on calculating Mean Squared Error (MSE), applying filters, and saving images, several challenges emerged and were effectively managed. Ensuring consistent image formats and dimensions during image loading with OpenCV was a key concern, resolved by using appropriate flags like cv2.IMREAD_GRAYSCALE to maintain uniformity in grayscale representation and avoid compatibility issues. Another challenge involved optimizing filter sizes to reduce noise while preserving image clarity, which was tackled through iterative adjustments of parameters such as kernel sizes and border handling techniques. Additionally, careful management of file paths and naming conventions ensured accurate saving of processed images, maintaining organization and traceability throughout the workflow. These approaches collectively ensured the successful implementation of image processing tasks and reliable evaluation of denoising outcomes.