EE431 Intro. to Image & Video Processing

HW2

Figures

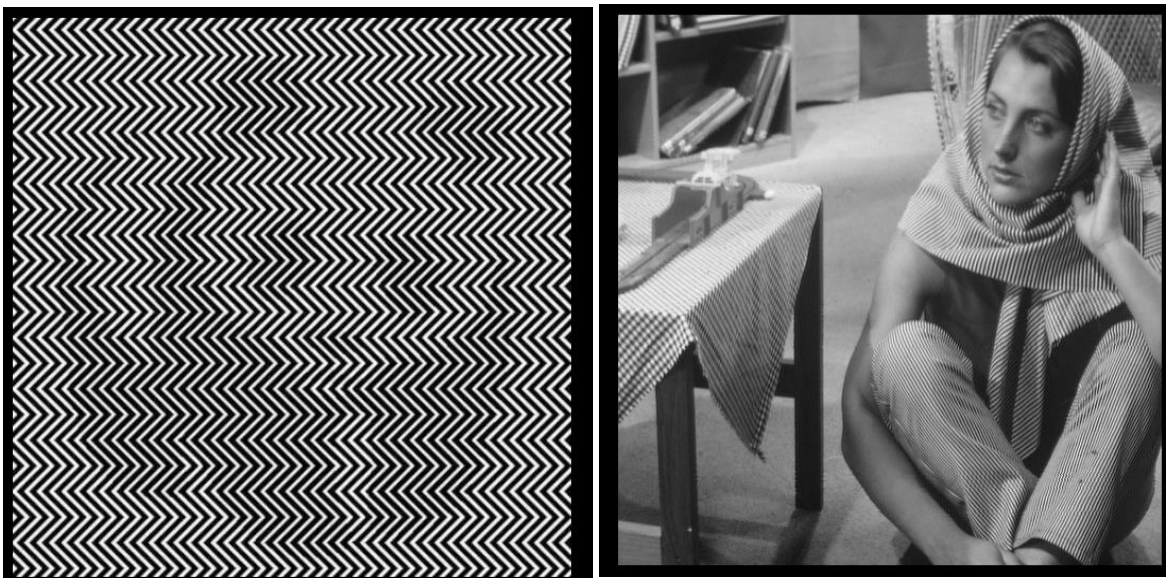

Figure 1,2 : Panda and Barbara Images

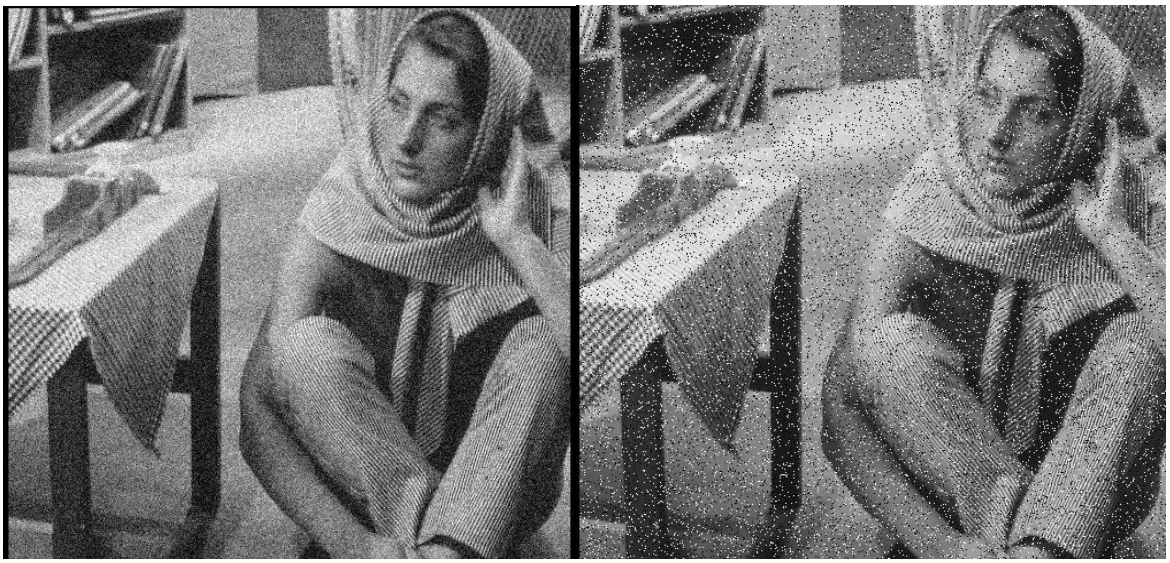

Figure 3,4 : Barbara with noise and barbara with salt pepper Images

Gauss Filter Part

Gaussfilter.C code implements a Gaussian filter for image processing as part of a homework. The gauss function is designed to apply a 3x3 Gaussian filter to an input image iteratively. The function takes as parameters a 2D array (img) representing the image, the number of columns (NC), the number of rows (NR), and the iteration count (count). The Gaussian filter is defined by a 3x3 mask initialized with values {{1, 2, 1}, {2, 4, 2}, {1, 2, 1}}.

The function utilizes a nested loop structure to iterate over the image pixels, convolving them with the Gaussian mask. The convolution is performed count times. To optimize memory usage, the code employs a pointer swapping technique, where pointers to the input image (img) and the result image (result) are swapped in each iteration. This avoids unnecessary copying of image data.

After the specified number of iterations, the final result is copied back to the original image array if the iteration count is odd. Memory cleanup is performed using the free_img function, and the processed image is returned.

