CENG 466 - Image Processing
THE2

Mert Uludoğan
2380996

Yiğitcan Özcan
2521847

I. Question 1: Pattern Extraction

Task Summary: The goal of this task is to extract patterns from two rug images using image processing techniques. The steps involve converting images to grayscale, applying edge detection filters (Sobel, Roberts, Prewitt), blurring with different kernel sizes, and analyzing binary images created from the most significant bit (MSB).

Techniques Overview:

- **Grayscale Conversion:** Simplifies the image by retaining only intensity values, making further processing more efficient.
- **Edge Detection Filters:** The following table shows the kernels for the edge detection methods used:

Filter	Kernel
Sobel (Horizontal)	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$
Sobel (Vertical)	$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
Roberts (Diagonal 1)	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Roberts (Diagonal 2)	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$
Prewitt (Horizontal)	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$
Prewitt (Vertical)	$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$

TABLE I
Edge Detection Kernels

- **Blurring:** Smoothens the image by averaging pixel values in a neighborhood, reducing noise and emphasizing patterns. Larger kernel sizes produce stronger blurs.
- **MSB Extraction:** Simplifies grayscale images into binary by isolating the most significant bit, highlighting dominant features for further analysis.

Implementation Results and Analysis:

- **Analysis 1: Filtering vs Smoothing-Filtering** Initial stage is applying the filters to raw images. The results suffice expectations of their definitions. It is more informative to compare them with the first smoothened then filtered ones. For example, Figure 1 shows a Prewitt filtering with non-blurred and blurred ones. There is not any significant changes at first sight. However, some areas, especially small squares between relatively bigger squares, darkened in blurred implementation. This is due to loss of details resulting from blurring. This loss is a choice in exchange for de-noising. Due to the figures in this implementation have not any significant blurs, we can not interpret the benefits of it. However, we can evaluate loss of little square details on those rug images as de-noising for better understanding. In the sense of those little details are the part of interest, blurring can be evaluated as accuracy advancing or receding depending on the goal. If we want to find most representative main edges it is good but if we look for details, it is not that good. The detail loss is more definitive on Figure 2. This time we applied Sobel filter on non-blurred and blurred ones of the same image. On the non-blurred filtered image, the details on the little squares are more significant than blurred one. However, the blurred one seems more representative to eye overall. If someone squint and try to catch overall structure of the pattern of those images, they recognize that blurred image is better on this task.

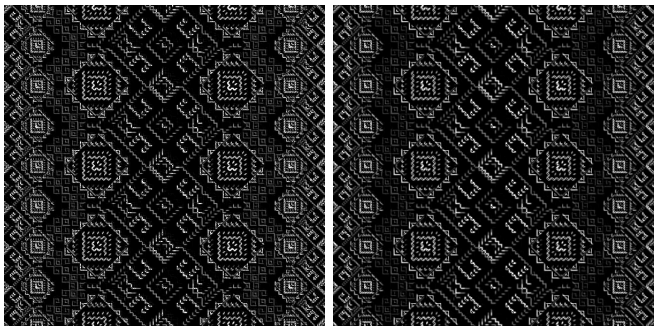

Fig. 1. Filtered Images with Prewitt Filters: Non-Blurred vs Blurred

- **Analysis 2: Different Blur Kernel Sizes** On the application of these tasks, each blurring implementation is combined with kernel sizes of 3, 5, and 7. Small kernel sizes apply very local blurring effects, achieves better results for de-noising. As the kernel sizes gets bigger, the blur effect strengthen, because in a sense, it averages bigger areas. As we can see in Figure 3, kernel-3 blurred image is more likely to original non-blurred filtering than kernel-7 filtered one. So, like in the choice of non-blurring vs blurring, it is a choice of goal. If the noise can be overcomed with small kernels, it is enough to apply

