Journal of NAMP 68, 1 (2024) 7-22

The Nigerian Association of Mathematical Physics

Journal homepage: https://nampjournals.org.ng

APPLICATION OF MATHEMATICAL CONVOLUTION APPROACH OF IMAGE SHARPENING TO DIGITAL AND SATELLITE IMAGING

***Charles Ekene Chika and [1]Chukwuebuka Chukwudi Chidiebere**

[1,*]*Department of Mathematics, Faculty of Physical Sciences, University of Nigeria, Nsukka*

ARTICLE INFO

Article history:
Received xxxxx
Revised xxxxx
Accepted xxxxx
Available online xxxxx

Keywords:
Mathematical convolution,
Image sharpening,
Laplacian,
Digital imaging,
Kernel.

ABSTRACT

This research paper explores the roles of convolution and Laplacian operators in image sharpening, with the primary objective of enhancing image clarity. The study delves into the mathematical foundations of these operators and their application in digital image processing. Methods are compared to evaluate their effectiveness in terms of computational efficiency and visual outcomes. The paper identifies the gaps in existing literature, particularly the lack of comprehensive literature on the derivation of the Laplacian kernel and reasons for post-processing methods after its convolution with an image. The literature in this paper helps understand the foundation of Laplacian kernel, offering valuable insights on its derivation, and setting the basis for other techniques that have been built on it, with applications in satellite imagery and digital photography. Also, comparison of Laplacian kernel with unsharp masking method is presented.

1. Introduction

Image sharpening is a crucial technique in digital image processing, aimed at enhancing the clarity and definition of images by accentuating edges and fine details. In recent years, the application of mathematical convolution approaches to image sharpening has gained significant attention, particularly in digital photography and satellite imaging [1], [2].

Numerous studies have investigated various methods for image sharpening. For instance, Gonzalez and Woods [3] provided a comprehensive overview of traditional sharpening techniques, including unsharp masking and high-pass filtering. More recently, adaptive sharpening method based on edge-preserving filters, demonstrating improved performance in preserving textures while enhancing edges have been proposed [4] [5]. In the context of satellite imaging, multi-scale sharpening approaches that effectively enhances both local and global contrast in remote sensing images have been developed [6].

*Corresponding author: Charles Ekene Chika
E-mail address: charles.chika@unn.edu.ng
https://doi.org/10.60787/10.60787/jnamp.v68no1.412

Despite these advancements, there remains a gap in the literature concerning the comprehensive exploration of mathematical convolution approaches, particularly the derivation and application of Laplacian operators for image sharpening. Existing studies often overlook the combined application of convolution and Laplacian operators, leaving room for exploring potential synergies between these techniques.

The present study aims to address this research gap by conducting a detailed comparative analysis of convolution and Laplacian operators in image sharpening. We investigated their individual and combined effects on image clarity, evaluating computational efficiency and visual outcomes. This research is motivated by the growing demand for high-quality image processing techniques in various fields, including digital photography, medical imaging, and remote sensing.

The significance of this study lies in its potential to enhance the quality of digital and satellite images, which has far-reaching implications. In digital photography, improved sharpening techniques can lead to more visually appealing and detailed images, benefiting both amateur and professional photographers. In satellite imaging, enhanced sharpening methods can improve the accuracy of land use classification, urban planning, and environmental monitoring [7].

By providing a deeper understanding of these methods and their applications, this research aims to contribute to the advancement of image processing techniques and their practical implementation in real-world scenarios. The findings of this study may pave the way for more efficient and effective image sharpening algorithms, ultimately improving the quality and utility of digital and satellite imagery across various domains.

2 Methodology: Mathematical Foundation

2.1 Convolution in Mathematics

Convolution is a fundamental mathematical operation applied in various fields of mathematics, engineering, and physics. It is an operation that combines two functions to produce a third function. It is sometimes denoted by the symbol "$*$".

Given two functions f and g, their convolution, denoted as $f * g$, is defined as the integral of the product of the two functions after one is reversed and shifted. In the continuous domain, it is expressed as:

$$(f * g)(x) = \int_{-\infty}^{\infty} f(t) \cdot g(x - t)\, dt$$

2.1.1 Properties of Convolution

Some applicable properties of convolution in image processing are:

- **Commutativity**: $f * g = g * f$
- **Associativity**: $(f * g) * h = f * (g * h)$

1. **Commutativity**: This means that the order of the convolution operations does not affect the final result. In image processing, commutativity allows for operations such as filtering and smoothing to be applied interchangeably without altering the outcome. For example, applying a blur filter on an image, followed by an edge detection filter produces the same result as applying the edge detection filter first, and then the blur filter. This property enables greater flexibility in

designing image processing pipelines and allows for experimentation with different sequences of operations to achieve desired effects.

