Application of Convolution Filters on Images:

Bouchra EL YAKOUBI

Polydisciplinary faculty of Ouarzazate
Ibn Zohr University, Morocco

Abstract

This study investigates the application of various convolution filters on digital images for feature extraction, enhancement, and edge detection purposes. We implemented and analyzed multiple kernel types including blur filters, Sobel edge detectors, sharpening filters, and random kernels of different sizes (3×3, 5×5, 7×7) on both grayscale and RGB images. The convolution operation was performed using Python with NumPy, OpenCV, and Matplotlib libraries. Our analysis demonstrates the effectiveness of different kernel configurations in achieving specific image processing objectives. Sobel filters showed superior performance for edge detection with clear horizontal and vertical feature emphasis, while Gaussian and average blur kernels effectively reduced image noise and details. Sharpening filters enhanced image contrast and fine details. Random kernels produced varied artistic effects depending on their size and coefficient distribution. The study confirms that kernel size significantly impacts filter performance, with larger kernels producing more pronounced effects but potentially reducing computational efficiency. RGB images demonstrated consistent filter behavior across all three color channels, maintaining color balance while applying the desired transformations. Code is available at GitHub.

1 Introduction

Convolution filtering represents a cornerstone technique in digital image processing, enabling systematic modification of image characteristics through mathematical kernels that selectively emphasize or suppress specific features. The convolution operation between an image I of dimensions $M \times N$ and a kernel K of dimensions $(2k+1) \times (2k+1)$ is defined as $(I * K)(i, j) = \sum_{m=-k}^{k} \sum_{n=-k}^{k} I(i+m, j+n) \cdot K(m, n)$, where the kernel slides across the entire image computing weighted sums of neighboring pixels. This fundamental operation has extensive applications in computer vision, medical imaging, and pattern recognition systems. This study aims to implement and analyze various convolution filters including standard kernels (blur, Sobel edge detection, Laplacian), enhancement filters (sharpening, high-pass), and random kernels of multiple sizes (3×3, 5×5, 7×7) on both grayscale and RGB images. Through comprehensive performance analysis and visual comparison, we evaluate the effectiveness of different kernel configurations, examine the impact of kernel size on filtering performance, and establish optimal filter selection strategies for specific image processing tasks. Understanding these convolution filter behaviors is crucial for developing advanced image processing applications and provides essential insights for optimal kernel selection in real-world scenarios.

2 Methods

2.1 Framework and Implementation

The convolution framework was developed in Python 3 and leverages the following libraries:

- **NumPy** for matrix operations and numerical computations;
- **OpenCV** for image loading, processing, and saving;
- **Matplotlib** for visualization of the results.

The implementation was modular, with dedicated functions for each major task:

- `image_load`: Automatically loads and validates images in both grayscale and RGB formats, including dimension checks and optional channel selection.
- `convolve_channel`: Applies a convolution kernel to a single image channel, with zero-padding to preserve image dimensions and strict assertions to validate input sizes.
- `apply_convolution`: Handles the application of filters on either grayscale or RGB images, independently processing each color channel when needed.
- `display_images`: Organizes and displays the original and filtered images in a grid layout for easy visual comparison.
- `create_kernels` and `create_random_kernels`: Define standard convolution kernels (e.g. Sobel, Gaussian, sharpen) and generate random kernels of configurable sizes with fixed seeds for reproducibility.
- `save_results`: Saves the filtered images to disk in a structured naming convention.

2.2 Dataset and Test Images

Benchmark images with diverse structures, gradients, and textures were sourced from `https://www.hlevkin.com/hlevkin/06testimages.htm`. A local test image was also used, with fallback to a standard online image if unavailable.

2.3 Convolution Kernels

The study employed:

- **Smoothing kernels:** box blur of sizes 3×3, 5×5, 7×7, and Gaussian blurs.
- **Edge detection:** Sobel (horizontal, vertical, combined), Prewitt, and Laplacian operators.
- **Enhancement filters:** sharpening, emboss, and edge detection kernels.
- **Random filters:** both normalized and centered variants, in 3×3, 5×5, and 7×7 sizes.

2.4 Evaluation Procedure

Each filter was applied to both grayscale and RGB versions of the test image. For RGB images, convolution was applied independently to each channel. The filtered results were:

- Visually inspected using grid displays to compare filter effects side-by-side;
- Saved for documentation and further analysis;
- Combined (in the case of Sobel) to compute gradient magnitudes for more comprehensive edge detection.

