CENG466 - Report

1st Umut Dağhan DAĞDAŞ
Computer Engineering
Middle East Technical University
Ankara, Turkey
daghan.dagdas@metu.edu.tr

2nd Halil Hilmi ÇAKANEL
Computer Engineering
Middle East Technical University
Ankara, Turkey
cakanel.hilmi@metu.edu.tr

***Abstract*—Abstract—This document shows some examples of edge detection filters, frequency domain filters, blurring filters, and image compression techniques such as Haar and Discrete Cosine Transform.**

***Index Terms*—Edge detection, image enhancement, image compression, Sobel filters, Haar transform, DCT, Gaussian filtering, noise reduction.**

I. Introduction

In this report, we analyze the problems in the Fundamentals of Image Processing and present our solutions to them in detail. There are three problems that are related to pattern extraction, image enhancement, and image compression respectively. We go over them step by step and share our findings with relevant figures for better illustration.

II. Problem 1 - Pattern Extraction

In this section, we explore the application of various edge detection filters to our images. Additionally, we apply these filters to blurred and binarized versions of the images and analyze their effects and results in detail.

There are three types of edge detection filters we use. These are Sobel, Prewitt, and Roberts filters. Sobel and Prewitt filters are 3x3 filters, and each of them calculates horizontal and vertical gradients. On the other hand, Roberts filter is a 2x2 filter that calculates diagonal gradients. These filters are given below.

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

Fig. 1: Sobel filters: G_x (horizontal gradient) and G_y (vertical gradient).

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

Fig. 2: Prewitt filters: G_x (horizontal gradient) and G_y (vertical gradient).

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

Fig. 3: Roberts filters: G_x (diagonal gradient along one direction) and G_y (diagonal gradient along the perpendicular direction).

We apply these filters to our images and get the following results. For simplicity, we show the results for one of the images.

(a) Original Image (b) Sobel Filters Applied

(c) Prewitt Filters Applied (d) Roberts Filters Applied

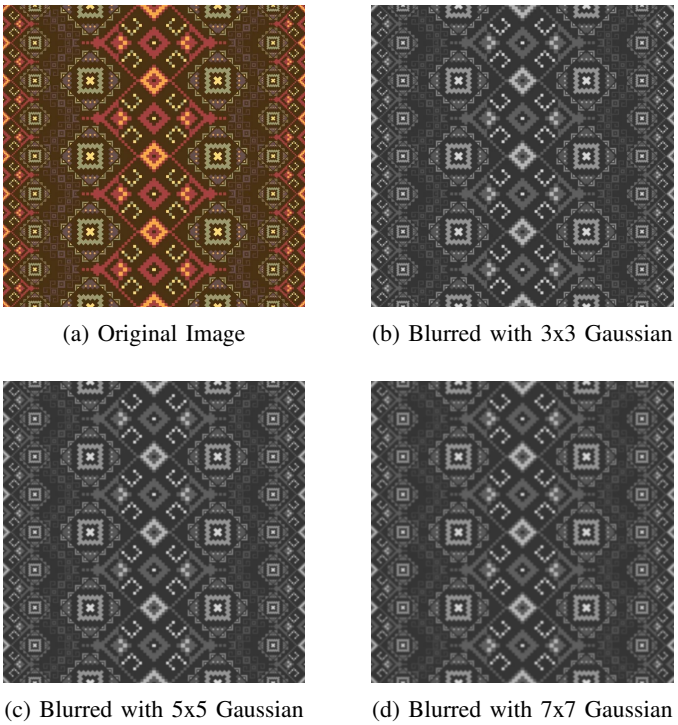
Fig. 4: Comparison of the original image and the results after applying Sobel, Prewitt, and Roberts filters.

We observe that Roberts filters produce sharper and thinner edges compared to Sobel and Prewitt filters. This is because they use 2x2 matrices, which focus on a smaller area and are capable of detecting finer details. However, this also makes them more sensitive to noise in the image, increasing the

likelihood of detecting false edges.

There is no easily observable difference between the outputs of Sobel and Prewitt filters. The main difference between them is the fact that Sobel filters produce slightly sharper outputs because they assign greater weight to the center pixels, which enhances the prominence of edges. This makes Sobel filters better at suppressing noise and detecting prominent edges compared to Prewitt filters, which use uniform weights and are more suitable for simpler or less noisy images.