2. **Associativity**: In image processing, the associativity property of convolution allows multiple convolution operations to be combined and performed in any order without affecting the final result. This is useful when applying multiple filters or kernels to an image successively. By exploiting associativity, complex image processing pipelines can be constructed by combining individual convolution operations, leading to more efficient and flexible image processing algorithms.

2.2 Discrete Convolution

In discrete domains, such as digital signal processing and computer vision, the convolution operation is defined similarly to its continuous counterpart. However, it uses summation in place of integration. Discrete convolution is particularly useful for processing digital signals represented as sequences of discrete values.

Let's consider two discrete functions $f(k)$ and $g(k)$. Their convolution, denoted as $f * g$, is defined as the sum of the element-wise multiplication of f and g with one function shifted relative to the other:

$$(f * g)(n) = \sum_{k=-\infty}^{\infty} f(k) \cdot g(n-k)$$

where:

- n is the current position or index at which the convolution result is computed.
- k is the discrete index or position within the summation.
- $f(k)$ and $g(k)$ are the values of the functions f and g at index k, respectively.

The summation is taken over all possible values of k, ranging from negative infinity to positive infinity. However, in practice, it is truncated to a finite range based on the available data. Discrete convolution forms the basis for the development of various algorithms and techniques for analyzing and processing discrete signals and images. [8], [9]

2.3 Convolution Operators in Image Sharpening

2.3.1 Definition and Basic Principles

The concept of convolution involves a mathematical operation where two functions are combined to produce a third function, representing the features of one, modified by the other. In image processing, these functions are an image and a kernel (filter or mask).

The convolution process can be visualized as sliding the kernel across the image, computing a weighted sum of the pixel values under the kernel at each position. The result of this operation is a new image that highlights specific features, such as edges or textures, depending on the kernel used. Mathematically, the convolution of an image I with a kernel K is given by:

$$(I * K)(x, y) = \sum_{i=-m}^{m} \sum_{j=-n}^{n} I(x-i, y-j) \cdot K(i, j)$$

where (x, y) are the coordinates of the image.

In image sharpening, convolution operators are used to enhance the high-frequency components of an image, which correspond to areas with rapid intensity changes, such as edges [3]. By applying a sharpening kernel, the convolution process highlights these high-frequency features, which results in a clearer and more defined image [10] [11].

2.3.2 Two-Dimensional Convolute Integer Operators

Two-dimensional convolute integer operators are a specific class of convolution operators that utilize integer values in their kernels. These operators are useful in digital image processing due to their computational efficiency and simplicity [12].

Two key characteristics of two-dimensional convolute integer operators are [12]:

1. **Integer coefficients**: The kernel values are restricted to integers, simplifying computational complexity and preventing quantization errors during convolution.

2. **Frequency sensitivity**: The operators are designed to selectively emphasize or suppress specific frequency ranges, allowing for targeted enhancement or attenuation of specific image features. For example, a kernel designed to emphasize high frequencies will enhance the edges and fine details of an image, making it appear sharper. Conversely, a kernel that targets low frequencies can be used for smoothing or blurring effects.

2.4 Laplacian Operator

The Laplace operator was first applied in celestial mechanics by Pierre-Simon de Laplace, to study planetary motion in space [13]. He introduced the Laplace operator in the context of his work on harmonic functions and other concepts related to it [14].

The Laplace operator has since then, been used to explain a variety of things. For example, it helps describe electric potentials, how heat and fluids move, and even some parts of quantum mechanics. It has also been recasted to the discrete space, where it has been used in applications related to image processing.

The Laplacian operator is a second-order differential operator and it is defined as the divergence of the gradient of the scalar field. In Cartesian coordinates, the Laplacian operator is expressed as the sum of second partial derivatives:

Divergence is denoted by ∇ and expressed as:

$$\nabla = \langle \frac{\partial}{\partial x}, \frac{\partial}{\partial y} \rangle$$

The gradient of a two-dimensional function, f, is given by:

$$\nabla f = \langle \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \rangle$$

Then, the Laplacian (that is, the divergence of the gradient) of f can be defined by the sum of unmixed second partial derivatives:

$$\nabla \cdot \nabla f = \nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

2.4.1 The Discrete Laplacian

The discrete Laplacian is a computational analog of the Laplacian operator in continuous domains, designed for discrete spaces. In this study, we use it for image sharpening in image processing.

Definition: In discrete spaces, the Laplacian operator is typically defined as a numerical approximation of the continuous Laplacian. One common formulation of the discrete Laplacian is based on finite differences using the central difference approximation.

2.4.2 Discretization of the Laplacian Operator using Central Difference Approximation

In a 2D grid, the discrete Laplacian $\nabla^2 f(x,y)$ at a point (x,y) can be approximated using differences along the horizontal and vertical directions to:

$$\nabla^2 fx,y) \approx f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

To discretize the Laplacian operator of a function $f(x,y)$ using a second-order central difference approximation, we'll approximate the second partial derivatives with respect to both x and y. Let's denote the grid spacing or step size in the x direction as Δx and in the y direction as Δy.

The Laplacian operator in two dimensions is given by:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Using the second-order central difference approximation, we can discretize the second partial derivatives as follows:

$$\frac{\partial^2 f}{\partial x^2} \approx \frac{f(x+\Delta x,y) - 2f(x,y) + f(x-\Delta x,y)}{(\Delta x)^2}$$

$$\frac{\partial^2 f}{\partial y^2} \approx \frac{f(x,y+\Delta y) - 2f(x,y) + f(x,y-\Delta y)}{(\Delta y)^2}$$

For the Laplace Operator, we take 1 as the step size for both x and y, which signifies the smallest change or the spacing between adjacent grid points in both the x and y directions. We have:

$$\frac{\partial^2 f}{\partial x^2} \approx \frac{f(x+1,y) - 2f(x,y) + f(x-1,y)}{(1)^2}$$

$$\frac{\partial^2 f}{\partial y^2} \approx \frac{f(x,y+1) - 2f(x,y) + f(x,y-1)}{(1)^2}$$

Now, we can sum these two approximations to obtain the discrete Laplacian operator

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

This expression represents the discretized Laplacian operator, where $f(x,y)$ is evaluated at neighbouring grid points in both the x and y directions.

2.5 The Laplacian Filter

The Laplacian filter is a commonly used image processing filter for edge detection and image enhancement. It is based on the Laplacian operator, which measures the second derivative of an image.

Recall, in a 2D grid, the discrete Laplacian $\nabla^2 f(x,y)$ at a point (x,y) can be approximated using differences along the horizontal and vertical directions and is given as:

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

The discretized Laplacian operator can be represented as a convolution kernel, which is a small matrix of weights that represent the coefficients at each point. A common approach is to use a 3x3 grid.

$$\begin{bmatrix} f(x-1,y+1) & f(x-1,y) & f(x-1,y-1) \\ f(x,y+1) & f(x,y) & f(x,y-1) \\ f(x+1,y+1) & f(x+1,y) & f(x+1,y-1) \end{bmatrix}$$

Taking the coefficients of each expression, the Laplacian filter kernel is expressed as:

$$L_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad (1)$$

This kernel represents the weights to be applied to the neighbouring pixels of the image during convolution. The central pixel is assigned a negative weight (-4), while the surrounding pixels are assigned positive weights (1). This reflects the change in intensity from the center to the surrounding pixels.

Larger Kernel for Better Edge Detection

A larger Laplacian kernel is used to get a better result in highlighting the edges in an image. This is achieved by considering more neighbouring pixels. In Equation 1, the diagonal neighbouring pixels are ignored, but when equal weights are assigned to every neighbouring pixel, we obtain a better Laplacian Kernel.

However, it is important to note that the weight of the center pixel should be the negative of the sum of the neighbouring pixels' equal weight, to maintain the balance of the Laplacian operator. The sum of the integers in the kernel should be zero so that it responds strongly to edges but not to uniform areas.

An extension of Equation 1 that considers every neighbouring pixel would be:

$$L_2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (2)$$

The larger Laplacian kernel improves edge detection because it considers a larger neighbourhood around each pixel. This reduces the sensitivity to small intensity fluctuations (noise) and focuses on larger intensity changes (edges).

Other variations of the Laplacian operator as pointed out in [3] are:

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1

-1	8	-1
-1	-1	-1

These are Negative variants of the matrixes in 1 and 2 respectively, with a positive weight at the center.

3 Application of Laplacian Matrix for Image Sharpening

In this section, we explore the practical application of the Laplacian matrix, derived in section 2, for image sharpening. We discuss how the Laplacian matrix, as a discrete representation of the Laplacian operator, can be effectively employed in image sharpening. The convolution of the Laplacian matrix for image sharpening is examined, along with experimental results demonstrating its effectiveness.

3.1 Laplacian Filter and Convolution

The Laplacian filter is also referred to as the Laplacian Matrix or Laplacian Kernel. In this section, we explore the convolution operation between the Laplacian filter and the input image, which serves as the foundation for Laplacian-based image sharpening techniques.

3.1.1 Convolution with Laplacian Kernel

When convoluting the Laplacian filter with an input image, each pixel undergoes a transformation based on its surrounding neighbourhood. The process is as follows:

1. **Kernel Centering:** The Laplacian kernel is centered on each pixel in the input image.

2. **Pixel Neighborhood:** The Kernel's weights are applied to the neighbourhood of the current pixel in consideration. This neighbourhood comprises the center pixel and the pixels directly adjacent to it, horizontally, vertically and diagonally, in both directions.

3. **Weighted Sum Calculation:** The weights of the Laplacian kernel on each pixel in the neighbourhood determine the contribution of each pixel to the final filtered value. The kernel is convolved with the pixel neighbourhood by performing element-wise multiplication between the kernel and the corresponding pixel values in the neighbourhood, then summing up the results.

4. **Resulting Pixel Value:** The sum of the element-wise products yields the new pixel value for the center pixel (pixel under consideration) in the filtered image. This value represents the intensity change or gradient magnitude at that pixel location. A high positive or negative value indicates a strong intensity change, such as an edge, while values close to zero suggest smoother regions.

5. **Iterative Convolution:** The convolution process is repeated in a loop for each pixel in the input image, covering the entire image.

By convolving the Laplacian kernel with the input image, we obtain a Laplacian-filtered image that highlights edges.

3.1.2 Mathematical Formulation

The convolution operation between the input image $f(x, y)$ and the Laplacian kernel h can be expressed as: 1

$$g(x,y) = \sum_{s=-k}^{k} \sum_{t=-k}^{k} h(s,t) \cdot f(x-s, y-t)$$

where:

- $g(x,y)$ is the filtered image,
- $h(s,t)$ represents the Laplacian kernel,
- $f(x-s, y-t)$ denotes the pixel values in the neighborhood of pixel (x,y),
- k is the step size.

Example: To provide a practical example, let's consider a simple 3x3 grayscale image and apply the Laplacian filter to it. We'll use the Laplacian kernel derived in section 3:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Let's suppose we have the following 3x3 grayscale image:

$$\begin{bmatrix} 10 & 20 & 30 \\ 40 & 50 & 60 \\ 70 & 80 & 90 \end{bmatrix}$$

Convolution Calculation:

Step 1: Zero Padding the Image

Add a border of zeros around the original image:

$$\text{OriginalImage} = \begin{bmatrix} 10 & 20 & 30 \\ 40 & 50 & 60 \\ 70 & 80 & 90 \end{bmatrix}$$

$$\text{PaddedImage} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 20 & 30 & 0 \\ 0 & 40 & 50 & 60 & 0 \\ 0 & 70 & 80 & 90 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Step 2: Apply the Kernel

For each element in the original image (3x3 part), we will center the kernel on that element in the padded image, multiply corresponding elements, and sum the results.

Element-wise Calculation g(1,1):

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 10 & 20 \\ 0 & 40 & 50 \end{bmatrix}$$

$$\text{Sum} = (0 \cdot 0) + (1 \cdot 0) + (0 \cdot 0) + (1 \cdot 0) + (-4 \cdot 10) + (1 \cdot 20) + (0 \cdot 0) + (1 \cdot 40) + (0 \cdot 50)$$

$$= 0 + 0 + 0 + 0 - 40 + 20 + 0 + 40 + 0$$

$$= 20$$

g(1,2):

$$\begin{bmatrix} 0 & 0 & 0 \\ 10 & 20 & 30 \\ 40 & 50 & 60 \end{bmatrix}$$

$$\text{Sum} = (0 \cdot 0) + (1 \cdot 0) + (0 \cdot 0) + (1 \cdot 10) + (-4 \cdot 20) + (1 \cdot 30) + (0 \cdot 40) + (1 \cdot 50) + (0 \cdot 60)$$

$$= 0 + 0 + 0 + 10 - 80 + 30 + 0 + 50 + 0$$

$$= 10$$

g(1,3):

$$\begin{bmatrix} 0 & 0 & 0 \\ 20 & 30 & 0 \\ 50 & 60 & 0 \end{bmatrix}$$

$$\text{Sum} = (0 \cdot 0) + (1 \cdot 0) + (0 \cdot 0) + (1 \cdot 20) + (-4 \cdot 30) + (1 \cdot 0) + (0 \cdot 50) + (1 \cdot 60) + (0 \cdot 0)$$

$$= 0 + 0 + 0 + 20 - 120 + 0 + 0 + 60 + 0$$

$$= -40$$

g(2,1):

$$\begin{bmatrix} 0 & 10 & 20 \\ 0 & 40 & 50 \\ 0 & 70 & 80 \end{bmatrix}$$

$$\text{Sum} = (0 \cdot 0) + (1 \cdot 10) + (0 \cdot 20) + (1 \cdot 0) + (-4 \cdot 40) + (1 \cdot 50) + (0 \cdot 0) + (1 \cdot 70) + (0 \cdot 80)$$

$$= 0 + 10 + 0 + 0 - 160 + 50 + 0 + 70 + 0$$

$$= -30$$

g(2,2):

$$\begin{bmatrix} 10 & 20 & 30 \\ 40 & 50 & 60 \\ 70 & 80 & 90 \end{bmatrix}$$

$$\text{Sum} = (0 \cdot 10) + (1 \cdot 20) + (0 \cdot 30) + (1 \cdot 40) + (-4 \cdot 50)$$
$$+(1 \cdot 60) + (0 \cdot 70) + (1 \cdot 80) + (0 \cdot 90)$$
$$= 0 + 20 + 0 + 40 - 200 + 60 + 0 + 80 + 0$$
$$= 0$$

g(2,3):

$$\begin{bmatrix} 20 & 30 & 0 \\ 50 & 60 & 0 \\ 80 & 90 & 0 \end{bmatrix}$$

$$\text{Sum} = (0 \cdot 20) + (1 \cdot 30) + (0 \cdot 0) + (1 \cdot 50) + (-4 \cdot 60) + (1 \cdot 0)$$
$$+(0 \cdot 80) + (1 \cdot 90) + (0 \cdot 0)$$
$$= 0 + 30 + 0 + 50 - 240 + 0 + 0 + 90 + 0$$
$$= -70$$

g(3,1):

$$\begin{bmatrix} 0 & 40 & 50 \\ 0 & 70 & 80 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\text{Sum} = (0 \cdot 0) + (1 \cdot 40) + (0 \cdot 50) + (1 \cdot 0) + (-4 \cdot 70) + (1 \cdot 80)$$
$$+(0 \cdot 0) + (1 \cdot 0) + (0 \cdot 0)$$
$$= 0 + 40 + 0 + 0 - 280 + 80 + 0 + 0 + 0$$
$$= -160$$

g(3,2):

$$\begin{bmatrix} 40 & 50 & 60 \\ 70 & 80 & 90 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\text{Sum} = (0 \cdot 40) + (1 \cdot 50) + (0 \cdot 60) + (1 \cdot 70) + (-4 \cdot 80) + (1 \cdot 90$$

$$) + (0 \cdot 0) + (1 \cdot 0) + (0 \cdot 0)$$

$$= 0 + 50 + 0 + 70 - 320 + 90 + 0 + 0 + 0$$

$$= -110$$

g(3,3):

$$\begin{bmatrix} 50 & 60 & 0 \\ 80 & 90 & 0 \\ 0 & 0 & 0 \\ \end{bmatrix}$$

$$\text{Sum} = (0 \cdot 50) + (1 \cdot 60) + (0 \cdot 0) + (1 \cdot 80) + (-4 \cdot 90) + (1 \cdot 0$$

$$) + (0 \cdot 0) + (1 \cdot 0) + (0 \cdot 0)$$

$$= 0 + 60 + 0 + 80 - 360 + 0 + 0 + 0 + 0$$

$$= -220$$

Final Result

The resulting matrix after applying the discrete convolution with zero padding is:

$$\text{Result} = \begin{bmatrix} 20 & 10 & -40 \\ -30 & 0 & -70 \\ -160 & -110 & -220 \end{bmatrix}$$

3.2 Implementation for Image Sharpening:

After the convolution of the Laplacian kernel and the input image is completed, the output is a Laplacian-filtered image that highlights edges. This Laplacian-filtered image is multiplied to a negative constant or positive constant depending on the Laplacian kernel used.

If a Laplacian kernel with a negative center weight was used during convolution, the Laplacian-filtered image is multiplied with a negative constant, if it had a positive center weight, then a positive constant [3].

Then the result of the multiplication is added back to the input image to produced a sharpened output image.

This process can be mathematically expressed:

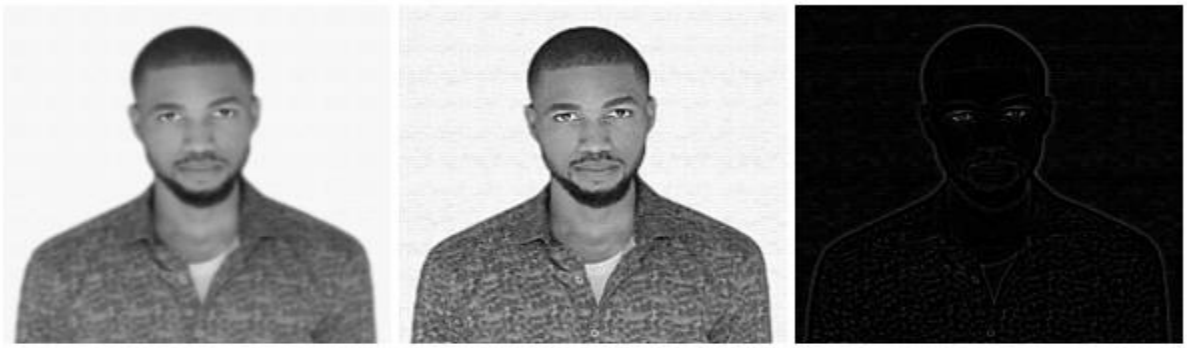
$$g(x, y) = f(x, y) + c * [\nabla^2 f(x, y)]$$

where $g(x, y)$ is the sharpened image, $f(x, y)$ is the input image, c is the constant that represents the scaling factor for the intensity to be applied to the Laplacian-filtered image. $\nabla^2 f(x, y)$ is the laplacian-filtered image.

4 Results:

(a) Grayscale Input Image (b) Sharpened Output Image (c) Laplacian-filtered Image

Figure 1: Using the Laplacian kernel in Eqn. 1 on Digital Image



(a) Grayscale Input Image (b) Sharpened Output Image (c) Laplacian-filtered Image

Figure 2: Using the Laplacian kernel in Eqn. 2 on Digital Image

(a) Grayscale Input Image (b) Sharpened Output Image (c) Laplacian-filtered Image

Figure 3: Using the Laplacian kernel in Eqn. 1 on Space Image

(a) Grayscale Input Image (b) Sharpened Output Image (c) Laplacian-filtered Image

Figure 4: Using the Laplacian kernel in Eqn. 2 on Space Image

Figure 1 and 2 are the input image, sharp image and their Laplacian kernels for digital photograph. Similarly, figure 3 and 4 are for satellite image. It can be seen from figure 1 - 4 that large kernel produces a sharper image. However, care has to be taken as too larger kernel can result to too sharp image and may result to increase in noise; this is based on our observation.

5 Discussion

5.1 Why Do we multiply by a negative constant if the Laplacian kernel has a negative center weight, before adding back to the input image?

In image sharpening, the Laplacian-filtered image highlights edges and details from the original image. However, directly adding it to the input image might subtract edges instead of enhancing them due to negative values. By multiplying the Laplacian image by a negative constant, edges are inverted, enhancing them when added back to the original image. Image sharpening is achieved through this process.

5.2 Comparative Analysis of Laplacian Kernel and Unsharp Masking for Image Sharpening

Unsharp Masking for Image Sharpening:

Unsharp masking is a method that enhances the sharpness of an image by subtracting a blurred version of the image from the original image, creating a mask that is used to highlight the edges and fine details. The steps involved in unsharp masking are:

1. **Blurring the Image**: Convolve the image with a Gaussian kernel to create a blurred version of the image.

2. **Creating the Mask**: Subtract the blurred image from the original image to generate the mask containing high-frequency components.

3. **Combining the Images**: Add the mask back to the original image, often with a scaling factor, to enhance sharpness.

Mathematically, the unsharp masking process is represented as [15] [3]:

$$I_{\text{sharpened}} = I + \alpha(I - G_\sigma * I)$$

where I is the original image, G_σ is the Gaussian blur operator with standard deviation σ, $*$ denotes convolution, and α is a scaling factor controlling the sharpening intensity.

While unsharp masking is effective for sharpening images, the Laplacian kernel offers several advantages:

1. **Sensitivity to edges**: The Laplacian kernel is highly sensitive to edges and corners, making it more effective at enhancing fine details and sharp features compared to unsharp masking, which can sometimes produce halos or ringing artifacts around edges [16].

2. **Isotropy**: The Laplacian kernel is isotropic, which means it responds equally to edges in all directions. This property ensures consistent sharpening across the entire image, regardless of the orientation of the edges [3].

3. **Computational efficiency**: Applying the Laplacian kernel involves a simple convolution operation, which can be computationally more efficient than the blurring step required for unsharp masking, especially for large images or real-time applications [10].

4. **Flexibility**: The Laplacian kernel can be modified or combined with other kernels to achieve different sharpening effects or target specific image characteristics, offering greater flexibility in image enhancement [17].

However, it is important to note that the Laplacian kernel can also amplify noise present in the image, especially when the scaling factor is set too high. Therefore, proper noise reduction techniques or adaptive scaling may be necessary to achieve optimal sharpening results while minimizing noise amplification [18] [19].

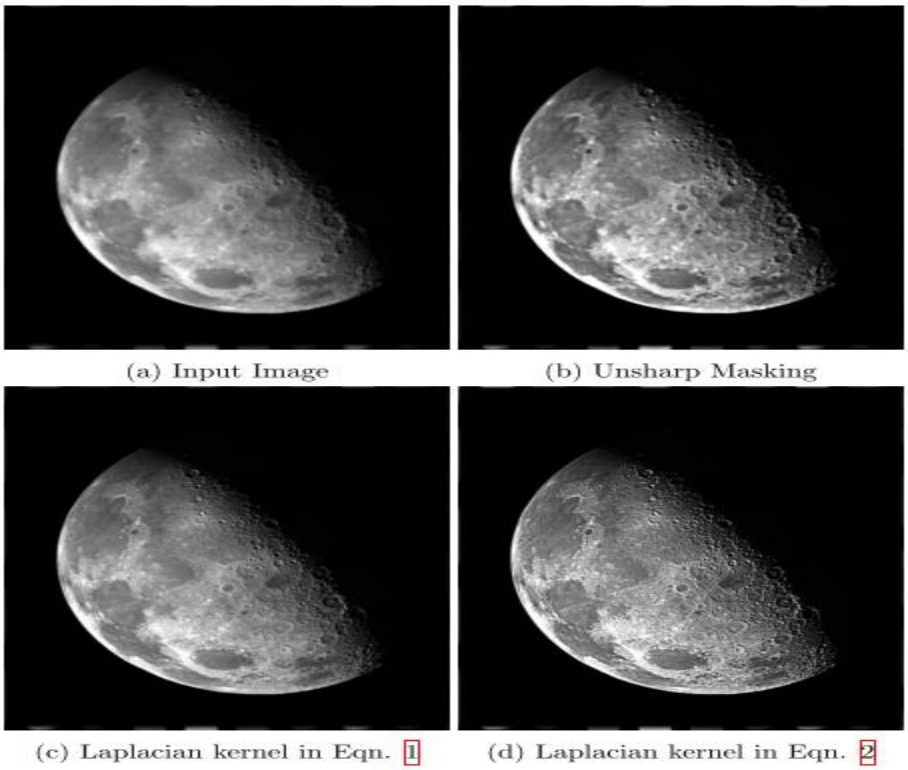


Figure 5: Images from convolution method using Laplacian kernel and Unsharp Masking method

Figure 5 shows that both methods produces sharpened image so one can choose preferred method based on the limitations and advantages stated above of which convolution method with Laplacian kernel will be preferred when computational cost is of interest.

Conclusion

Convolution is a very useful mathematical tool for image sharpening. The direct derivation of its kernels and its algebraic properties like commutativity and associativity makes it simple and flexible to use in application.

Both the Laplacian kernel and unsharp masking are effective methods for image sharpening, each with its strengths and applications. The Laplacian kernel offers a fast, simple, and direct approach to edge enhancement, making it ideal for applications requiring quick and efficient edge detection. With its balanced approach to noise reduction and edge enhancement, Unsharp masking may produce more visually pleasing results, especially for general-purpose image sharpening.