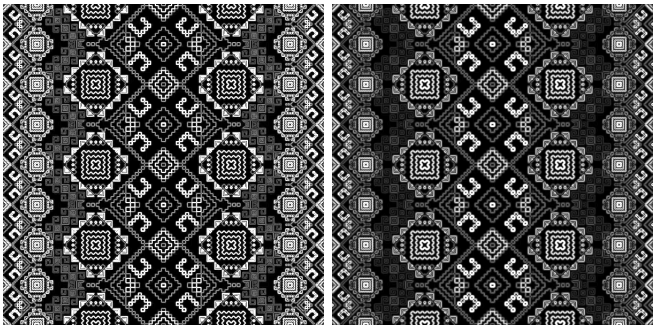

Fig. 2. Filtered Images with Sobel Filters: Non-Blurred vs Blurred

it because applying bigger kernels amy result in loss of details. However, we may look for the sudden changes which are resistant to be degraded on blurring. So, it can be more convenient to apply bigger kernel blurring to achieve those changes. It also can be achieved with different type of kernels, which changes the focus of kernels to achieve higher or lower averaging, which is out of scope of this report.

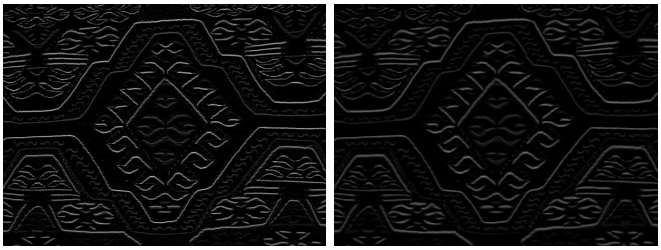

Fig. 3. Filtered Images with Robert: Kernel 3 vs Kernel 7

- **Analysis 3: Grayscale vs Binary Channel** This analysis is about binarization of image. The most significant bit of binary representation of grayscale channel carries the most important information, generally. This is a great way to suppress noises, and also facilitate to compress image. In edge detection, if the interested changes are significantly lighter or darker than the other areas, using bits of its channel is very convenient. Examples are shown in Figure 4. While grayscale image holds some of the little details and shapes; msb channel degraded them and even deleted some of them because those little and less lightened shapes are represented without usage of most signifant bit(msb). This is good when we want to find very representative part of the changes. However, taking only the msb results in taking the very light areas and keeping rest of them as dark. So if the contrast is low and values are accumulated between lets say 4th and 3rd bit of the 8 bit representation, we will get a noisy image with msb representation. Also, if the image has very high noises, those areas will appear as if the part of the image, which fails finding useful information. Of course those effects can be balanced with combination of efficient blurring and contrast strecthing.

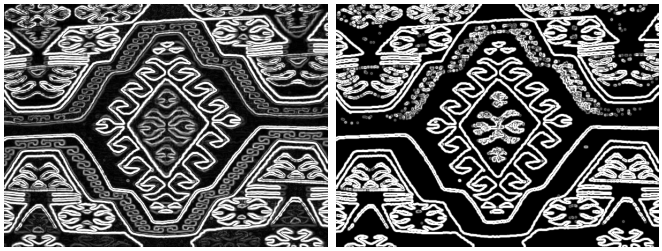

Fig. 4. Filtered Images with Robert: Kernel 3 vs Kernel 7

- **Analysis 4: Different Edge Detectors** Edge detection is the process of finding difference of the channel values with in a small window. Different edge detection algorithms provide unique advantages and trade-offs. The three edge detectors used in this analysis—Sobel, Prewitt, and Roberts—show varying sensitivity and emphasis on horizontal, vertical, and diagonal edges.

  Sobel Edge Detector:

  The Sobel filter emphasizes gradients along horizontal and vertical directions. Due to its larger kernel size compared to Roberts, it smooths out noise to some extent, making it particularly effective for images with minor noise. In Figure 5, the Sobel filter outputs well-defined and continuous edges for both blurred image. This is especially visible in high-contrast regions, such as the borders of the squares and inner details of the rug patterns. However, Sobel is less sensitive to smaller features due to its averaging effect. This averaging effect is the same effect of Gaussian blurring however they can be used together in same pipeline.