The main function serves as the entry point for the program. It checks for the correct number of command-line arguments, reads the input image from a PGM file specified by the user, and saves the original image to a new PGM file ("Im1.pgm"). The Gaussian filter is then applied to the image using the gaussf function, and the filtered image is saved to another PGM file ("Im2.pgm"). Both the original and filtered images are displayed, and memory is freed to prevent memory leaks.

This Gaussian filter implementation contributes to the broader objective of the homework assignment, which involves creating a program capable of performing Gaussian filtering and median filtering on input images based on user-specified parameters.

Results of Gauss Filters

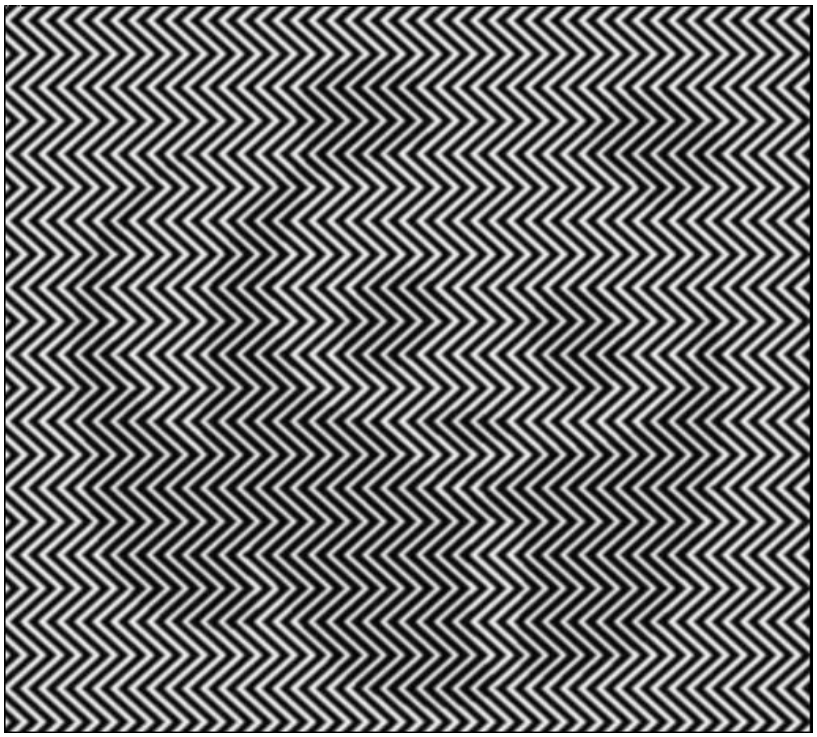

Figure 5: Gauss filtered Panda Image

Figure 6: Gauss Filtered Barbara Image

Figure 7: Gauss Filtered Barbara with salt pepper Image

Figure 8: Gauss Filtered Barbara with Noise Image

Median Filter Part

MedianFilter.C code implements a median filter as part of a program for image processing, as specified in the assignment. The medianf function is designed to apply a 3x3 median filter to an input image iteratively. It takes as parameters a 2D array (img) representing the image, the number of columns (NC), the number of rows (NR), and the iteration count (count). The median filter operation is performed count times on the image.

Within the function, a nested loop structure iterates over the pixels of the image. For each pixel, a 3x3 neighborhood is considered, and the pixel values within this neighborhood are stored in an array called values. A simple sorting algorithm is then applied to this array to arrange the pixel values in ascending order. The median value, which is the value at index 4 in the sorted array (considering zero-based indexing), is assigned to the corresponding pixel in the result image.

To optimize memory usage, the code utilizes a pointer swapping technique, where pointers to the input image (img) and the result image (result) are swapped in each iteration. This avoids unnecessary copying of image data. After the specified number of iterations, the unused result array is freed using the free_img function, and the processed image is returned.

The main function, which serves as the entry point for the program, checks for the correct number of command-line arguments, reads the input image from a PGM file specified by the user, and saves the original image to a new PGM file ("Im1.pgm"). The median filter is then applied to the image using the medianf function, and the filtered image is saved to another PGM file ("Im3.pgm"). Both the original and filtered images are displayed, and memory is freed to prevent memory leaks.

Results of Median Filter

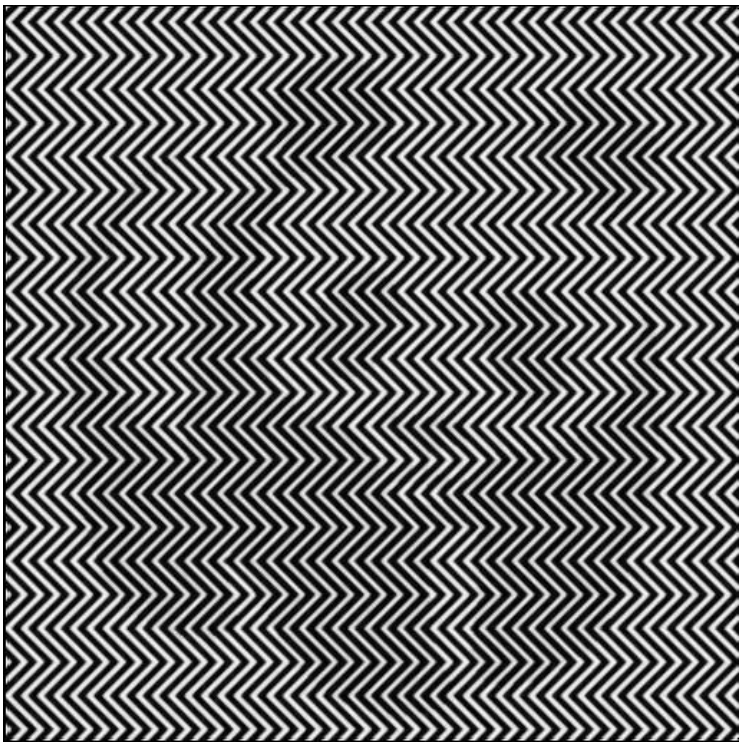

Figure 9: Median Filtered Panda Image

Figure 10: Median Filtered Barbara Image

Figure 11: Median Filtered Barbara with salt pepper Image

Figure 12: Median Filtered Barbara with noise Image

Conclusion

In conclusion, the provided C codes constitutes a comprehensive solution to the assigned homework, addressing the implementation of both Gaussian and median filters for image processing. The Gaussian filter, implemented within the gaussf function, applies a 3x3 convolution operation count times, effectively smoothing the input image. The code demonstrates a memory-efficient approach using pointer swapping to minimize unnecessary data copying during each iteration. Furthermore, the implementation follows the specified vertical and horizontal convolution sequence with a predefined 3x3 Gaussian mask.

The median filter, implemented in the medianf function, employs a 3x3 neighborhood approach to replace each pixel with the median value within its local region. The code efficiently handles iterations using pointer swapping, contributing to improved memory management. The sorting algorithm utilized for finding the median within the neighborhood ensures that the processed image retains relevant structural information while reducing the impact of noise.

In the main function, the program reads a specified PGM image file and a user-defined count parameter from the command line. It then applies both the Gaussian and median filters to create a set of processed images. The original image, the Gaussian-filtered version, and the median-filtered version are displayed side by side, providing a visual representation of the effects of these filtering operations on the input image.

Overall, this homework assignment not only demonstrates the successful implementation of Gaussian and median filters but also showcases good coding practices, including efficient memory management and modular function design. The combination of both filters enables the program to offer a versatile image processing tool, allowing users to explore the effects of smoothing and noise reduction on different types of images.