Understanding the advantages and limitations of each method allows practitioners to choose the appropriate technique based on the specific requirements of their application. For real-time edge enhancement tasks, the Laplacian kernel is better. Unsharp masking may be the preferred choice when requiring high-quality image sharpening with reduced noise levels.

References

[1] Morteza Ghahremani and Hassan Ghassemian. “Remote sensing image fusion using ripplet transform and compressed sensing”. In: *IEEE Geoscience and Remote Sensing Letters* 12.3 (2014), pp. 502–506.

[2] Dhara J. Sangani, Rajesh A. Thakker, S.D. Panchal, and Rajesh Gogineni. “Pansharpening of satellite images with convolutional sparse coding and adaptive PCNN-based approach”. In: *Journal of the Indian Society of Remote Sensing* 49.12 (2021), pp. 2989–3004.

[3] Rafael C. Gonzalez and Richard E. Woods. Digital Image Processing. 3rd ed. Pearson Prentice Hall, 2008.

[4] Wenming Tang, Yuanhao Gong, Linyu Su, Wenhui Wu, and Guoping Qiu. “Structure adaptive filtering for edge-preserving image smoothing”. In: *Image and Graphics: 11th International Conference, ICIG 2021, Haikou, China, August 6–8, 2021, Proceedings, Part III 11* (2021), pp. 265–276.

[5] Hui Lv, Pengfei Shan, Hongfang Shi, and Li Zhao. “An adaptive bilateral filtering method based on improved convolution kernel used for infrared image enhancement”. In: *Signal, Image and Video Processing* 16.8 (2022), pp. 2231–2237.

[6] Li-bao Zhang, Yang Sun, and Jue Zhang. “Pan-sharpening based on common saliency feature analysis and multiscale spatial information extraction for multiple remote sensing images”. In: *International Journal of Remote Sensing* 41 (2020), pp. 3095–3118.

[7] Honglyun Park, Namkyung Kim, Sangwoo Park, and Jaewan Choi. “Sharpening of Worldview-3 Satellite Images by Generating Optimal High-Spatial-Resolution Images”. In: *Applied Sciences* (2020). doi: 10.3390/app10207313.

[8] Kholmat Shadimetov and Shermamat Esanov. “The discrete convolution operator D m [β]”. In: AIP Conference Proceedings. Vol. 2365. 1. AIP Publishing LLC. 2021, p. 020033.

[9] Giulio Prevedello and Ken R Duffy. “Discrete convolution statistic for hypothesis testing”. In: Communications in Statistics-Theory and Methods 51.12 (2022), pp. 4097–4118.

[10] Richard Szeliski. Computer vision: algorithms and applications. Springer Nature, 2022. 18

[11] William K Pratt. Digital image processing: PIKS Scientific inside. Vol. 4. Wiley Online Library, 2007.

[12] Thomas R Edwards. “Two-dimensional convolute integer operators for digital image processing”. In: Image Processing Algorithms and Techniques. Vol. 1244. SPIE. 1990, pp. 26–44.

[13] Gengxiang Chen et al. “Laplace neural operator for complex geometries”. In: arXiv preprint arXiv:2302.08166 (2023).

[14] Robert Dautray and Jacques-Louis Lions. “The Laplace Operator”. In: Mathematical Analysis and Numerical Methods for Science and Technology. Springer, 2000, pp. 220– 658.

[15] John C Russ. The image processing handbook. CRC press, 2006.

[16] Raman Maini and Himanshu Aggarwal. “Study and comparison of various image edge detection techniques”. In: International journal of image processing (IJIP) 3.1 (2009), pp. 1–11.

[17] Maria MP Petrou and Costas Petrou. Image processing: the fundamentals. John Wiley & Sons, 2010.

[18] Haotian Xu, Weiwei Xu, and Wenchao Chen. “Seismic erratic noise suppression based on Laplacian-scaled mixture prior”. In: Second International Meeting for Applied Geoscience & Energy. Society of Exploration Geophysicists and American Association of Petroleum . . . 2022, pp. 2967–2971.

[19] Xiucai Ding and Hau-Tieng Wu. “Impact of signal-to-noise ratio and bandwidth on graph Laplacian spectrum from high-dimensional noisy point cloud”. In: IEEE Transactions on Information Theory 69.3 (2022), pp. 1899–1931.

Appendix Suplimentary Material Link to Repository of Programming Implementation of sharpening methods using Python on https://github.com/Ebukachuqz/convolution-with-laplacian-kernels-and-unsharp-masking-in-python**Github**