  Roberts Edge Detector:

  The Roberts edge detector, being a 2x2 kernel, is the simplest and most localized filter. Its diagonal focus allows it to detect finer details but makes it highly sensitive to noise. In Figure 5, the application of Roberts on blurred image results in sharper but slightly fragmented edges. This behavior highlights its strength in capturing finer details, such as diagonal edges, but also exposes its limitation in processing noisy or highly detailed regions without additional smoothing.

  Prewitt Edge Detector:

  The Prewitt filter is similar to Sobel but uses uniform weights across its kernel, resulting in less gradient amplification. In Figure 5, the Prewitt filter captures edges effectively for smooth gradients while being less responsive to noise compared to Roberts. However, it is outperformed by Sobel in high-contrast regions due to the lack of gradient amplification. The results show that Prewitt strikes a balance between noise suppression and edge definition.

  Comparison Across Filters:

  The choice of edge detector should be guided by the goal of the analysis:

  – **Sobel** is well-suited for detecting prominent edges

in noisy or complex images.

– **Roberts** is ideal for detecting finer details in cleaner images or with sufficient pre-blurring, but not a state-of-art solution.
– **Prewitt** is a good intermediate option for general-purpose edge detection, especially when noise levels are moderate.

Fig. 5. Filtered Images: Comparison of Sobel, Roberts, and Prewitt Edge Detectors on Blurred Images

II. Question 2: Image Enhancement

A. *Image Enhancement Implementation Details*

Firstly each image was split into its blue, green, and red channels using cv2.split and each channel was saved as a separate grayscale image to observe the channel-specific information, e.g., b1_blue_channel.png.

Then we applied a Gaussian blur to each image using different kernel sizes (e.g., (21, 21) for b1). This filter smooths the image by reducing high-frequency noise (like random speckles or sharp edges). Also we applied a median blur to each image using different kernel sizes. This is particularly effective in removing salt-and-pepper noise while preserving edges. We saved the blurred versions of the images.

Fourier Transform was used to filter the images in the frequency domain. Each channel was individually processed. The 2D Fourier Transform (np.fft.fft2) converts the image from the spatial domain to the frequency domain, where each pixel represents a frequency component of the image. The frequency representation was shifted (np.fft.fftshift) to center the low frequencies. Three types of filters were created based on the distance of frequencies from the center:

Ideal Low-Pass Filter (ILP):

- Allows only low frequencies (smooth details) to pass and removes high frequencies (sharp edges, noise).
- Frequencies within a cutoff distance (50 pixels) were kept.

Band-Pass Filter (BP):

- Allows frequencies within a specific range (between 30 and 70) to pass while removing both very low and very high frequencies.
- Useful for preserving textures.

Band-Reject Filter (BR):

- Removes frequencies within a specific range (between 30 and 70), keeping both very low and very high frequencies.
- Often used to remove repetitive patterns or certain noise types.

After filtering, the inverse Fourier Transform (np.fft.ifft2) was applied to bring the filtered image back to the spatial domain. Then we normalized the outputs to a [0–255] range so they could be saved as standard images.

After applying the filters to each channel (blue, green, red), the filtered channels were merged back into a single color image using cv2.merge. For each image (b1, b2, b3), the results of the three filters were saved.

B. *Discussion on Image Enhancement Methods*

*1) **Selection of parameters**:*

In order to select the parameters of the Gaussian and median filters, we simply try different filter sizes and investigate the outputs. Making the filters too small caused the lack of noise removal; on the other hand, making the filters too large resulted in extremely blurry images.

Also for the Fourier transform filters, same rule is valid.

*2) **Effects of each filter and their dependency on the image and the parameters**:*

Ideal Low-Pass Filter smooths the image by removing high-frequency components (fine details and noise). It enhances smooth areas while blurring sharp edges.
Images with significant fine details or high-frequency noise will lose sharpness or texture. The cutoff frequency determines the extent of smoothing. A lower cutoff leads to heavier blurring, while a higher cutoff preserves more detail and less effective to remove noises.

