Graph-Based Representation of Digital Images for Adaptive Edge Sharpening

Kevin Sie - 13525053
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: kevinsie23@gmail.com , 13525053@std.stei.itb.ac.id

***Abstract*— In the rapid development of modern technology, the importance of cameras in smartphones has evolved from supplementary features to a primary selling point. This drives the urgency to continuously improve the quality of images for every new generation of smartphones. One of the more common image processing techniques to achieve this is image sharpening. Image processing commonly treats images as a 2D matrix, where each pixel is processed locally without considering the image as a whole. This paper proposes an alternative approach to image sharpening, where an image is treated as a weighted graph, with each pixel as its vertex, while intensity differences between neighbouring pixels form weighted edges. By modeling an image as a weighted graph, the proposed method performs adaptive edge enhancement based on intensity differences between neighbouring pixels, enabling selective sharpening of significant image structures while reducing unnecessary enhancement in flat regions. This paper contributes a different perspective in sharpening images, with potential advantages in handling images with high noise levels.**

Keywords—sharpening kernel, convolution, weighted graph, matrix, digital image sharpening

I. Introduction

In computer science, a digital image is commonly represented as a finite collection of discrete elements known as pixels. Each pixel has its own spatial coordinates and an intensity value that determines its visual appearance in the image. As digital photography has become ubiquitous across various situations and device hardware, effective image enhancement techniques have become essential to produce high-quality images. One of the most widely used enhancement techniques is image sharpening, a high-pass filtering operation that improves edge clarity and enhances local contrast within an image.

Traditionally, image sharpening methods treat an image as a two-dimensional matrix and rely on convolution kernels to process local pixel neighbourhoods. Convolution kernels, commonly represented as small 3-by-3 matrices with specific values, are designed to emphasize edges and fine details in an image. By incorporating these kernels across all pixels, sharpening algorithms are able to produce images with clearer and more distinct edges, which is highly desirable in image processing.

However, conventional sharpening methods primarily focus on local pixel intensity variations and may unintentionally amplify image noise alongside meaningful image structures. To address this limitation, this paper proposes an alternative graph-based representation of digital images, where pixels are modeled as vertices, while the differences in intensity between neighbouring pixels as weighted edges. Through this new approach, the sharpening process can consider structural relationships and connectivity between pixels rather than relying solely on traditional convolution-based local filtering.

This study applies weighted graph theory to image sharpening in order to explore a more adaptive enhancement approach that considers the relationships between neighbouring pixels and their values. The proposed method evaluates the effectiveness of graph-based adaptive sharpening compared to conventional convolution-based sharpening, particularly in images containing noticeable levels of noise. Through this approach, the paper aims to provide a graph-based perspective for image sharpening and demonstrate the potential application of weighted graph theory in digital image processing.

II. Theoretical Framework

A. Digital Image Representation

$$f(x,y)=\begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

Fig 1. Digital Image Representation for Computer Processing
(Source: Gonzalez and Woods [2018])

A digital image is represented as a two-dimensional matrix consisting of numerical intensity values, where each element x is called a pixel. Each pixel can be represented as an image intensity function $f(x,y)$, where x and y denote the spatial coordinates, while $f(x,y)$ represents the intensity value at that coordinate, with its range being from $0 \leq f(x,y) \leq 255$.

A color digital image is usually formed in RGB color space by using red, green, and blue components images. Same as for the mathematical function $f(x,y)$, these three primary colors

also have the same 8-bit integer value representation ranging from 0 to 255. The combination of different intensities from these three colors creates a wide spectrum of different colors. Meanwhile for grayscale images, though similarly represented by an 8-bit integer value, it simplifies the image by only having one channel of color, ranging from 0 that corresponds to black to 255 that corresponds to white. By reducing each pixel to a single intensity channel, grayscale representations simplifies image processing operations by reducing each pixel to a single intensity value.

Pixels in a digital image are spatially connected to neighbouring pixels based on their coordinates. One common neighbouring model is the 4-neighbourhood structure, where each pixel is connected to adjacent pixels in both the vertical and horizontal directions. These neighbouring relationships are important in image processing operations, since many filtering and enhancement techniques rely on local pixel interactions, including sharpening.

Difference in intensity values between these neighbouring pixels often indicate the presence of edges or structural transitions within an image. Regions with large intensity differences usually correspond to object boundaries or fine image details, making local intensity variation an important component in image sharpening and edge enhancement operations.

