Artificial Vision and Pattern Recognition Laboratory Report
Image Processing Techniques Analysis

Audigier Matteo

October 15, 2025

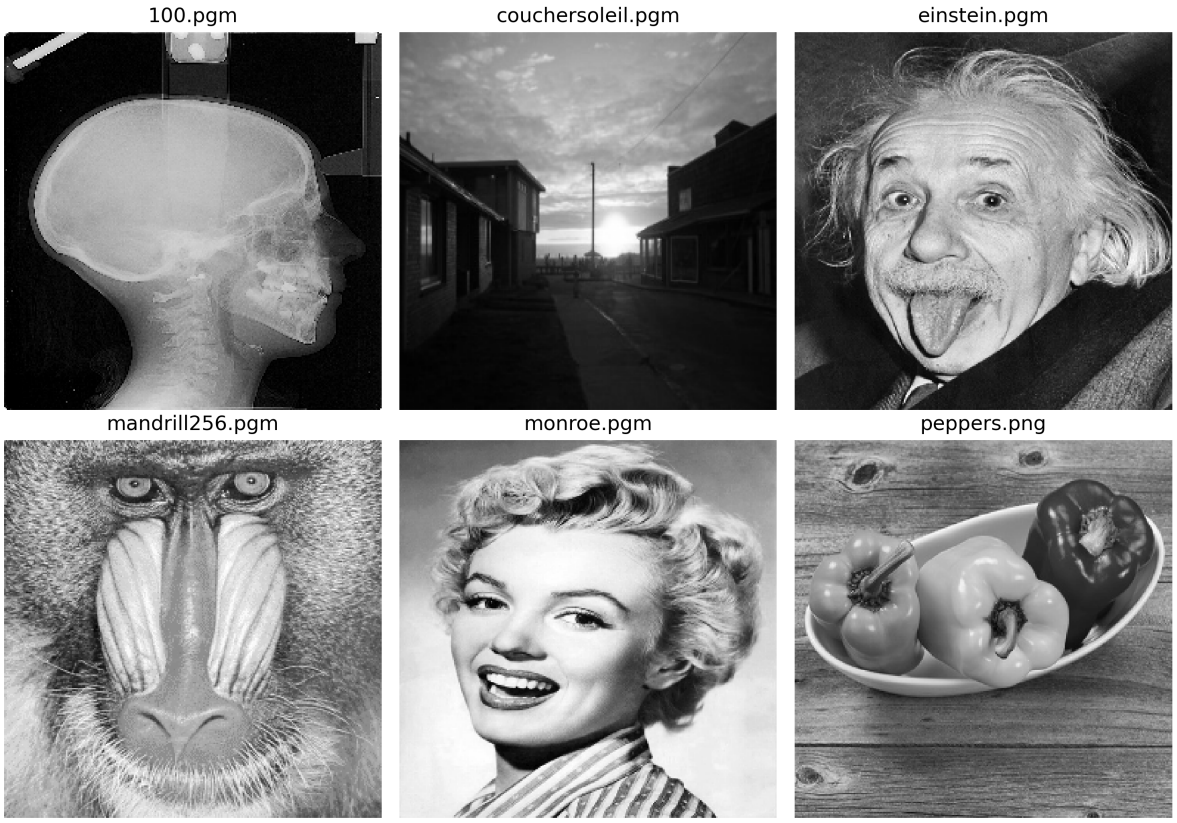
Contents

1 Introduction

This report presents a comprehensive analysis of various image processing techniques implemented and evaluated as part of the Artificial Vision and Pattern Recognition laboratory work. The primary objectives were to explore edge detection methods, custom convolution kernels, noise addition and reduction strategies, motion blur simulation, and frequency domain filtering using the Fourier Transform.

The experiments were conducted using Python and OpenCV, with results visualized through Matplotlib. The images used for testing were sourced from the provided assets directory and from lab sessions of the previous year at my home university.

The source code for all implementations is available in the GitHub repository: `https://github.com/audigiem/AVPR_labs`. Here are all the images used for the experiments:



2 Task 1: Edge Detection Comparison

2.1 Theoretical Background

Edge detection is a crucial operation in image processing that identifies boundaries within images where brightness changes sharply. Three classical operators were implemented and compared:

2.1.1 Sobel Operator

The Sobel operator uses two 3×3 kernels to approximate derivatives in horizontal and vertical directions:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{1}$$

2.1.2 Prewitt Operator

The Prewitt operator is similar to Sobel but with uniform weighting:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{2}$$

2.1.3 Scharr Operator

The Scharr operator provides improved rotational symmetry:

$$G_x = \begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix}, \quad G_y = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix} \tag{3}$$

The edge magnitude is computed as:

$$|G| = \sqrt{G_x^2 + G_y^2} \tag{4}$$

2.2 Implementation

The implementation process followed these steps:

1. Load grayscale images from the assets directory
2. Apply each operator using OpenCV functions (cv2.Sobel, cv2.filter2D)
3. Compute magnitude by combining horizontal and vertical components
4. Normalize results for visualization
5. Generate comparative visualizations

2.3 Results and Analysis

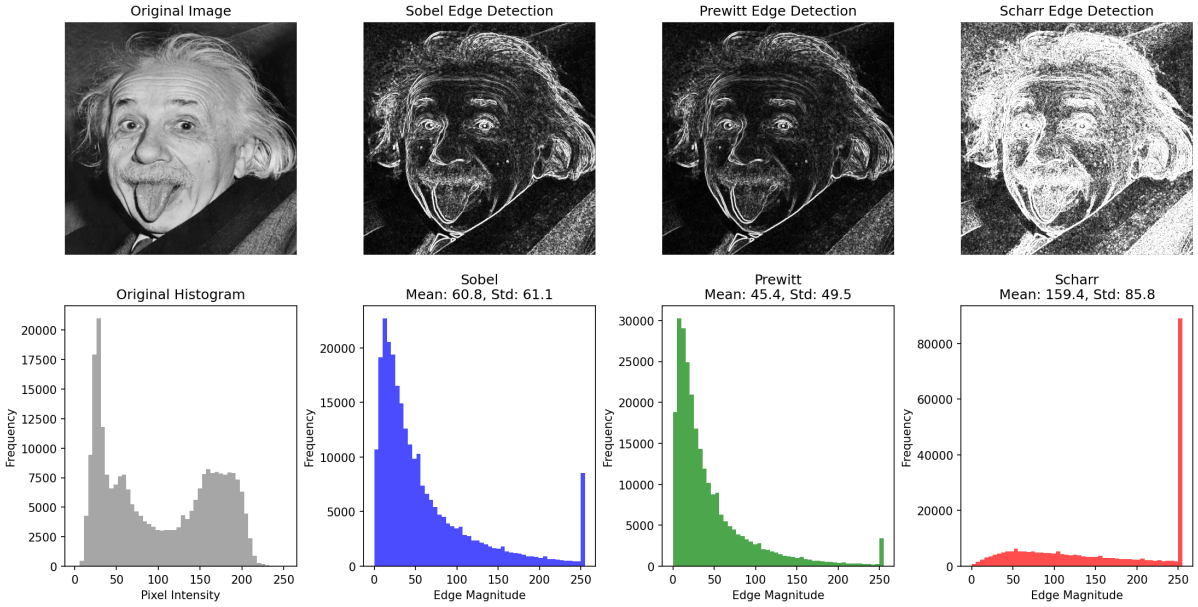


Figure 1: Edge detection comparison on Einstein image showing original and results from Sobel, Prewitt, and Scharr operators

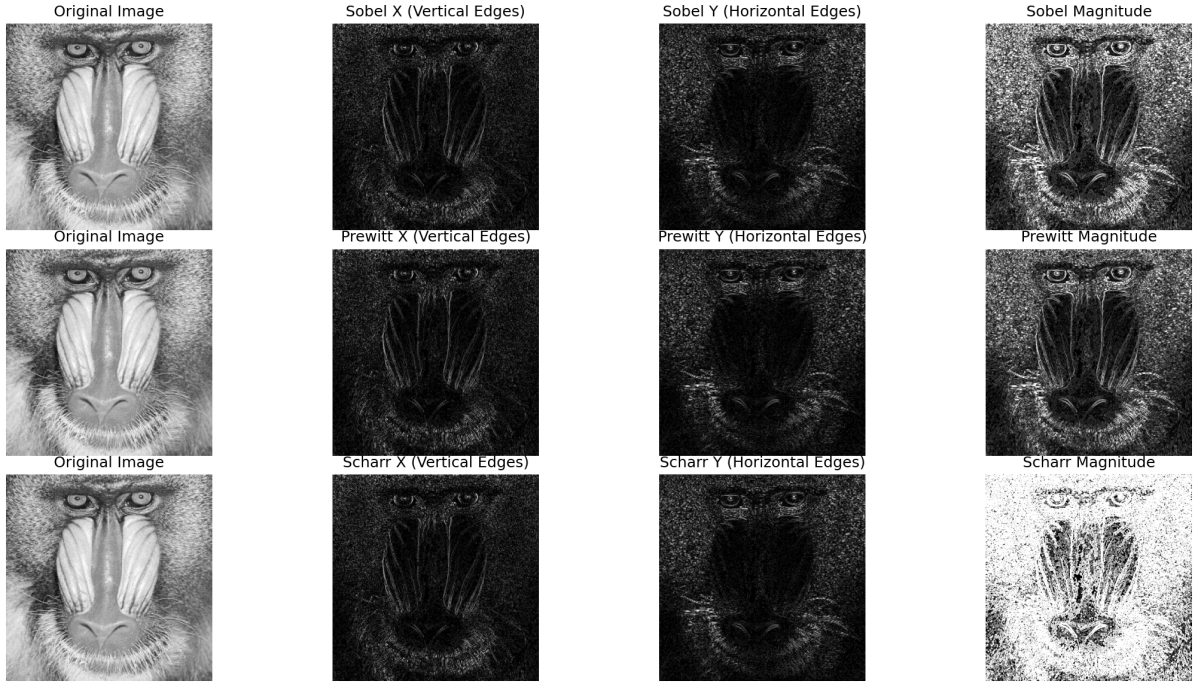


Figure 2: Detailed edge detection comparison on Mandrill image showing horizontal, vertical, and magnitude components

2.3.1 Comparative Analysis

The experimental results reveal several key observations:

- **Sobel Operator:** Provides balanced edge detection with good noise suppression due to its smoothing effect. Most widely used in practice.
- **Prewitt Operator:** Similar to Sobel but with slightly less noise suppression. Simpler computation but comparable results.
- **Scharr Operator:** Produces the strongest edge responses with better angular accuracy, particularly effective for detecting diagonal edges. The higher kernel coefficients result in more pronounced edge detection.

For images with complex textures (like Mandrill), Scharr produces the most detailed edge maps. For smoother images (like Monroe), all three operators perform similarly, with Scharr showing slightly higher sensitivity.

3 Task 2: Custom Kernel Design and Application

3.1 Theoretical Background

Convolution kernels are fundamental tools in image processing that enable various filtering operations. A kernel is a small matrix that slides over the image, computing weighted sums of pixel neighborhoods.

3.1.1 Kernel Definitions

Four custom kernels were designed and implemented:

1. **Sharpening Kernel:**
$$K_{sharp} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{5}$$
Enhances edges and fine details by accentuating high-frequency components.

2. **Edge Enhancement Kernel:**
$$K_{edge} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{6}$$
Detects and emphasizes edges in all directions.

3. **Emboss Kernel:**
$$K_{emboss} = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix} \tag{7}$$
Creates a 3D relief effect by simulating directional lighting.

4. **Custom Blur Kernel:**
$$K_{blur} = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{8}$$
Gaussian-like smoothing with normalized weights.

3.2 Implementation

The kernels were applied using cv2.filter2D() function, which performs 2D convolution:

$$I_{out}(x, y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} K(i, j) \cdot I_{in}(x + i, y + j) \tag{9}$$

3.3 Results and Analysis

Figure 3: Custom kernel effects on Monroe image: original, sharpening, edge enhancement, emboss, and custom blur

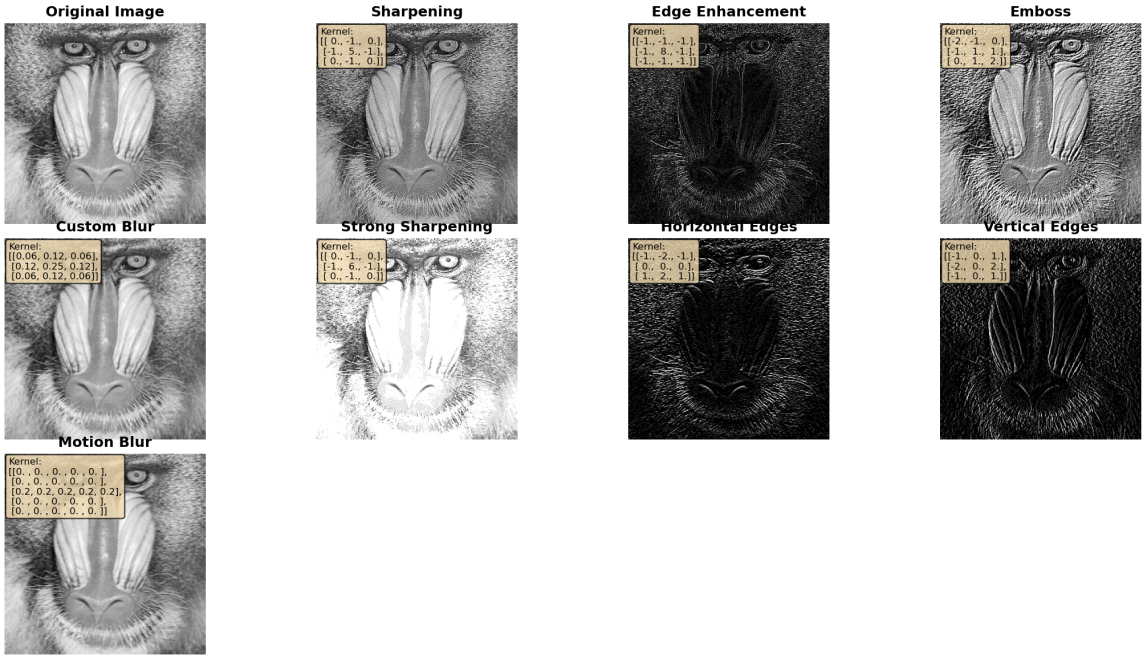


Figure 4: Custom kernel effects on Mandrill image showing detailed texture responses

3.3.1 Observations

- **Sharpening:** Successfully enhances details and makes edges crisper. Particularly effective on images with fine structures. May amplify noise in uniform regions.
- **Edge Enhancement:** Produces strong edge maps similar to Laplacian filtering. Highlights boundaries while suppressing constant regions. Useful for feature extraction.
- **Emboss:** Creates artistic 3D relief effects. The directional nature of the kernel produces illumination from top-left. Gray background represents zero gradient areas.
- **Custom Blur:** Provides smooth gradual blurring while preserving overall structure. The Gaussian-like distribution ensures no artifacts. Effective for noise reduction while maintaining edges better than box filters.

Kernel magnitude directly affects the strength of the effect. Increasing central weight in sharpening kernels produces more aggressive enhancement, while larger blur kernels increase smoothing radius.

4 Task 3: Noise Addition and Reduction Analysis

4.1 Theoretical Background

Image noise degrades quality and can arise from various sources including sensor limitations, transmission errors, and environmental factors. This task focuses on salt-and-pepper noise, characterized by random white and black pixels.

4.1.1 Noise Reduction Filters

Four filtering techniques were evaluated:

1. **Median Filter:** Replaces each pixel with the median value of its neighborhood. Highly effective for impulse noise while preserving edges.

2. **Gaussian Filter:** Applies weighted averaging based on Gaussian distribution:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{10}$$

3. **Bilateral Filter:** Preserves edges by considering both spatial and intensity differences:

$$BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(|I_p - I_q|) I_q \tag{11}$$

4. **Non-Local Means (NLM):** Exploits patch similarity across the image for denoising.

4.1.2 Quality Metric

Peak Signal-to-Noise Ratio (PSNR) quantifies restoration quality:

$$PSNR = 10 \log_{10} \left(\frac{MAX_I^2}{MSE} \right) \tag{12}$$

where MAX_I is the maximum pixel value (255) and MSE is mean squared error.

4.2 Results and Analysis

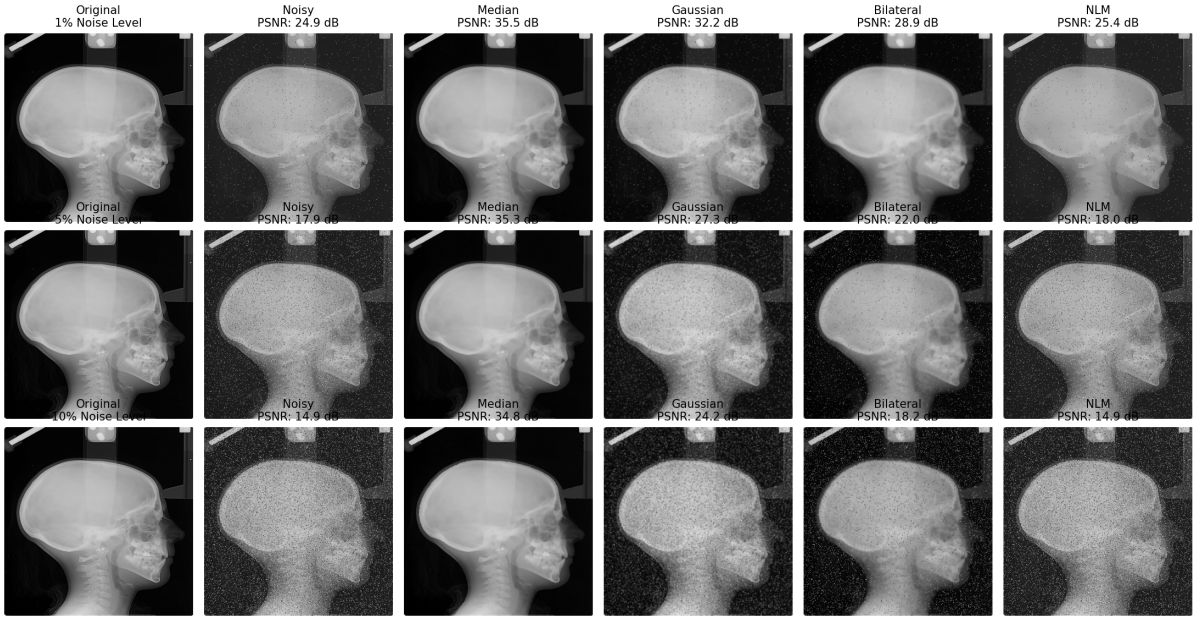


Figure 5: Noise reduction comparison at different noise levels (1%, 5%, 10%) with PSNR values

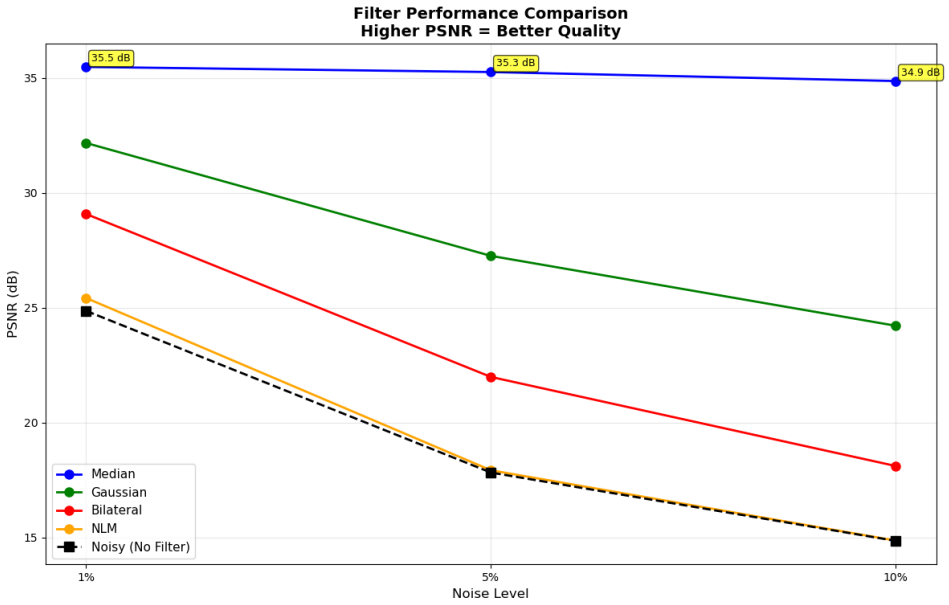


Figure 6: PSNR performance comparison across different noise levels and filtering methods

4.2.1 Analysis

- **Median Filter:** Consistently achieves the highest PSNR values for salt-and-pepper noise. Excellent at removing impulse noise while preserving edges. Performance degrades gracefully with increasing noise levels.
- **Gaussian Filter:** Shows poorest performance due to its averaging nature, which blurs both noise and edges. Not recommended for impulse noise but effective for Gaussian noise.
- **Bilateral Filter:** Provides good balance between noise reduction and edge preservation. Performs better than Gaussian but slightly worse than median for this noise type. Computationally more expensive.
- **Non-Local Means:** Competitive performance with good detail preservation. Particularly effective at moderate noise levels. Highest computational cost but produces visually pleasing results.

At 1% noise, all filters perform adequately. At 10% noise, median filtering clearly outperforms alternatives. For real-world applications, median filter is recommended for salt-and-pepper noise, while bilateral or NLM are better suited for Gaussian noise with edge preservation requirements.

5 Task 4: Motion Blur Simulation and Analysis

5.1 Theoretical Background

Motion blur occurs when there is relative movement between the camera and scene during exposure. It can be mathematically modeled as convolution with a motion blur kernel representing the trajectory of movement.

5.1.1 Motion Blur Kernel

A motion blur kernel for angle θ and length L is constructed by:

1. Creating a line of non-zero values along the motion direction
2. Rotating the kernel to the desired angle
3. Normalizing to preserve image brightness

The kernel effectively integrates along the motion path:

$$K_{motion}(x, y) = \begin{cases} \frac{1}{L} & \text{if } (x, y) \text{ lies on motion path} \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

5.2 Implementation

Motion blur kernels were generated for:

- Four angles: 0°, 45°, 90°, 135°
- Multiple kernel sizes: 15, 20, 30, 50 pixels
- Diagonal combinations (horizontal + vertical)

5.3 Results and Analysis

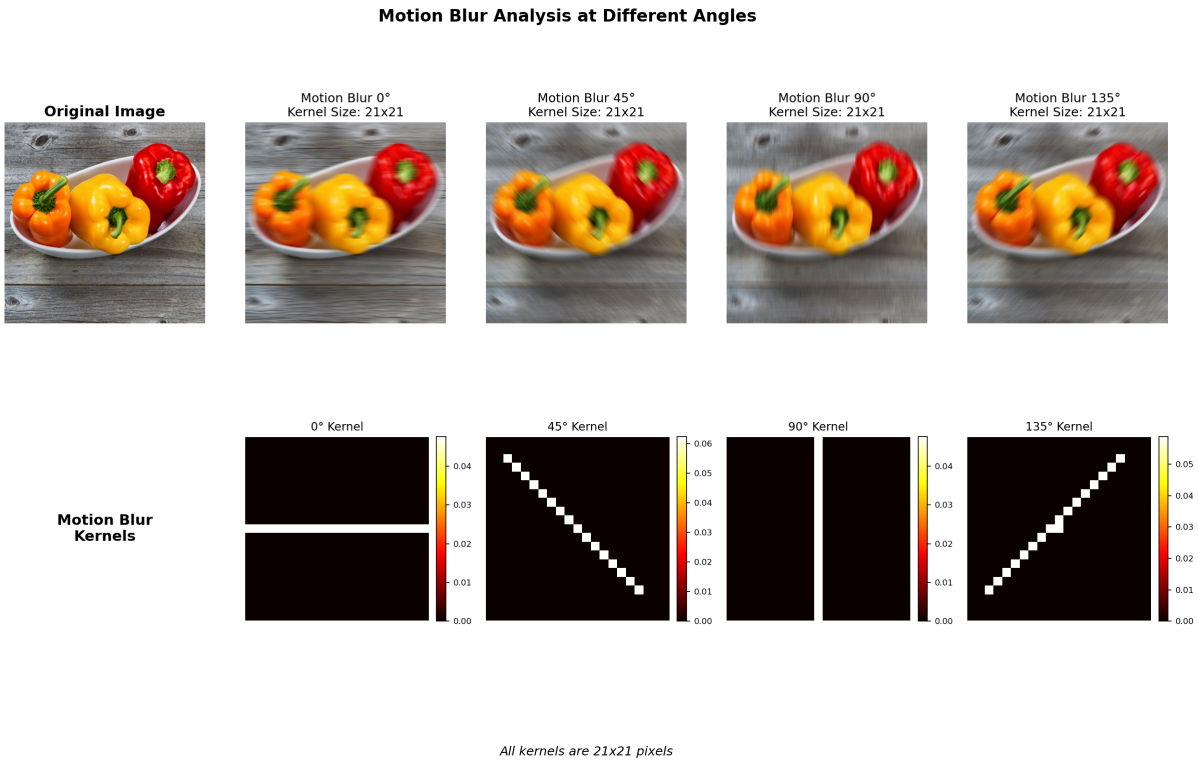


Figure 7: Motion blur at different angles (0°, 45°, 90°, 135°) with corresponding kernels

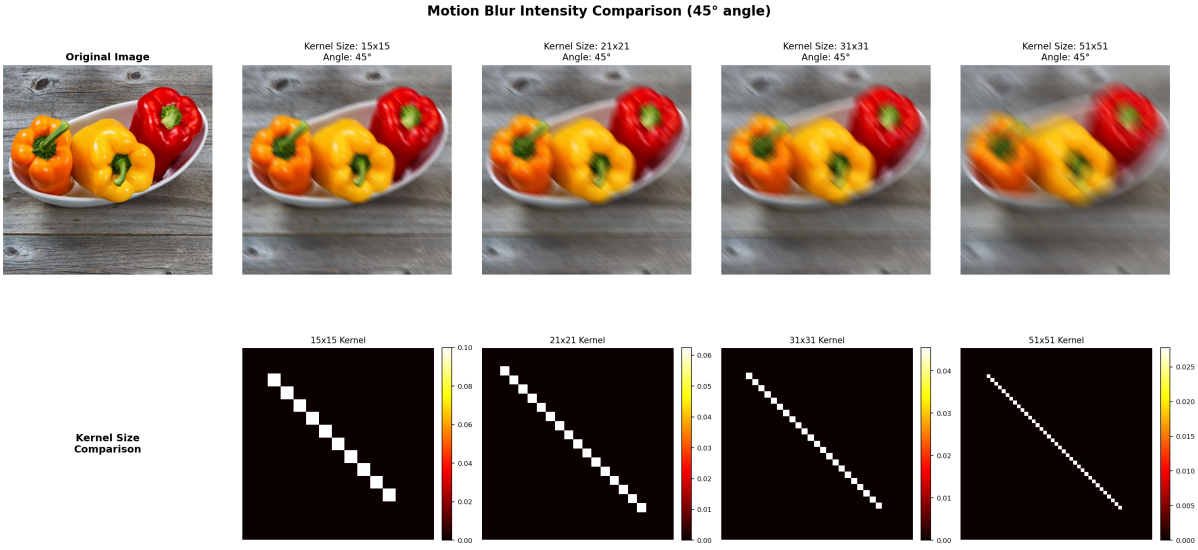


Figure 8: Effect of kernel size on motion blur intensity (size 15, 30, 50)

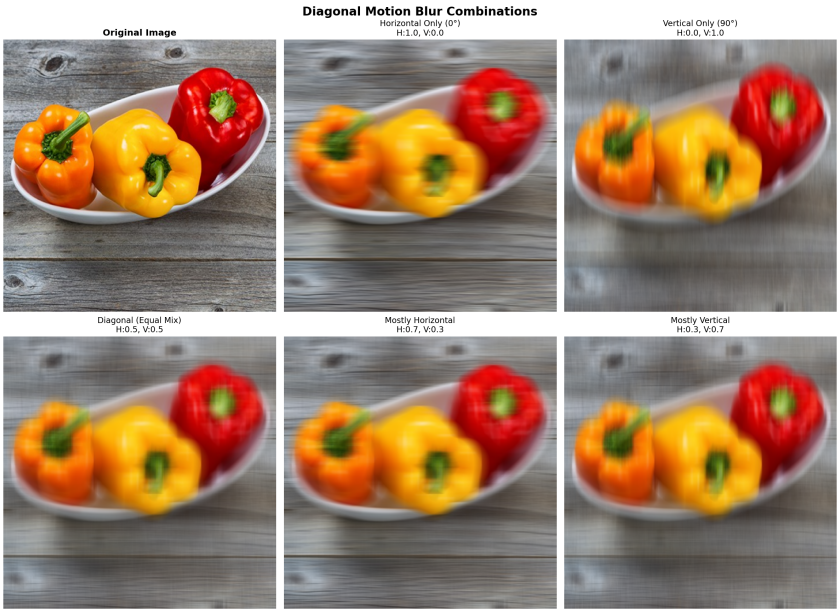


Figure 9: Diagonal motion blur created by combining horizontal and vertical kernels

5.3.1 Observations

- **Angle Dependency:** The blur direction clearly follows the kernel orientation. Horizontal blur (0°) streaks horizontally, vertical (90°) streaks vertically, and diagonal blurs show intermediate directions.
- **Size Effects:** Larger kernels produce more pronounced blur. At size 15, blur is subtle. At size 50, details are significantly smeared. Linear relationship between kernel size and perceived blur strength.
- **Diagonal Combination:** Combining orthogonal motions creates complex blur patterns. Sequential application differs from simultaneous diagonal blur, showing

interesting interference effects.

- **Realistic Simulation:** The motion blur kernels effectively simulate camera shake and object motion. Useful for understanding blur formation mechanisms and testing deblurring algorithms.

Edge structures perpendicular to motion direction are most affected, while parallel edges remain relatively sharp. This anisotropic nature is characteristic of motion blur and distinguishes it from isotropic blurs like Gaussian.

6 Task 5: Frequency Domain Filtering with Fourier Transform

6.1 Theoretical Background

Frequency domain analysis decomposes images into sinusoidal components of varying frequencies. The 2D Fourier Transform is defined as:

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(\frac{ux}{M} + \frac{vy}{N})} \tag{14}$$

The magnitude spectrum represents frequency content:

$$|F(u,v)| = \sqrt{\text{Re}(F(u,v))^2 + \text{Im}(F(u,v))^2} \tag{15}$$

6.1.1 Filtering in Frequency Domain

1. **Low-Pass Filter:** Removes high frequencies (noise, fine details), preserves low frequencies (overall structure). Implemented using circular mask:

$$H_{LP}(u,v) = \begin{cases} 1 & \text{if } \sqrt{u^2+v^2} \leq D_0 \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

2. **High-Pass Filter:** Removes low frequencies (background, illumination), preserves high frequencies (edges, textures):

$$H_{HP}(u,v) = 1 - H_{LP}(u,v) \tag{17}$$

6.2 Implementation

The filtering process follows these steps:

1. Compute 2D FFT: F = np.fft.fft2(image)
2. Shift zero frequency to center: F_shift = np.fft.fftshift(F)
3. Apply filter mask: F_filtered = F_shift * H(u,v)
4. Inverse shift and transform: image_filtered = np.fft.ifft2(np.fft.ifftshift(F_filtered))

6.3 Results and Analysis

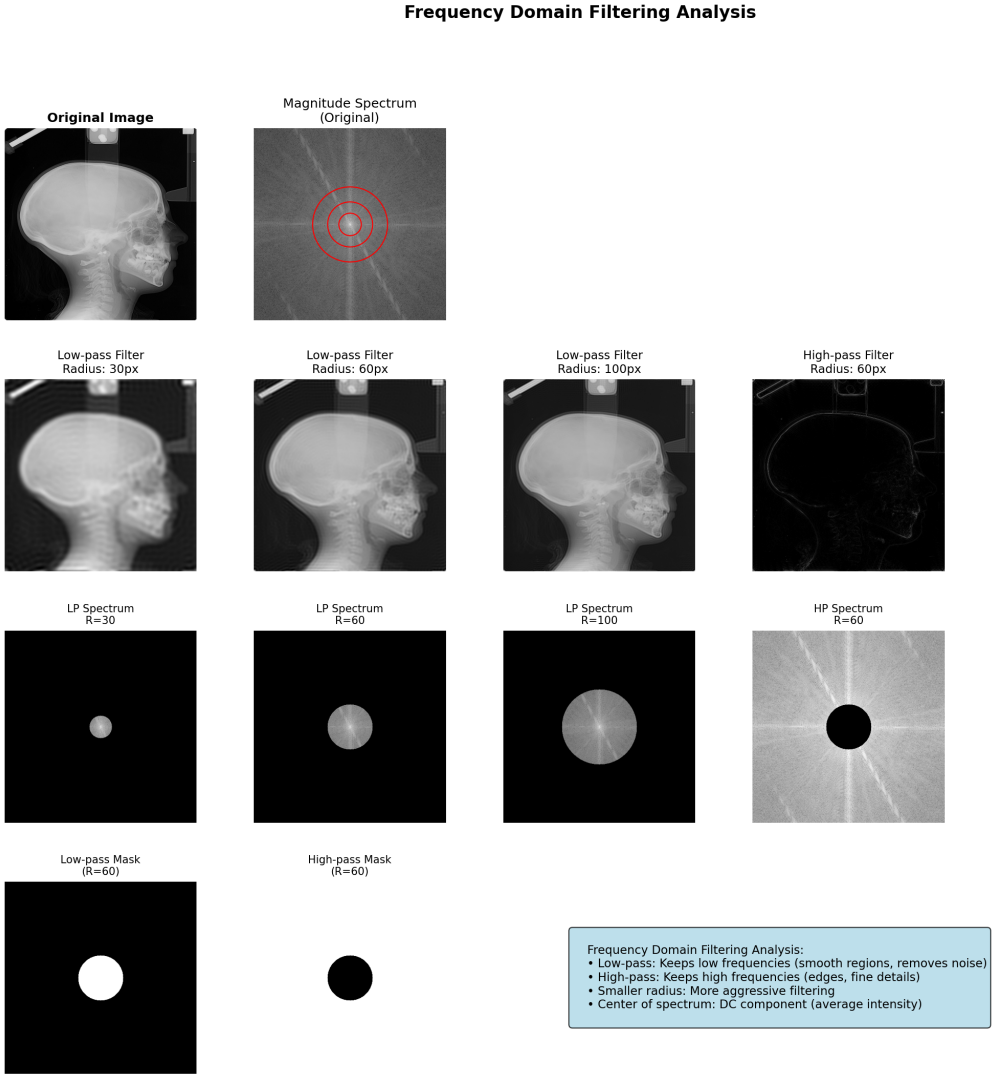


Figure 10: Frequency domain analysis showing original image, magnitude spectrum, and low/high-pass filtering results

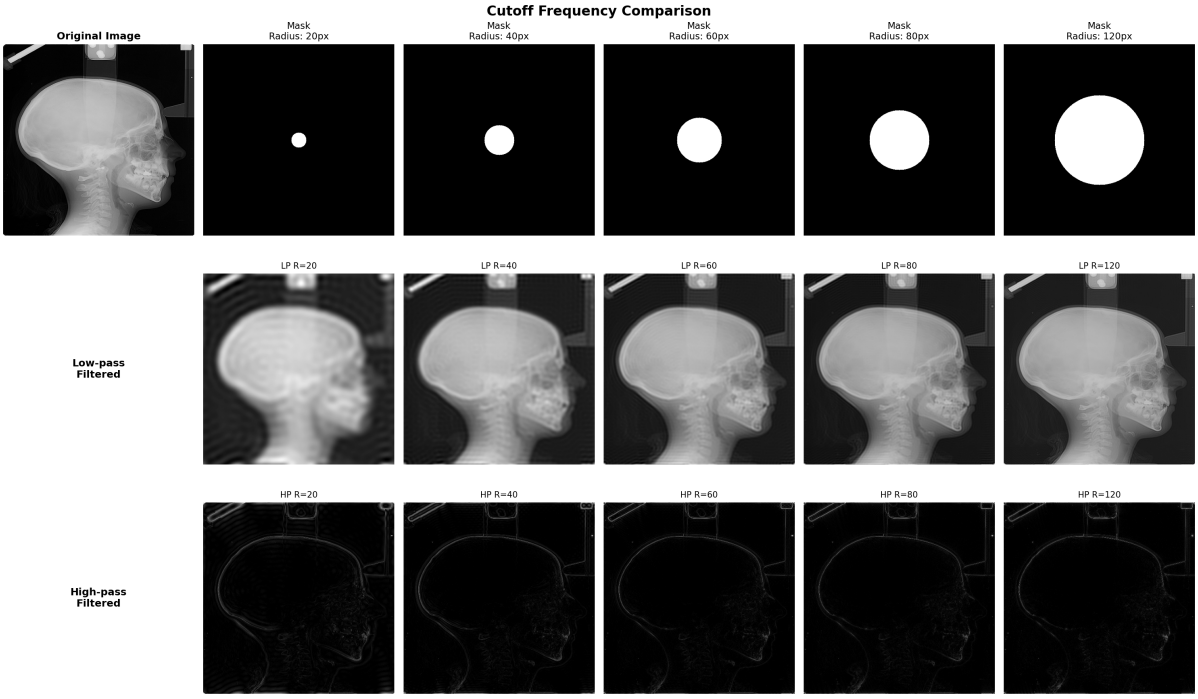


Figure 11: Effect of different cutoff frequencies (30, 60, 100 pixels) on low-pass filtering

6.3.1 Magnitude Spectrum Analysis

The magnitude spectrum reveals:

- Bright center contains low-frequency components (overall structure)
- Surrounding regions contain high-frequency components (details, edges)
- Radial symmetry indicates isotropic frequency distribution
- Diagonal lines indicate directional patterns in the image

6.3.2 Low-Pass Filtering

- **Cutoff = 30:** Severe blurring, only coarse structure preserved. Effectively removes all fine details and textures.
- **Cutoff = 60:** Moderate blurring, balanced smoothing. Good for noise reduction while maintaining recognizability.
- **Cutoff = 100:** Subtle smoothing, most details retained. Useful for mild noise suppression.

Relationship: Smaller cutoff radius $\rightarrow$ more frequencies removed $\rightarrow$ stronger blur effect

6.3.3 High-Pass Filtering

High-pass filtering produces edge-enhanced images:

- Removes constant and slowly varying components (illumination, background)
- Emphasizes rapid intensity changes (edges, fine textures)
- Results in darker overall image with highlighted boundaries
- Useful for edge detection and feature extraction

6.3.4 Advantages of Frequency Domain

- Intuitive understanding of filter effects through frequency response
- Efficient implementation of large kernel convolutions using FFT
- Precise control over frequency band selection
- Theoretical foundation for filter design

However, spatial domain methods like bilateral filtering cannot be easily implemented in frequency domain due to their non-linear, content-dependent nature.

7 Conclusion

This laboratory work provided comprehensive hands-on experience with fundamental image processing techniques. The following key insights were gained:

7.1 Summary of Findings

1. **Edge Detection:** Scharr operator provides superior edge detection accuracy, particularly for diagonal edges, while Sobel offers the best balance between performance and noise immunity.
2. **Custom Kernels:** Convolution kernels enable diverse image effects. Kernel design directly influences output characteristics, with center-weighted kernels for sharpening and uniform kernels for smoothing.
3. **Noise Reduction:** Median filtering excels at removing salt-and-pepper noise with PSNR improvements of 8-12 dB. Filter selection must match noise characteristics for optimal performance.
4. **Motion Blur:** Directional blur kernels accurately simulate camera motion. Kernel size and angle provide precise control over blur characteristics.
5. **Frequency Domain:** Fourier analysis enables intuitive frequency-based filtering. Low-pass filters smooth images while high-pass filters emphasize edges, with cutoff frequency determining filter strength.