Band-Pass Filter retains only the frequencies within a specified range, emphasizing textures or details at certain scales.

This filter is most effective on images with noise or patterns concentrated in the chosen frequency range. The lower and upper cutoff frequencies define the preserved range. A narrow range isolates finer patterns, while a broader range includes coarser details.

Band-Reject Filter removes specific frequency ranges while keeping both low- and high-frequency components. It can suppress repetitive patterns, such as grid-like noise.

Best for images where noise or unwanted details are concentrated within a specific frequency band. The rejection frequency range is critical. Wider bands remove more noise but may also erase useful image features.

*3) **Can we remove the noises and our filter suggestion**:*

We think we have successfully removed the noise from the images.

Fig. 6. Noiseless images

However, there might be better solutions. A more effective approach might involve analyzing the images to identify the specific frequencies of the noise and then designing a frequency filter that removes those noise in those frequencies. In image processing best solutions are generally not generic but image specific.

III. Question 3: Image Compression

A. Image Compression Implementation Details

We first defined the apply_compression function, which is responsible for image compression. This function takes an image file path, an output name prefix, and a list of compression percentages (n_values). We read each image in grayscale to simplify the processing.

Then we applied the Haar Wavelet Transform to the image, which divided it into four components:

- cA: the low-frequency component (approximation).
- cH, cV, cD: High-frequency components representing horizontal, vertical, and diagonal details.

These were the coefficients prepared for the steps of compression.

Discrete Cosine Transform was carried out on the image to get its frequency domain. This would pack all the energy of the image into the top-left corner of the matrix; this corresponds to low-frequency details.

For Haar Wavelet Compression, we have zeroed all the wavelet coefficients (cA, cH, cV, cD) below a certain threshold that retains only the most significant N%. We then reconstruct the image by taking the inverse of the Haar Wavelet Transform. We retain the top N% most significant values of the DCT matrix and set all others to zero. Finally, the inverse DCT is used to reconstruct the compressed image.

For each reconstructed image, we calculated the MSE for measuring how much information is lost during compression. In that sense, separate MSEs were computed for the Haar Wavelet and DCT methods.

We saved the reconstructed images from both methods with filenames including the level of compression applied. We recorded the results of compressions in a dictionary, including MSEs and the reconstructed images. This way, we could compare the results of the two methods for different levels of compression.

B. Mean Squared Error Results

To observe the information loss we can:

Normalize the MSE by the range of pixel values to get a percentage-like measure. For an 8-bit image with pixel values in [0,255][0,255]:

$$MSE_{normalized} = \frac{MSE}{255^2} \times 100 \tag{1}$$

This gives a percentage of how much the pixel intensity has deviated due to compression, or we can compute *Peak Signal-to-Noise Ratio (PSNR)* by:

$$PSNR = 10 \times log_{10}(\frac{255^2}{MSE}) \tag{2}$$

to describe image quality. The higher the PSNR; the better the image quality, and the smaller the information loss.

C. *Discussion on Findings*

*1) **Compressed Sizes**:* The compression implementation raises a question due to our implementation: We obtained correct Haar and DCT results, but the compression convenience stems from the abundance of zero values introduced by these operations. To achieve significant compression quality, we convert our image arrays to the JPEG format, with the compression effect being observable at the 90% compression quality.

N (%)	Size Haar (KB)	Size DCT (KB)
1	164.93	285.88
10	334.95	460.21
50	452.25	474.93

TABLE II
COMPARISON OF HAAR AND DCT COMPRESSION SIZES FOR **C1.JPG** (ORIGINAL SIZE: 491.10 KB, JPEG QUALITY: 90%).

As we can see from Table 1, 2, and 3:

Haar compression consistently demonstrates superior performance in reducing file size, particularly at lower retention percentages. This is because the Haar wavelet transform inherently localizes energy efficiently in the low-frequency components, which allows significant parts of the data to be discarded without losing much information. This property results in smaller file sizes and a more balanced compression effect across different images.