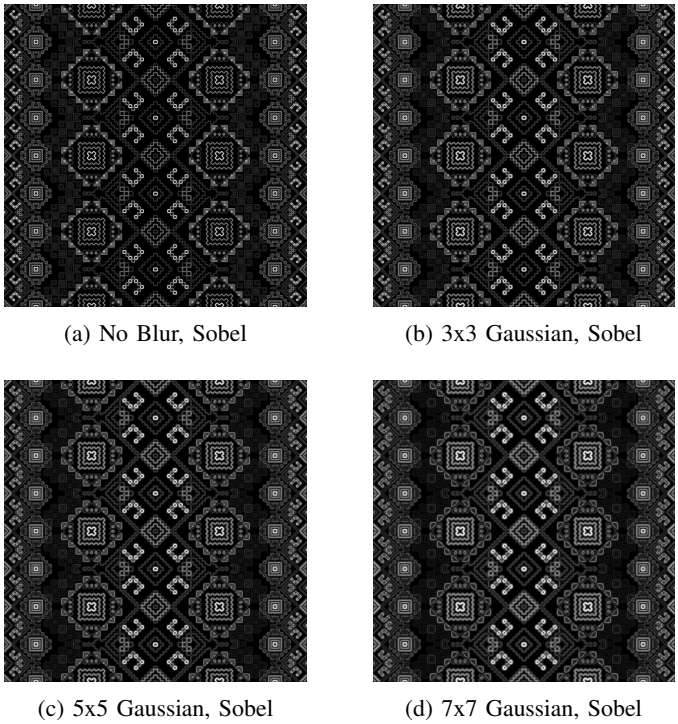
In the next step, we blur the images using three different Gaussian filters with sizes 3x3, 5x5, and 7x7. Then, we apply Sobel, Prewitt, and Roberts filters to the blurred images.



(a) Original Image (b) Blurred with 3x3 Gaussian

(c) Blurred with 5x5 Gaussian (d) Blurred with 7x7 Gaussian

Fig. 5: Comparison of the original image with the blurred images with different filter sizes.



(a) No Blur, Sobel (b) 3x3 Gaussian, Sobel

(c) 5x5 Gaussian, Sobel (d) 7x7 Gaussian, Sobel

Fig. 6: Comparison of the results after applying Sobel filters to different blurred images

In this blurring approach, as shown in Fig. 6, we observe that as the size of the Gaussian filter increases, the image becomes smoother, causing edge detection filters to lose finer details. Instead, they focus on detecting more prominent edges. The difference can easily be seen when the image generated by applying 3x3 Gaussian and Sobels filter is compared with the image generated by applying 7x7 Gaussian and Sobel filters.

Again, for simplicity, we have omitted the outputs of the Prewitt and Roberts filters applied to the blurred images in the report, as their effects would essentially be the same.

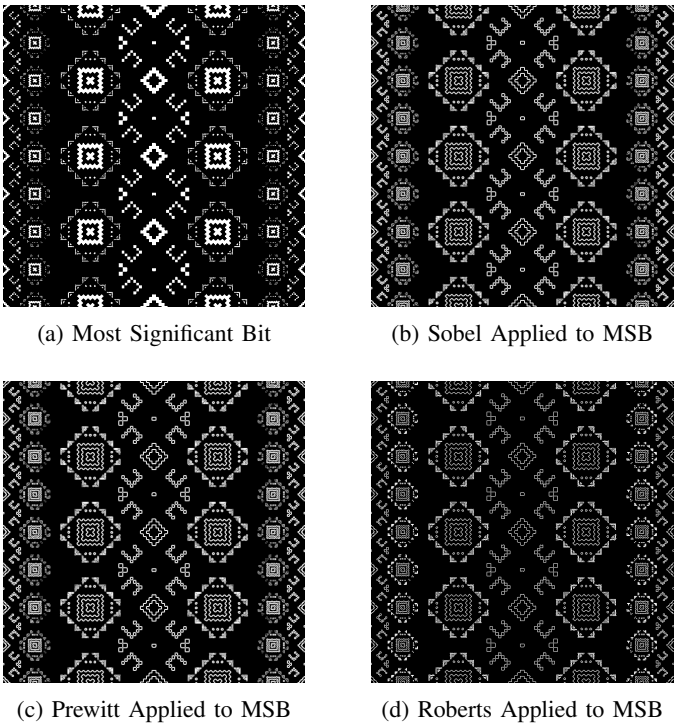
As the final step of this section, we apply bit-plane slicing to the original image using the most significant bit. This process results in a binarized, noisy image, as shown below.

(a) Original Image (b) Most Significant Bit

Fig. 7: Comparison of the original image with the image obtained through bit-plane slicing using the most significant bit

When edge detection filters are applied to this binarized image, the results shown in Fig. 8 are obtained. Due to the noise present in the binarized image, the results are not satisfactory. It is evident that there are numerous false edges that do not exist in the original image. Additionally, it can be observed that our findings from the previous steps are consistent here: Roberts filters produce sharper edges, while Sobel and Prewitt filters generate somewhat smoother edges. Therefore, there is more noise in the image produced by Roberts filters due to the existing noise and the filters' tendency to detect these noise artifacts as edges.

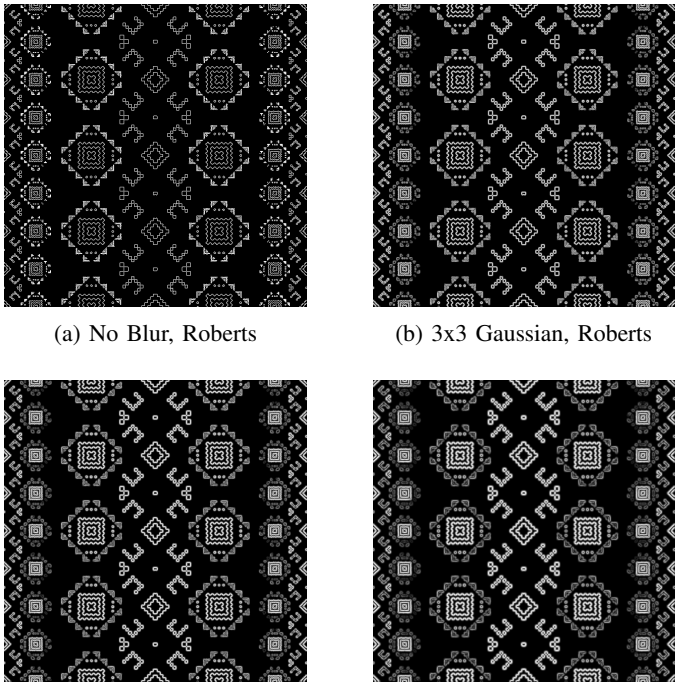
In order to eliminate the noise and produce smoother and more accurate edges, we apply Gaussian filters to the binarized image as we have done to the original image previously. The results can be seen in Fig. 9. When the size of the Gaussian filter increases, we detect smoother and more accurate edges. This time we provided the results only for Roberts filters, for which the results are more apparent.



(a) Most Significant Bit (b) Sobel Applied to MSB

(c) Prewitt Applied to MSB (d) Roberts Applied to MSB

Fig. 8: Comparison of the MSB image with results after applying Sobel, Prewitt, and Roberts filters.



(a) No Blur, Roberts (b) 3x3 Gaussian, Roberts

(c) 5x5 Gaussian, Roberts (d) 7x7 Gaussian, Roberts

Fig. 9: Comparison of the results after applying Roberts filters to different blurred images

As a result, in this section, we applied different edge detection filters, including Sobel, Prewitt, and Roberts filters, to original, blurred, and binarized images. Through our analysis, we observed the unique features of each filter and how they perform in various situations.

Sobel and Prewitt filters, as 3x3 filters, provided smoother and less noisy edges, making them better suited for detecting prominent edges in cleaner images. On the other hand, Roberts filters, with their 2x2 design, were able to detect finer details but were more sensitive to noise, which often resulted in false edges, especially in noisy or binarized images.

When we applied Gaussian blurring, we noticed that increasing the size of the Gaussian filter reduced noise and produced smoother edges. However, it also caused the filters to lose finer details, showing that there is a trade-off between noise reduction and detail preservation. Choosing the right filter size depends on the specific needs of the task.

Finally, when we applied edge detection filters to the binarized images using bit-plane slicing, we found that the results were heavily affected by noise, leading to many false edges. Gaussian blurring helped to improve these results, especially for Roberts filters, but it also showed the importance of preprocessing steps in edge detection.

III. Problem 2 - Image Enhancement

In this section, we apply some image enhancement techniques in order to remove different kinds of noise in images.

(a) Figure 1 (b) Figure 2 (c) Figure 3

Fig. 10: Given images with different kinds of spatial noise

We first examined the given images in the spatial and Fourier domains and decided to apply the enhancement techniques in the spatial domain to the first and second images,and in the Fourier domain to the third image. We have learned that these kind of noise in Figure 1 and Figure 2 is called salt-and-pepper noise.

We applied two filters in the spatial domain : Gaussian [1] and median [2] filters.

Median filter can be perceived as improved version of mean filter. They both take neighbor pixels into consideration when deciding the color. But, the mean filter takes the mean, whereas the median filter takes the median value. Thus, the median filter is better than the mean filter in preserving useful details.

Gaussian filter is also like mean filter also used for blurring image and reducing noise. But, it uses a different kernel representing a bell-shaped hump which is also known as Mexican hat.

Gaussian kernel in 2-D has the form

$$G(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

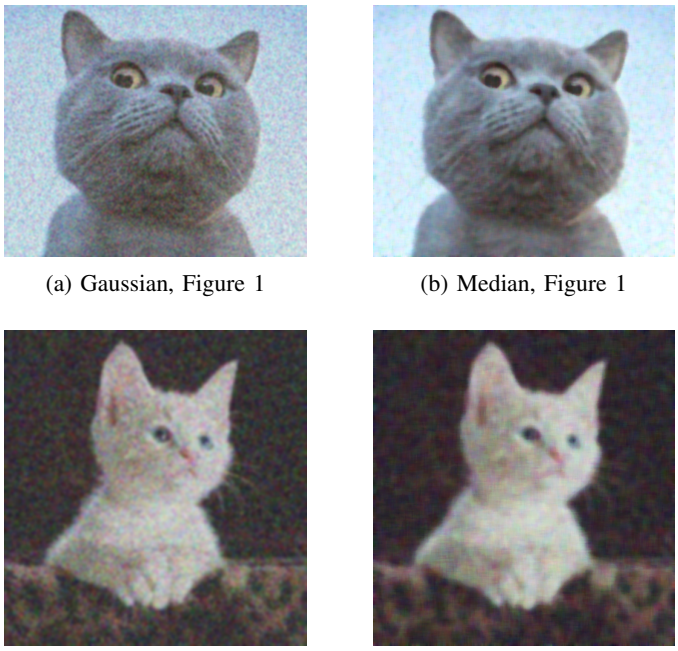
```
        img_r = img[:, :, 0]
        img_g = img[:, :, 1]
        img_b = img[:, :, 2]

        img_r_median = cv2.medianBlur(img_r,
            13)
        img_g_median = cv2.medianBlur(img_g,
            13)
        img_b_median = cv2.medianBlur(img_b,
            13)

        img_r_gaussian =
            cv2.GaussianBlur(img_r, (15,15),
            0)
        img_g_gaussian =
            cv2.GaussianBlur(img_g, (15,15),
            0)
        img_b_gaussian =
            cv2.GaussianBlur(img_b, (15,15),
            0)

        img_final_median =
            np.dstack((img_r_median,
            img_g_median, img_b_median))
        img_final_gaussian =
            np.dstack((img_r_gaussian,
            img_g_gaussian, img_b_gaussian))
        write_image(img_final_median,
            f"spatial/b{i+1}_median.png")
        write_image(img_final_gaussian,
            f"spatial/b{i+1}_gaussian.png")
```

We have applied the Gaussian and Median filters to the channels separately and combined them together.



(a) Gaussian, Figure 1 (b) Median, Figure 1

(c) Gaussian, Figure 2 (d) Median, Figure 2

Fig. 11: Output of the spatial domain enhancement techniques

We have applied 13x13 kernel in Median filtering and 15x15 kernel in Gaussian Filtering. Increasing the size of kernel results in more blurry image while removing noises better. These kernel sizes are close to the optimal where we can both store the information and remove the noises at the same time.

Then, we have applied three filters in Fourier domain: Ideal Low Pass filter, Band-pass filter and Band-reject filter.

Ideal Low Pass filter is designed to attenuate the frequencies higher than a specific cut-off frequency while preserving the lower frequencies.

Band-pass filter passes the frequencies within a specific range and attenuates others.

Band-reject filter can be considered as the compliment of band-pass filter which attenuates the frequencies within a specific range and passes others.

We first applied Fourier transform to the image in order to apply Fourier domain image enhancement techniques. Then, examined the channels in Fourier and tuned our filtering parameters. We combined the channels after applying the filters.

(a) Ideal Low Pass (b) Band-pass (c) Band-reject

Fig. 12: Output of the Fourier domain enhancement techniques

Since the image does not contain noise that can be suppressed in a circular or frequency-specific manner, and the main details of the image are concentrated in the center of the frequency domain, we observed no practical difference between the filters.

The central region of the image's frequency spectrum contains the most critical information, so it cannot be discarded. For the band-pass filter, we set the inner cutoff parameter to 0, ensuring that the central frequencies were included. This caused the band-pass filter to behave similarly to an ideal low-pass filter in this scenario. Similarly, the band-reject filter was designed to retain the same central portion of the frequency spectrum as the other filters, further contributing to the lack of significant differences in their outcomes.

IV. Problem 3 - Image Compression

In this section we apply compression techniques to images using Haar Transform and Discrete Cosine Transform (DCT).

We start with Haar transform. 1D Haar wavelet function is shown below.

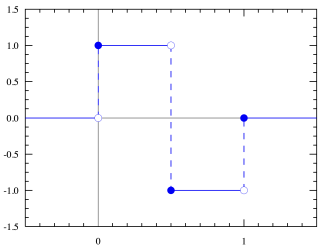


Fig. 13: 1D Haar Wavelet

We are supposed to generate basis matrices based on this simple wavelet function by scaling and translating as shown below. [3]

$$h_0(x) = h_{00}(x) = \frac{1}{\sqrt{N}}$$

$$h_n(x) = h_{jk}(x) = \begin{cases} \frac{2^{j/2}}{\sqrt{N}}, & \frac{k-1}{2^j} \le x < \frac{k-0.5}{2^j}, \\ -\frac{2^{j/2}}{\sqrt{N}}, & \frac{k-0.5}{2^j} \le x < \frac{k}{2^j}, \\ 0, & \text{otherwise.} \end{cases}$$

Fig. 14: Compact Dyadic Wavelet Basis Functions

In the next step, these basis wavelet functions are discretized and Haar transformation matrix H_{NN} is obtained.

$$\mathbf{H}_8 = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 \end{bmatrix}$$

Fig. 15: An example of Haar matrix for N=8

In order to produce the transformed image, we are supposed to apply the transformation below.

$$F_H = H_{NN} f H_{NN}^T$$

However, since this only applies to square images, we use the generalized version for MxN images:

$$F_H = H_{MM} f H_{NN}^T$$

This Haar transformation divides the image into four sub-bands as shown in Fig. 13. LL (Low-Low) captures the low-frequency content in both horizontal and vertical directions, representing the smoothed version of the image. LH (Low-High): Captures low-frequency content horizontally and high-frequency content vertically, highlighting horizontal edges. HL (High-Low): Captures high-frequency content horizontally and low-frequency content vertically, highlighting vertical edges. HH (High-High): Captures high-frequency content in both directions, highlighting diagonal edges or fine textures.

$$\text{HWT}(f) = \begin{bmatrix} LL & LH \\ HL & HH \end{bmatrix}$$

Fig. 16: Haar Wavelet Transform (HWT) decomposes an image f into four sub-bands: LL (Low-Low), LH (Low-High), HL (High-Low), and HH (High-High).

After producing the transformed image, we flatten all the subbands into one dimensional array and concatenate them.

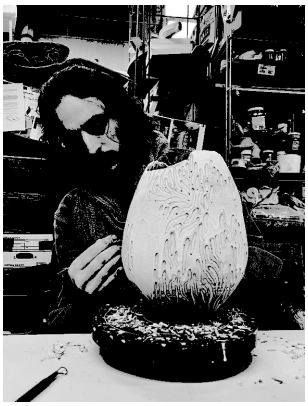
```
# Haar Transform
LL, LH, HL, HH = haar_transform(img)

LL_flat = LL.flatten()
LH_flat = LH.flatten()
HL_flat = HL.flatten()
HH_flat = HH.flatten()
all_haar = np.concatenate([LL_flat, LH_flat,
    HL_flat, HH_flat])
```

After the concatenation, we sort all the elements in the concatenated array and only retain the highest N% coefficients and set the rest to zero. After that, we undo all the operations and reconstruct the original image based on the retained coefficients. The results are given in Fig. 14.