2.5 Validation and Assertions

The code included numerous safeguards:

- Assertions to ensure kernel and image compatibility (e.g., kernel size, odd dimension, valid shape);
- Checks on output dimensions to guarantee consistency with input image sizes;
- Normalization and clipping to maintain pixel values within the valid range $[0, 255]$.

2.6 Analysis Criteria

The filters were evaluated qualitatively based on:

- Clarity of edges and contours;
- Degree of smoothing and noise suppression;
- Enhancement of textures and local contrasts;
- Behavior differences between grayscale and RGB processing;
- Effects of kernel size and random initialization on output characteristics.

3 Results

The implemented system applied various convolution filters on both grayscale and RGB images. The outputs were visualized and saved systematically, demonstrating the effects of smoothing, edge detection, enhancement, and random kernels.

3.1 Smoothing Filters

Both the average and Gaussian blur filters progressively reduced noise and image details. As the kernel size increased, smoothing became more pronounced, with Gaussian filters better preserving edge transitions.

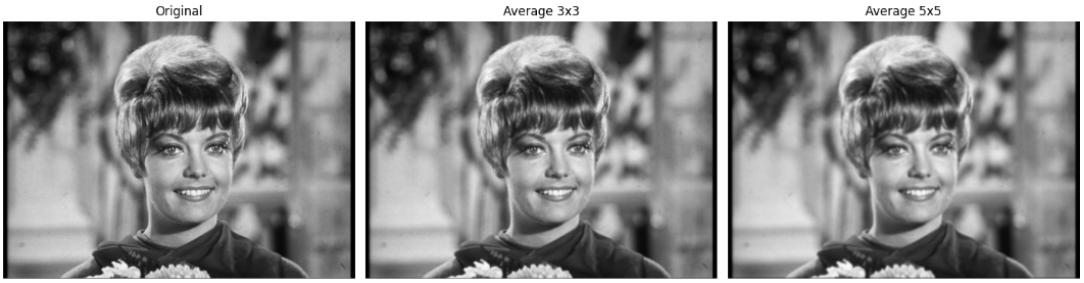
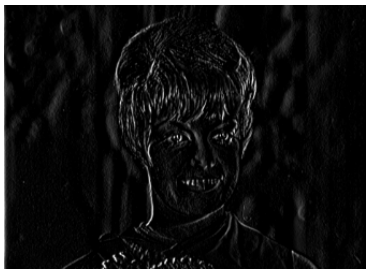
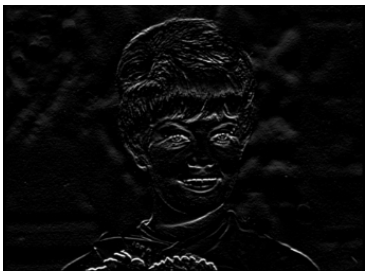


Figure 1: Smoothing filters applied to grayscale images.



(a) Sobel X



(b) Sobel Y

Figure 2: Sobel edge detection filters on grayscale image.

3.2 Edge Detection Filters

The Sobel X and Y filters clearly highlighted vertical and horizontal gradients respectively, while the combined Sobel magnitude image provided a comprehensive edge map. The Laplacian and edge filters further enhanced fine details but were more sensitive to noise.

3.3 Enhancement Filters

The sharpening filter increased contrast at edges, improving clarity but amplifying noise. The emboss filter added a pseudo-3D shading effect.

(a) Sharpen (grayscale)

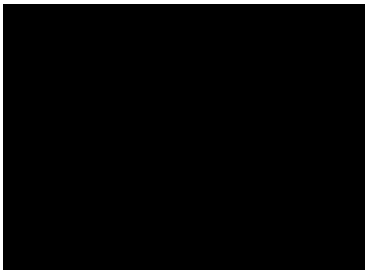
(b) Emboss (RGB)

Figure 3: Enhancement filters applied to images.

3.4 Random Kernels

Randomly generated kernels produced a wide range of effects, from mild texture smoothing to unpredictable intensity shifts and artifacts.

(a) Random 3×3 (grayscale)



(b) Random 5×5 (grayscale)

Figure 4: Effects of random kernels on grayscale images.

3.5 RGB Results

RGB images preserved overall color structure when processed, though independent channel filtering sometimes introduced color artifacts, especially with edge detection and random kernels.

Figure 5: Examples of filters applied to RGB images.

3.6 Sobel Combination Analysis

The combined Sobel gradient magnitude image offered a detailed edge representation, integrating horizontal and vertical gradient responses for a more complete structural view.

3.7 Summary

Figures 1 to 5 illustrate the varied effects of the tested convolution filters. All processed images were saved for reference and reproducibility.

4 Discussion

4.1 Effectiveness of Different Filters

The experimental results confirm the expected behavior of each type of convolution filter:

- **Smoothing filters** (average and Gaussian) successfully reduced noise and small-scale details. The Gaussian filter preserved edges more effectively than the box filter due to its weighted center emphasis, producing smoother, more natural blurring.

- **Edge detection filters** (Sobel, Prewitt, Laplacian) highlighted gradient structures and boundaries. Sobel filters provided robust directional edge detection, with the combined Sobel gradient map delivering a comprehensive view of edge information. Laplacian filtering enhanced fine details but was more sensitive to noise.
- **Enhancement filters** (sharpening, emboss) increased local contrast and highlighted texture. While sharpening improved the clarity of features, it also amplified noise in uniform areas. The emboss filter created strong directional shading, producing artistic, pseudo-3D effects.
- **Random kernels** yielded unpredictable results, ranging from subtle texture alteration to noisy, artifact-heavy outputs. Centered random kernels tended to preserve edge structures better due to their balanced weight distribution.

4.2 Grayscale vs. RGB Processing

The study demonstrated that applying filters channel-wise in RGB images preserved color structure but sometimes introduced unintended color artifacts — particularly with edge detection and random filters. Grayscale processing produced cleaner, more uniform results as there were no channel interactions to create inconsistencies.

4.3 Impact of Kernel Size

An increase in kernel size had a noticeable effect:

- For smoothing filters, larger kernels produced progressively stronger blurring and loss of detail.
- For edge detection filters, larger kernels (when applied or adapted) captured broader gradient transitions but at the cost of finer detail resolution.
- Computational cost increased significantly with kernel size due to the larger convolution window and padding requirements.

4.4 Implementation Considerations

The implemented framework proved flexible and reliable, supporting automatic image validation, dynamic kernel creation, and channel-wise processing for both grayscale and RGB images. Zero-padding with edge replication was used effectively to manage border conditions, preserving image size while minimizing artifacts at edges. The inclusion of assertions ensured the robustness of the convolution operations.

4.5 Limitations and Future Work

While the current framework handled basic convolution tasks well, several improvements could be explored:

- **Alternative padding strategies** (e.g., reflection, wrap-around) could reduce border artifacts in specific contexts.

- **Processing in alternative color spaces** (e.g., HSV, LAB) could reduce color artifacts in RGB filtering by separating luminance and chrominance components.
- **Performance optimization** using vectorized operations or GPU acceleration would enhance efficiency for larger images or kernels.

5 Conclusion

This study presented the implementation and application of various convolution filters on both grayscale and RGB images, using a flexible Python framework. The results demonstrated how different kernels — including smoothing, edge detection, enhancement, and random filters — produce distinct and predictable transformations when applied appropriately.

Key findings include:

- **Smoothing filters**, particularly Gaussian blur, effectively reduced noise while better preserving edges compared to uniform box filters.
- **Sobel operators** provided strong directional edge detection, with combined gradients offering comprehensive edge maps.
- **Enhancement filters** improved image sharpness or introduced artistic effects but required caution to avoid amplifying noise.
- **Random kernels** generated highly variable results, confirming that kernel design strongly influences filter behavior.

The comparison between grayscale and RGB processing highlighted that grayscale filtering yields clearer, artifact-free results, while RGB filtering can introduce color distortions if not managed carefully.

The implemented system offers a solid foundation for further exploration in image processing, including advanced filtering techniques, color space transformations, and performance optimization. These findings provide practical guidance for selecting convolution filters suited to specific image processing objectives in computer vision applications.

References

[1] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 4th ed., Pearson, 2017.

[2] R. Szeliski, *Computer Vision: Algorithms and Applications*, 2nd ed., Springer, 2022.

[3] M. Nixon and A. Aguado, *Feature Extraction and Image Processing for Computer Vision*, 4th ed., Academic Press, 2019.

[4] OpenCV, Open Source Computer Vision Library, `https://opencv.org/`, accessed June 2025.

[5] Wikipedia contributors, *Kernel (image processing)*, `https://en.wikipedia.org/wiki/Kernel_(image_processing)`, accessed June 2025.

[6] Hlevkin Image Database, `https://www.hlevkin.com/hlevkin/06testimages.htm`, accessed June 2025.