N (%)	Size Haar (KB)	Size DCT (KB)
1	357.65	1073.00
10	1122.48	1522.98
50	1315.55	1288.33

TABLE III
COMPARISON OF HAAR AND DCT COMPRESSION SIZES FOR **C2.JPG** (ORIGINAL SIZE: 1337.94 KB, JPEG QUALITY: 90%).

In contrast, the DCT-based compression shows a less consistent and often imbalanced compression effect. At 10% retention, the file size of DCT-compressed images sometimes exceed the original image size, as seen in Table 2. This behavior is primarily due to the interaction between the retained DCT coefficients and the JPEG compression process. When a moderate number of coefficients are retained, the resulting matrix may lack sufficient sparsity, leading to inefficiency in JPEG encoding. Additionally, the retained coefficients might include higher-frequency components that are less compressible, causing a larger file size.

N (%)	Size Haar (KB)	Size DCT (KB)
1	59.42	326.52
10	181.53	432.00
50	293.18	292.60

TABLE IV
COMPARISON OF HAAR AND DCT COMPRESSION SIZES FOR **C3.JPG** (ORIGINAL SIZE: 293.48 KB, JPEG QUALITY: 90%).

However, as the retention percentage increases to 50%, the file size of DCT-compressed images decreases, often becoming more comparable to Haar-compressed images. This suggests that retaining more coefficients creates a more coherent matrix structure, which the JPEG encoder can handle more effectively. Despite this improvement, DCT compression still struggles to achieve the same level of size reduction as Haar compression, especially at lower retention levels.

In summary, Haar compression achieves better size reduction and consistency across different retention levels due to its ability to localize energy effectively. DCT compression, while powerful for certain applications, demonstrates imbalanced results in this implementation due to sparsity inefficiencies and its interaction with the JPEG encoding process.

*2) **Loss of information**:* The loss of information caused by compression can be measured using Normalized Mean Squared Error (MSE) and Peak Signal-to-Noise Ratio (PSNR). We can also judge it by looking at the quality of the reconstructed images.

Before comparisons, it is important to understand how PSNR values are interpreted. PSNR values below 20 dB are considered poor and indicate noticeable distortion. Values between 20–30 dB are moderate, showing some visible differences, while values above 30 dB are generally good, representing high-quality images with minimal visible loss.

As we can see from Tables 4, 5, and 6:

Haar compression generally has higher MSE values than DCT compression, especially at lower percentages ($N = 1\%$). This means Haar compression changes the pixel values more compared to DCT. However, the visual results from Haar compression at $N = 1\%$ and $N = 10\%$ reveal significant distortions, as we can see on Figure 6 and 7. At $N = 1\%$, the images are heavily degraded, with most recognizable features lost, making them visually unacceptable. At $N = 10\%$, while slightly better, the distortion is still quite noticeable, showing that Haar compression struggles to retain sufficient information at such low retention levels.

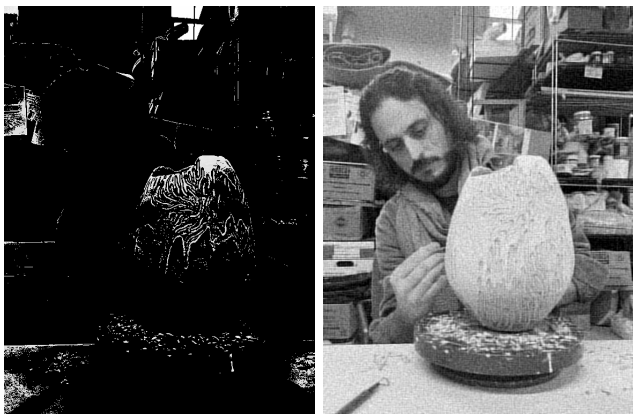

Fig. 7. HWT vs DCT results w N=1 for **c1.jpg**

DCT compression, on the other hand, shows lower MSE values, meaning it keeps more accurate pixel information. The PSNR values for DCT are also higher, which indicates better image quality. For example, at $N = 10\%$, DCT achieves a PSNR of around 40 dB, showing that the image quality is very

good and visually acceptable even at moderate compression levels. Even at $N = 1\%$, DCT achieves very acceptable quality as can be seen on Figure 8.