(a) Original Image

(b) Haar Recon. 1 Percent



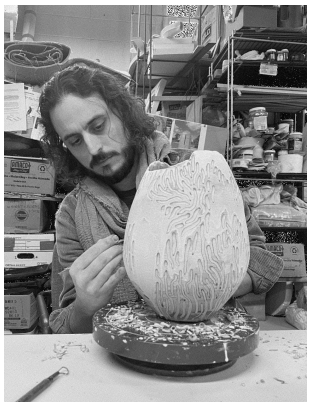
(c) Haar Recon. 10 Percent

(d) Haar Recon. 50 Percent

Fig. 17: Comparison of the original image and the results after reconstructing Haar transformed images

(a) Original Image

(b) DCT Recon. 1 Percent



(c) DCT Recon. 10 Percent

(d) DCT Recon. 50 Percent

Fig. 18: Comparison of the original image and the results after reconstructing Discrete Cosine transformed images

The sizes of the saved compressed files are 66 KB, 389 KB, and 2.3 MB, respectively, for 1 percent, 10 percent, and 50 percent. These files are stored in the simple .npz format, so they do not reflect the sizes achievable with image compression methods like JPG. It is evident that significant compression occurs when the percentage of retained coefficients is decreased. However, we also start losing the context of the original image.

We repeat the process for DCT using

$$F_C = C_{NN} f C_{NN}^T$$

where C_{NN} is defined as:

$$C_{NN}(k,l) = \begin{cases} \frac{1}{\sqrt{N}}, & \text{for } l = 0, \\ \sqrt{\frac{2}{N}} \cos\left(\frac{(2k+1)l\pi}{2N}\right), & \text{otherwise.} \end{cases}$$

The square image situation mentioned previously also applies here, so we use:

$$F_C = C_{MM} f C_{NN}^T$$

When DCT is applied to the image, the following results are obtained.

The sizes of the saved compressed files are 180 KB, 1.5 MB, and 7.3 MB, respectively, for 1 percent, 10 percent, and 50 percent. The most noticeable difference between the reconstruction methods is that, even in the 1 percent case, DCT is able to reconstruct the image without losing context compared to Haar. Their difference is more distinguishable when mean squared errors are compared.

```
Image: c1, N=1%
  Haar MSE: 19662.1741
  DCT MSE: 324.5169

Image: c1, N=10%
  Haar MSE: 5816.8646
  DCT MSE: 77.2048

Image: c1, N=50%
  Haar MSE: 1.5731
  DCT MSE: 4.7310
```

Fig. 19: MSE between original image and reconstructed images

It is seen that DCT outperforms Haar on small percentages. However, for larger percentages Haar is significantly better than DCT in terms of both MSE and compression ratios. The choice of transform depends on the type of image and the

desired balance between compression efficiency and visual quality.

V. Conclusion

To conclude, we have implemented, examined and reported some advanced techniques in image processing, including edge detection, spatial and frequency domain filtering, and compression. Our focus was to apply these methods to various image-related challenges and evaluate their effectiveness:

Edge Detection: We implemented three classic edge detection filters. Sobel, Prewitt, and Roberts to highlight edges in grayscale images. Each method successfully extracted horizontal, vertical, and diagonal edges, with differences in sensitivity and noise robustness. Sobel and Prewitt filters provided smoother results, while Roberts was more sensitive to sharp transitions. Frequency Domain Filtering: Fourier transforms were applied to analyze images in the frequency domain. We designed filters for low-pass, band-pass, and band-reject operations, allowing us to enhance or suppress specific frequency components. This approach proved effective in removing noise and isolating key image features. However, filter parameters like radius required careful tuning for optimal results. Compression Using Wavelet and DCT: We explored image compression through wavelet (Haar) and Discrete Cosine Transform (DCT). Both techniques retained critical image details while significantly reducing data size by preserving only a percentage of the top coefficients. The trade-off between compression ratio and image quality was assessed using metrics like Mean Squared Error (MSE). Wavelet compression demonstrated better spatial localization, while DCT provided superior energy compaction for smooth regions. Each task highlighted the trade-offs and challenges associated with specific methods, from balancing sensitivity and noise suppression in edge detection to fine-tuning parameters for effective filtering and compression. This hands-on exploration reinforced the importance of selecting the right technique based on the specific requirements of the application.

References

[1] R. Fisher, "Gaussian smoothing and its applications in image processing," The Hypermedia Image Processing Reference (HIPR2), The University of Edinburgh. [Online]. Available: https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm

[2] R. Fisher, "Median filtering and its applications in image processing," The Hypermedia Image Processing Reference (HIPR2), The University of Edinburgh. [Online]. Available: https://homepages.inf.ed.ac.uk/rbf/HIPR2/median.htm

[3] ScienceDirect, "Haar Transform and its applications in image processing," [Online]. Available: https://www.sciencedirect.com/topics/computer-science/haar-transform