B. Convolution and Kernels

Discrete convolution is a mathematical operation that combines two sequences to produce a third sequence, representing how one sequence modifies the other. Convolution of (a_i) *and* (b_i) is

$$(a * b)_n = \sum_{\substack{i,j \\ i+j=n}} a_i \,.\, b_j$$

In image processing, one sequence represents the image signal, while the other represents a filtering kernel. The convolution operation combines information from neighbouring pixels to generate a modified output image.

In digital images, discrete convolution is often used to modify pixel values based on its neighbouring pixels. In convolution-based filtering, a small matrix known as a kernel is moved across the image. Common convolution kernel sizes are 5 x 5, 3 x 3, and 1 x 1. Different kernels can achieve different functions. Among the various kernel sizes available, 3 × 3 kernels are commonly used due to their computational efficiency and ability to capture local image features. During convolution, the kernel is centered on a pixel and multiplied element-wise with the surrounding pixel values. The resulting products are then summed to produce the new intensity value of the center pixel. This value will then become the new value of the pixel. The kernel is systematically applied across the image until all pixels have been processed.

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Fig 2. Sharpening Kernel

(Source: Author)

Basic convolution kernels are foundational in image processing, designed to perform simple yet essential operations on images. One of which is the sharpening kernel, designed to enhance the edges and details in an image, making image edges appear sharper and more visually distinct.

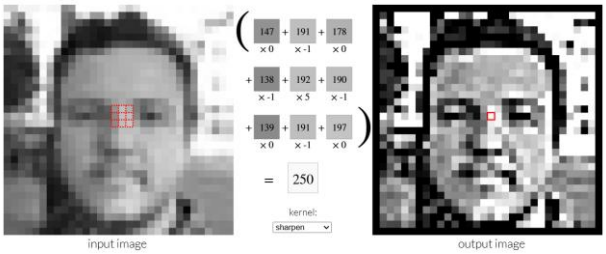

Fig 3. Example Usage Of Sharpening Kernel

(Source: https://setosa.io/ev/image-kernels/)

Sharpening kernels work by amplifying the differences in pixel intensity, weighing heavier on the value of the pixel correlated to it while subtracting intensity values from neighbouring pixels adjacent to it. From a signal-processing perspective, sharpening can be interpreted as a form of high-pass filtering. Since image edges correspond to rapid intensity changes, emphasizing high-frequency components enhances boundaries and fine details within an image.

Although convolution-based sharpening is computationally efficient and widely adopted, its decision-making process is restricted to local pixel neighbourhoods. Consequently, noise patterns may be amplified together with meaningful image structures, as both appear as high-frequency intensity variations. Furthermore, convolution kernels do not explicitly model structural connectivity between distant image regions.

C. Graph Theory Fundamentals

A graph is formally defined as $G = (V, E)$, where V stands for vertices and E stands for edges. A graph can't contain no vertices, but it may contain no edges. In this paper, vertices represents the pixels of a digital image, while edges represents the difference in intensity between neighbouring pixels. This representation allows image structures and intensity relationships to be analyzed through graph connectivity.

There are a few ways a graph can be classified to depending on what components we view it from. Two of which that are relevant to this paper are undirected and weighted graphs.

1. Undirected Graph

   An undirected graph is a graph where each of its edges doesn't have an orientation. This causes and edge that connects two different vertices have a mutual relationship. That is the traversal from u to v is equivalent from v to u. This can be applied to the relationship between pixels, as two neighbouring pixels will both have the same differences.

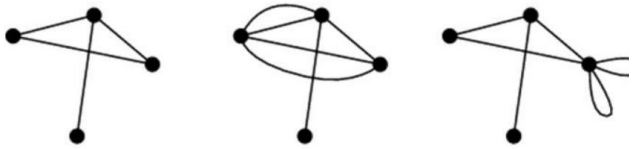

Fig 4. Undirected Graphs

(Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/20-Graf-Bagian1-2026.pdf)

2. Weighted Graph

A weighted graph is a graph where each of its edges are assigned a certain value, usually known as weight. These weights may represent various factors such as distance, cost, or in this case, intensity difference. With each pixel having its own intensity value, a difference between these two values are needed in order to determined the processing output of an image.

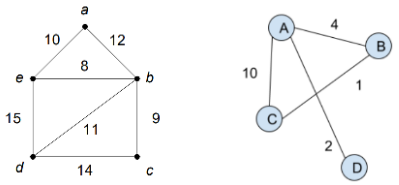

Fig 5. Weighted Graph

(Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/20-Graf-Bagian1-2026.pdf)

Two vertices in a graph are considered adjacent if they are connected by an edge. In graph-based image representations, adjacency is determined by the spatial neighbourhood relationships between pixels. Adjacent pixels are connected through graph edges, allowing local image structures to be represented as a network of interconnected vertices. These adjacency relationships define the connectivity of the image graph and provide a framework for analyzing intensity variations and structural patterns through graph-based processing methods.

As mentioned before, the 4-neighbourhood structure is one of the more common adjacency model in image processing. In this model, each pixel is connected to its horizontally and vertically adjacent pixels. For a pixel located at coordinate (x, y), its neighbouring pixels consist of $(x-1, y)$, $(x+1, y)$, $(x, y-1)$, and $(x, y+1)$. This neighbourhood model is commonly used due to its computational simplicity and efficient local connectivity representation. The edge weight between two adjacent pixels is calculated using the absolute difference of their intensity values,

$$w(u, v) = |I(u) - I(v)|$$

where $I(u)$ and $I(v)$ represent the intensity values of neighbouring pixels u and v.

D. Adaptive Edge Enhancement

Adaptive edge enhancement relies on the observation that significant image structures are often characterized by noticeable intensity differences between neighboring pixels. In graph-based image representation, this information is captured through edge weights, which are computed as the absolute difference between the intensity values of adjacent pixels,

$$w(u, v) = |I(u) - I(v)|$$

Larger weight values indicate stronger intensity transitions and are therefore more likely to correspond to meaningful image edges, boundaries, or structural details.

However, not all intensity differences represent important image features. Small variations may originate from minor texture changes, sensor imperfections, or image noise. To distinguish significant edges from insignificant intensity fluctuations, a threshold value (T) is introduced. An edge is considered significant only if its weight exceeds the predefined threshold,

$$w(u, v) > T$$

Conversely, when

$$w(u, v) \leq T$$

the intensity difference is regarded as insufficiently significant and is therefore excluded from the sharpening process. This thresholding mechanism prevents unnecessary enhancement of weak variations and reduces the likelihood of amplifying noise.

After significant edges have been identified, the sharpening process is performed adaptively based on the magnitude of the edge weight. Stronger edges receive a greater enhancement, while weaker edges receive little or no modification. The sharpening strength can be expressed as

$$\Delta = B \frac{w(u, v)}{255}$$

where (B) represents the maximum boost parameter and (Δ) denotes the intensity adjustment applied to the corresponding pixels. As a result, the enhancement process becomes adaptive to local image structures rather than applying a uniform sharpening effect across the entire image.

This graph-based adaptive strategy allows the sharpening process to emphasize meaningful image details while limiting the enhancement of insignificant intensity variations, resulting in clearer edge representation and improved visual quality.

E. Algorithmic Complexity

Traditional convolution-based sharpening processes each pixel using a fixed-size convolution kernel. For an image of size $n \times m$and a kernel of size $k \times k$, the computational complexity is generally represented as

$$O(nmk^2)$$

since every pixel must be processed using all k^2elements of the kernel.

In practical image sharpening applications, the kernel size is typically fixed, such as 3×3 or 5×5. Therefore, k can be treated as a constant, allowing the complexity to be simplified to

$$O(nm)$$

This indicates that the computational cost grows linearly with the number of pixels in the image. As a result, convolution-based sharpening is considered computationally efficient and well-suited for practical image processing tasks.

In graph-based image representation, each pixel is modeled as a vertex, resulting in approximately nm vertices for an image of size $n \times m$. Using a 4-neighbourhood adjacency model, each pixel is connected to its neighbouring pixels through graph edges. As a result, the total number of edges grows proportionally to the number of pixels in the image.

The construction of the image graph therefore requires computational complexity proportional to the number of vertices and edges:

$$O(|V| + |E|)$$

which is approximately linear with respect to image size.

The graph-based sharpening process operates by traversing graph edges and analyzing intensity differences between neighbouring pixels. Since each edge must be evaluated during processing, the sharpening stage has computational complexity proportional to the total number of edges:

$$O(|E|)$$

For large digital images, this complexity remains approximately linear relative to image size. However, graph-based approaches generally introduce additional computational overhead due to graph storage, adjacency representation, and edge-weight calculations.

Compared to traditional convolution-based sharpening, graph-based image processing generally requires higher computational and memory costs due to explicit graph construction and edge management. Nevertheless, graph representations provide greater flexibility for modeling structural relationships between pixels, allowing image enhancement methods to incorporate connectivity information beyond local convolution filtering.

III. Methodology

A. Research Workflow

This study begins by preparing a set data of digital images used as experiments. Each of these images will then be grayscaled and blurred using a Gaussian filter in order to simulate image degradation and reduce image sharpening.

After the preprocessing stage, the image is transformed into a graph structure, with pixels representing vertices and neighboring pixel relationships representing weighted edges. The edge weights are obtained by finding the absolute difference between the intensity values of adjacent pixels.

The graph will then be processed using the proposed adaptive sharpening algorithm. Edges with weight values greater than the threshold will be enhanced according to the value itself. Finally, the sharpened image is compared against the result produced by a traditional convolution-based sharpening kernel.

B. Image Preparation

The experiment uses digital images obtained from publicly available image datasets. Prior to graph construction, each image was converted from RGB color space into grayscale representation. Grayscale images simplify the processing pipeline by reducing each pixel to a single intensity value while preserving structural information necessary for edge analysis. The Gaussian blur operation was performed using a kernel size of 5×5 with a standard deviation of 1.5.

C. Graph Construction

After preprocessing, the image is transformed into a weighted graph representation. Using an $n \times m$ image with each pixel being represented as a vertex, a graph containing nm vertices is created. To establish the connections between the pixels, a 4-neighbourhood adjacency model is used. This graph representation captures local intensity variations and allows image structures to be analyzed through graph connectivity relationships.

D. Adaptive Sharpening Algorithm

The sharpening method used for this study runs on the weighted graph representation of the image. The weight of each edge of the graph will be compared to a predefined threshold value of T. An edge is considered significant if

$$w(u, v) > T$$

Only significant edges are selected for enhancement. This is done in order to prevent insignificant edges from being amplified unnecessarily.

The sharpening strength of each edge is determined adaptively according to its weight. The enhancement magnitude is computed as

$$\Delta = B \frac{w(u, v)}{255}$$

where B represents the maximum boost parameter and Δ denotes the intensity adjustment applied to the corresponding pixels.

If one pixel contains a higher value than its neighboring pixel, the brighter pixel is further increased while the darker pixel is decreased by the computed enhancement value. This operation increases the contrast between the edges, producing a sharpening effect.

After all graph edges have been processed, pixel values are constrained to the valid grayscale range between 0 and 255 to ensure that the resulting image remains visually valid.

E. Evaluation Procedure

To assess the effectiveness, the proposed graph-based adaptive sharpening method is compared with a traditional convolution-based sharpening technique.

The traditional method utilizes a standard sharpening kernel represented by

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Both methods are applied to the same set of blurred images. Analyzation of the resulting images are done based on visual quality, edge clarity, and the preservation of image details. In addition, execution durations are also noted to compare the complexity of graph-based sharpening and conventional convolution-based sharpening.

IV. Implementation

A. Libraries and Development Environment

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
import time
```

Fig 6. Python Libraries
(Source: Author)

The proposed graph-based sharpening method was implemented using Python. Several external libraries were used to support image processing, graph construction, numerical computation, visualization, and runtime measurement.

TABLE I.
Program Libraries

Library	Purpose
OpenCV (cv2)	Image loading, grayscale conversion, Gaussian blur, and convolution-based sharpening
NumPy	Numerical computations and matrix manipulation
NetworkX	Construction and traversal of weighted image graphs
Matplotlib	Visualization of experimental results
time	Runtime measurement and performance analysis

The machine used in this study have the following specifications:

- Operating System: Windows 11
- Processor: 13th Gen Intel(R) Core(TM) i7-13650HX (2.60 GHz)
- Memory: 32 GB RAM

B. Image Preprocessing

```python
# Load Image
image = cv2.imread("image.png")
if image is None:
    raise Exception("Image not found!")

# Convert to Grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Add Blur
blurred = cv2.GaussianBlur(
    gray,
    (5, 5),
    1.5
)
```

Fig 7. Image Preprocessing
(Source: Author)

The code snippet shown presents the preprocessing stage of the input image before the sharpening process is performed. First, the image is loaded using OpenCV and converted from RGB color space into a grayscale representation. This conversion reduces each pixel to a single intensity value, simplifying graph construction and edge-weight computation.

To simulate image degradation and reduce edge sharpness, a Gaussian blur operation with a kernel size of 5×5 and a standard deviation of 1.5 is applied. The blurred image serves as the input for both the proposed graph-based sharpening method and the traditional convolution-based sharpening method.

C. Graph Construction

```python
# Graph Construction
height, width = blurred.shape

for y in range(height):
    for x in range(width):
        intensity = int(blurred[y, x])
        G.add_node(
            (x, y),
            intensity=intensity
        )

        if x < width - 1:
            diff = abs(
                intensity
                - int(blurred[y, x + 1])
            )
            G.add_edge(
                (x, y),
                (x + 1, y),
                weight=diff
            )

        if y < height - 1:
            diff = abs(
                intensity
                - int(blurred[y + 1, x])
            )
            G.add_edge(
                (x, y),
                (x, y + 1),
                weight=diff
            )
```

Fig 8. Graph Construction
(Source: Author)

The code snippet shown presents the process of transforming the grayscaled image into a weighted graph representation. The algorithm iterates through every pixel in the image and creates a corresponding vertex in the graph. Each vertex stores the pixel intensity value as an attribute. Therefore, an image of size $n \times m$ produces a graph containing $n \times m$ vertices.

D. *Adaptive Edge Enhancement*

```
# Adaptive Graph Sharpening
result = blurred.copy().astype(np.int32)

threshold = 20
max_boost = 45
processed_edges = 0

graph_start = time.time()

for (u, v, data) in G.edges(data=True):
    weight = data["weight"]

    if weight > threshold:
        x1, y1 = u
        x2, y2 = v

        processed_edges += 1

        val1 = int(blurred[y1, x1])
        val2 = int(blurred[y2, x2])

        delta = int(
            max_boost
            * weight
            / 255
        )

        if delta < 1:
            delta = 1

        if val1 > val2:
            result[y1, x1] += delta
            result[y2, x2] -= delta
        else:
            result[y1, x1] -= delta
            result[y2, x2] += delta
```

Fig 9. Adaptive Graph Sharpening
(Source: Author)

The code snippet shown implements the proposed adaptive graph-based sharpening method. The algorithm utilizes the weighted graph representation constructed in the previous stage, where edge weights correspond to intensity differences between neighboring pixels. A threshold value is first defined to determine whether an edge represents a meaningful intensity transition. Only edges whose weights exceed the threshold are processed.

The sharpening process is performed by increasing the intensity of the brighter pixel while decreasing the intensity of the darker pixel. This operation enlarges the contrast between neighboring pixels, making image edges appear more distinct. By scaling the enhancement strength according to the edge weight, the proposed method is able to adapt its sharpening behavior to local image structures rather than applying a uniform enhancement across the entire image.

E. *Traditional Sharpening*

```
# Traditional Sharpening
kernel = np.array([
    [0, -1, 0],
    [-1, 5, -1],
    [0, -1, 0]
])

traditional = cv2.filter2D(
    blurred,
    -1,
    kernel
)
```

Fig 10. Traditional Sharpening
(Source: Author)

The code snippet shown implements the traditional convolution-based sharpening method used as a baseline for comparison. A sharpening kernel is first declared to emphasize local intensity variations and image edges. The sharpening operation is then applied using a two-dimensional convolution process, where the kernel is evaluated across every pixel in the image. Since the same kernel is applied uniformly throughout the image, the sharpening effect remains fixed regardless of local image characteristics. The resulting image is used as a reference for evaluating the effectiveness of the proposed graph-based sharpening approach.

V. Tests and Results

A. *Experimental Setup*

Fig 11. Potrait of Charlie Chaplin

Fig 12. Image of A Wooden Door

(Source Fig 11: https://www.chegg.com/homework-help/questions-and-answers/enclosed-find-five-images-noisy1jpg--noisy5jpg--given-noisy-images-part-try-find-noise-typ-q51367086)

(Source Fig 12: https://r0k.us/graphics/kodak/kodim02.html/)

Two images containing different structural characteristics were used for evaluation. Figure 6 shows a grayscale image with a resolution of 512 x 619 pixels, containing a portrait of Charlie Chaplin with a lot of noise in both the person and the background. Meanwhile figure 7 shows a color image with a resolution of 768 x 512 pixels, containing a close up look of a wooden door.

TABLE II.
PROGRAM PARAMETERS

Parameter	Value
Threshold	20
Max Boost	45
Gaussian Kernel	5 x 5
Sigma	1.5

A threshold value of 20 and a maximum boost value of 45 were chosen as moderate parameter values that allow significant image edges to be enhanced while limiting unnecessary sharpening of weaker intensity variations. The selected values provide a balance between sharpening strength and noise amplification.

The portrait image of Charlie Chaplin was selected because it contains a noticeable amount of noise in both the subject and

background regions. This characteristic makes it suitable for evaluating how the proposed graph-based sharpening method and the traditional convolution-based approach behave when processing noisy images. In contrast, the wooden door image was chosen because it contains numerous fine structural details and textures that are not primarily caused by noise. This image allows the sharpening performance of both methods to be assessed on meaningful image details and edge structures, highlighting their ability to enhance subtle features while preserving visual quality.

B. Visual Comparison

Fig 13. Processed Image of Charlie Chaplin

(Source: Author)

The figure above presents the results obtained from the Charlie Chaplin image, consisting of the original image, the blurred image, and the sharpened images produced by the traditional and graph-based approaches. A noticeable difference can be observed in the sharpening quality of the two methods. The traditional sharpening approach produces a sharper and clearer image, with facial features and object boundaries appearing more distinct. In contrast, the graph-based approach generates a more conservative sharpening effect, resulting in an image that remains less sharp than both the original image and the traditional sharpening result.

Despite producing a less aggressive sharpening effect, the graph-based approach demonstrates a different behavior in terms of noise enhancement. The traditional method amplifies not only meaningful image edges but also a significant amount of noise, which can be observed from the increased visibility of bright speckles in the dark background and facial regions. Meanwhile, the graph-based approach introduces less noise amplification due to its selective edge enhancement mechanism. As a result, the main subject remains visually prominent while smoother regions experience less unnecessary sharpening.

Fig 14. Processed Image of A Wooden Door

(Source: Author)

The figure above presents the results obtained from the wooden door image, consisting of the original image, the blurred image, and the sharpened images produced by the traditional and graph-based approaches. Similar to the results observed for the Charlie Chaplin image, the traditional sharpening method produces a noticeably sharper image than the graph-based approach. The graph-based result remains closer to the blurred image and appears slightly darker than the other images, indicating a more conservative enhancement process.

Unlike the Charlie Chaplin image, the wooden door image contains numerous structural details and texture patterns on the wooden surface. The traditional sharpening method enhances these details more aggressively, making the grooves, edges, and surface textures appear clearer and more defined. In contrast, the graph-based approach preserves a smoother appearance, causing some of the finer details to be less visible. While this reduces the prominence of small texture variations, it also avoids excessive enhancement and produces a more uniform visual result.

C. Runtime Comparison

TABLE III.

RUNTIME COMPARISON

Image	Traditional	Graph-Based
Charlie Chaplin	0.000298 s	1.524944 s
Wooden Door	0.000301 s	1.919795 s

Based on the runtime comparison shown in Table III, the traditional sharpening approach is significantly faster than the proposed graph-based approach. For both test images, the traditional method completes the sharpening process in approximately 0.0003 seconds, whereas the graph-based method requires more than 1.5 seconds. This substantial difference is primarily caused by the graph construction stage. Unlike traditional sharpening, which directly applies a convolution kernel to the image matrix, the graph-based approach must first construct a weighted graph representation of the image, where each pixel is represented as a vertex and connected to its neighboring pixels through weighted edges.

After the graph has been constructed, the algorithm must traverse and evaluate each edge to determine whether it satisfies the sharpening criteria. These additional operations introduce a significant computational overhead, resulting in considerably longer execution times. Furthermore, even when excluding the graph construction stage, the graph-based sharpening process alone remains substantially slower than the traditional approach. This behavior is consistent with the additional complexity associated with graph representation and edge-based processing. The results also suggest that image size has a noticeable impact on the runtime of the graph-based method. The Wooden Door image requires a longer processing time than the Charlie Chaplin image, which is expected because larger images contain more pixels, vertices, and edges that must be processed during graph construction and sharpening.

D. Results Analysis

Based on the experimental results, the graph-based approach demonstrates a more adaptive and selective enhancement process by sharpening only regions associated with significant intensity differences. As a result, the method preserves the overall image structure while reducing unnecessary

enhancement in noise-prone regions compared to the traditional convolution-based approach.

However, the proposed method also exhibits several limitations. The most significant drawback is its weaker sharpening effect compared to traditional sharpening. In both test images, the resulting graph-based outputs appear less sharp and contain fewer enhanced fine details than those produced by the convolution-based method. Consequently, some image features remain visually closer to the blurred image than to the original image.

The results reveal a clear trade-off between sharpening strength, noise amplification, and detail preservation. While the traditional approach generates sharper images and emphasizes fine details more effectively, it also amplifies a greater amount of image noise. In contrast, the graph-based approach applies a more conservative enhancement strategy, reducing noise amplification at the expense of overall sharpness and detail visibility.

From a computational perspective, the graph-based method is significantly slower due to the overhead associated with graph construction and edge processing. This additional complexity makes the approach considerably less efficient than traditional convolution-based sharpening, particularly when processing larger images.

VI. Conclusion

This paper presented a graph-based representation of digital images for adaptive edge sharpening. In the proposed method, each pixel was modeled as a vertex and connected to its neighbouring pixels through weighted edges based on intensity differences. Significant edges were then selectively enhanced using an adaptive sharpening mechanism.

Experimental results demonstrated that the graph-based approach is capable of performing selective edge enhancement while reducing unnecessary sharpening in noise-prone regions. Compared to the traditional convolution-based sharpening method, the proposed approach produced a more conservative sharpening effect and amplified less image noise. However, the resulting images generally appeared less sharp and contained fewer enhanced fine details than those produced by the traditional method.

Runtime evaluation showed that the graph-based approach requires significantly longer processing times due to graph construction and edge-based processing operations. While this additional computational complexity reduces efficiency, the method illustrates how graph theory concepts can be applied to image processing tasks and provides a foundation for more advanced graph-based enhancement techniques.

Future work may explore alternative graph structures, adaptive edge-weighting schemes, graph Laplacian operators, or graph signal processing methods to improve sharpening quality while maintaining the advantages of selective enhancement and noise suppression.

Attachment

Github: Source Code of “Graph-Based Representation of Digital Images for Adaptive Edge Sharpening”
Here

Acknowledgment

The author would like to express sincere gratitude to God Almighty for His blessings and guidance throughout the completion of this paper.

The author would also like to extend heartfelt appreciation to Dr. Ir. Rinaldi Munir, M.T., lecturer of IF1220 Discrete Mathematics, for his guidance, instruction, and encouragement throughout the semester. Special thanks are also given to the author's parents for their unwavering support during the preparation of this paper.

In addition, the author would like to thank fellow students from classes K01 and K02 for the valuable insights, discussions, and support they provided. Finally, the author wishes to express deepest gratitude to his partner, who has always been a constant source of encouragement and support, and who remained by his side through both the challenges and successes encountered during the writing of this paper.

References

[1] R. C. Gonzalez and R. E. Woods, Digital Image Processing, 4th ed. Hoboken, NJ, USA: Pearson, 2018.

[2] T. D. Pham, "Kriging-Weighted Laplacian Kernels for Grayscale Image Sharpening," IEEE Access, vol. 10, pp. 57094-57106, May 2022.

[3] Y. Chen et al., "Plant Image Recognition with Deep Learning: A Review," Computers and Electronics in Agriculture, vol. 212, article 108072, Jul. 2023.

[4] G. Cheung, E. Magli, Y. Tanaka, and M. Ng, "Graph Spectral Image Processing," arXiv:1801.04749, Jan. 2018.

[5] R. Munir, "Graf Bagian 1," Institut Teknologi Bandung, 2026. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/20-Graf-Bagian1-2026.pdf. [Accessed: Jun. 12, 2026].

[6] R. Munir, "Graf Bagian 2," Institut Teknologi Bandung, 2026. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/21-Graf-Bagian2-2026.pdf. [Accessed: Jun. 12, 2026].

Statement

I hereby declare that the paper I wrote is my own writing, not an adaptation or translation of someone else's paper, and is not plagiarized.

Bandung, 17 June 2026

Kevin Sie, 13525053