N (%)	Normalized MSE Haar (%)	Normalized MSE DCT (%)	PSNR Haar (dB)	PSNR DCT (dB)
1	30.12	0.50	5.21	23.02
10	8.94	0.12	10.49	29.25
50	0.0024	0.0073	46.17	41.38

TABLE V
COMPARISON HWT AND DCT COMPRESION MSE AND PSNR VALUES FOR `c1.jpg`.

At higher percentages ($N = 50\%$), both Haar and DCT have very small MSE values and high PSNR values (e.g., over 60 dB for Haar in `c2.jpg`), meaning almost no information is lost. The reconstructed images at this level are nearly identical to the original ones.

When looking at the images, Haar compression introduces extreme degradation at lower percentages ($N = 1\%$) and fails to retain sufficient detail. At higher percentages ($N = 50\%$), it performs well and produces high-quality reconstructions. DCT compression, in contrast, maintains better visual fidelity even at lower retention levels but may introduce small blocky patterns in some areas.

Fig. 8. HWT vs DCT results w N=10 for **c3.jpg**

N (%)	Normalized MSE Haar (%)	Normalized MSE DCT (%)	PSNR Haar (dB)	PSNR DCT (dB)
1	23.84	0.054	6.23	32.66
10	10.49	0.0084	9.79	40.75
50	0.000081	0.00041	60.86	53.84

TABLE VI
COMPARISON HWT AND DCT COMPRESION MSE AND PSNR VALUES FOR `c2.jpg`.

In summary, DCT compression keeps the pixel details more accurately, as shown by its lower MSE and higher PSNR. Haar compression, while it reduces file sizes effectively, introduces significant distortion at very low retention levels. The choice between Haar and DCT depends on the acceptable level of distortion and the compression goals. If retaining image quality is critical, DCT is the better option; if maximum size reduction is needed, Haar can be considered, but only at higher retention levels.

N (%)	Normalized MSE Haar (%)	Normalized MSE DCT (%)	PSNR Haar (dB)	PSNR DCT (dB)
1	26.23	0.068	5.81	31.65
10	11.68	0.012	9.32	39.35
50	0.000017	0.00034	67.72	54.67

TABLE VII
COMPARISON HWT AND DCT COMPRESION MSE AND PSNR VALUES FOR `c3.jpg`.

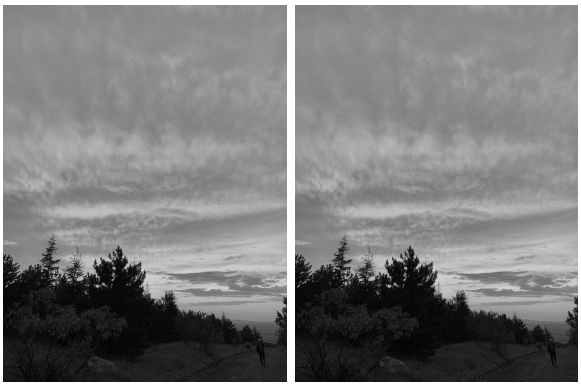

Fig. 9. Original vs DCT results w N=1 for **c2.jpg**

REFERENCES

Rafael C. Gonzales and Richard E. Woods, *Digital Image Processing*. New Jersey: Pearson Education Inc., 2008.

Shyam Singh Rajput, Nafis Uddin Khan, Amit Kumar Singh, and Karm Veer Arya, *Digital Image Enhancement and Reconstruction*. Elsevier Inc., 2023.

Peak Signal-to-Noise Ratio. Retrieved from: https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio