IMAGE WATERMARKING USING DWT

19ECE457 -
WAVELETS AND APPLICATIONS

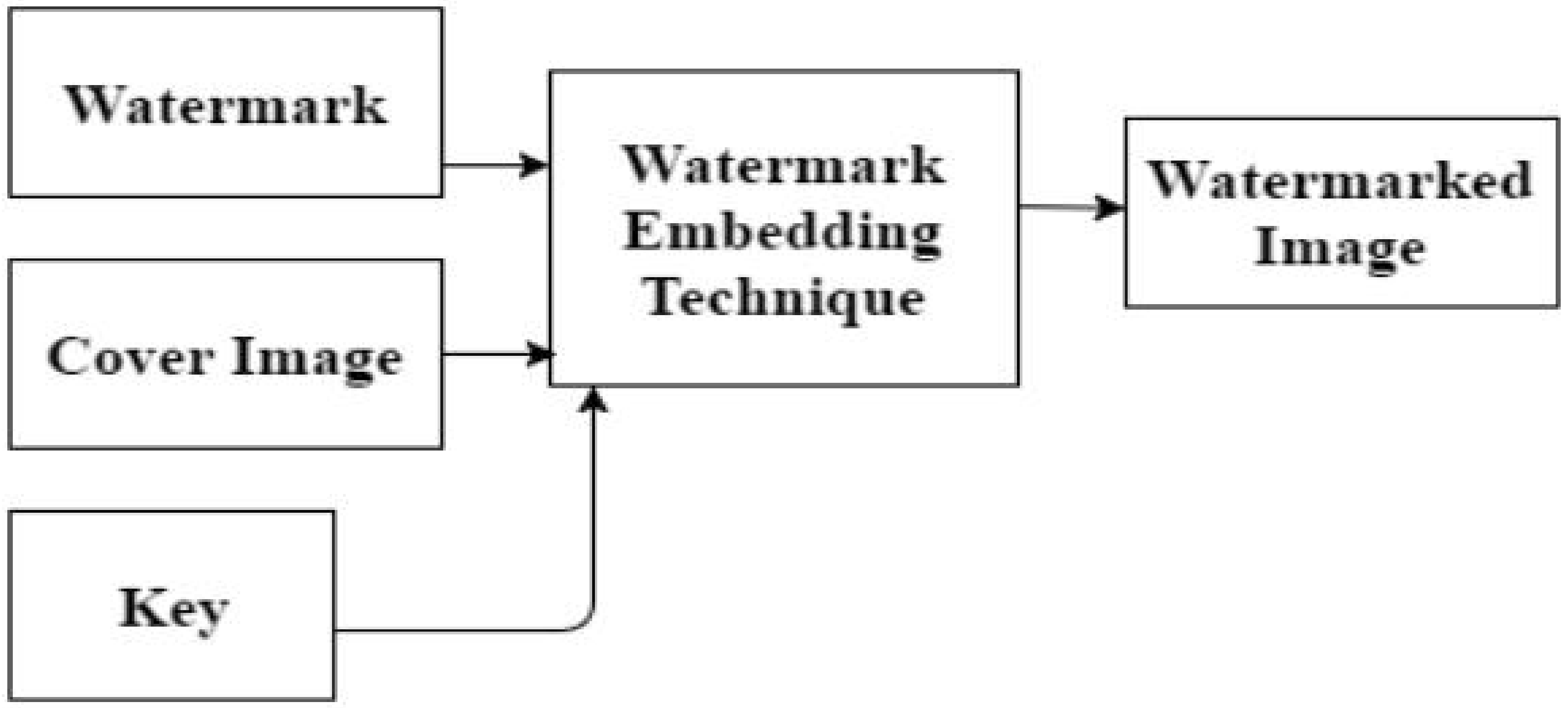
TEAM MEMBERS

NAME	ROLL NUMBER
CHAMARTHI RESHMANTH RAJU	CB.EN.U4ECE21109
GOTTUMUKKALA VENKATA MANOJ VARMA	CB.EN.U4ECE21116
MADHURITU SINHA ROY	CB.EN.U4ECE21126

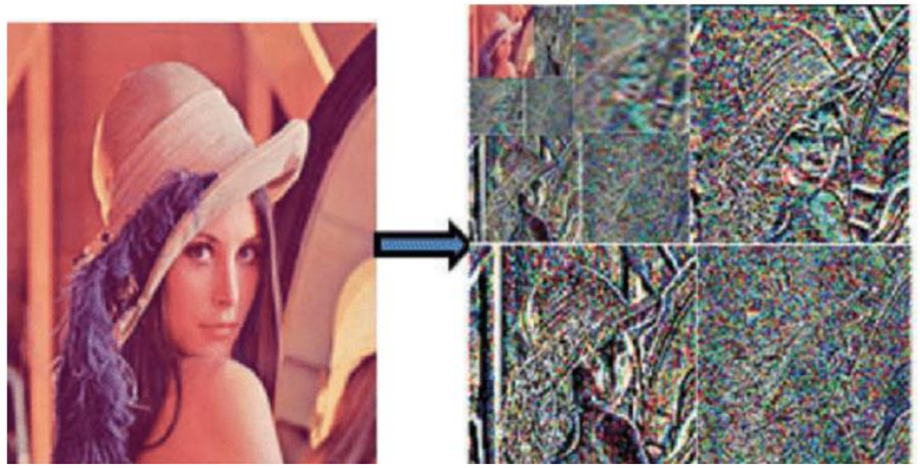
PROBLEM STATEMENT

- ❖The unauthorized distribution and piracy of digital content, such as images, audio, video, and documents, pose significant challenges for content creators and copyright owners. To protect their intellectual property rights and ensure the integrity of their work, there is a need for a reliable and robust digital watermarking method.
- ❖However, existing watermarking techniques often suffer from issues such as poor visibility, susceptibility to attacks, and vulnerability to alterations. These limitations undermine the effectiveness of watermarking as a means to identify original authors and owners, thereby hindering efforts to combat piracy and unlawful access.
- ❖This project aims to address these challenges by developing a digital watermarking method based on the combination of discrete wavelet transform (DWT).

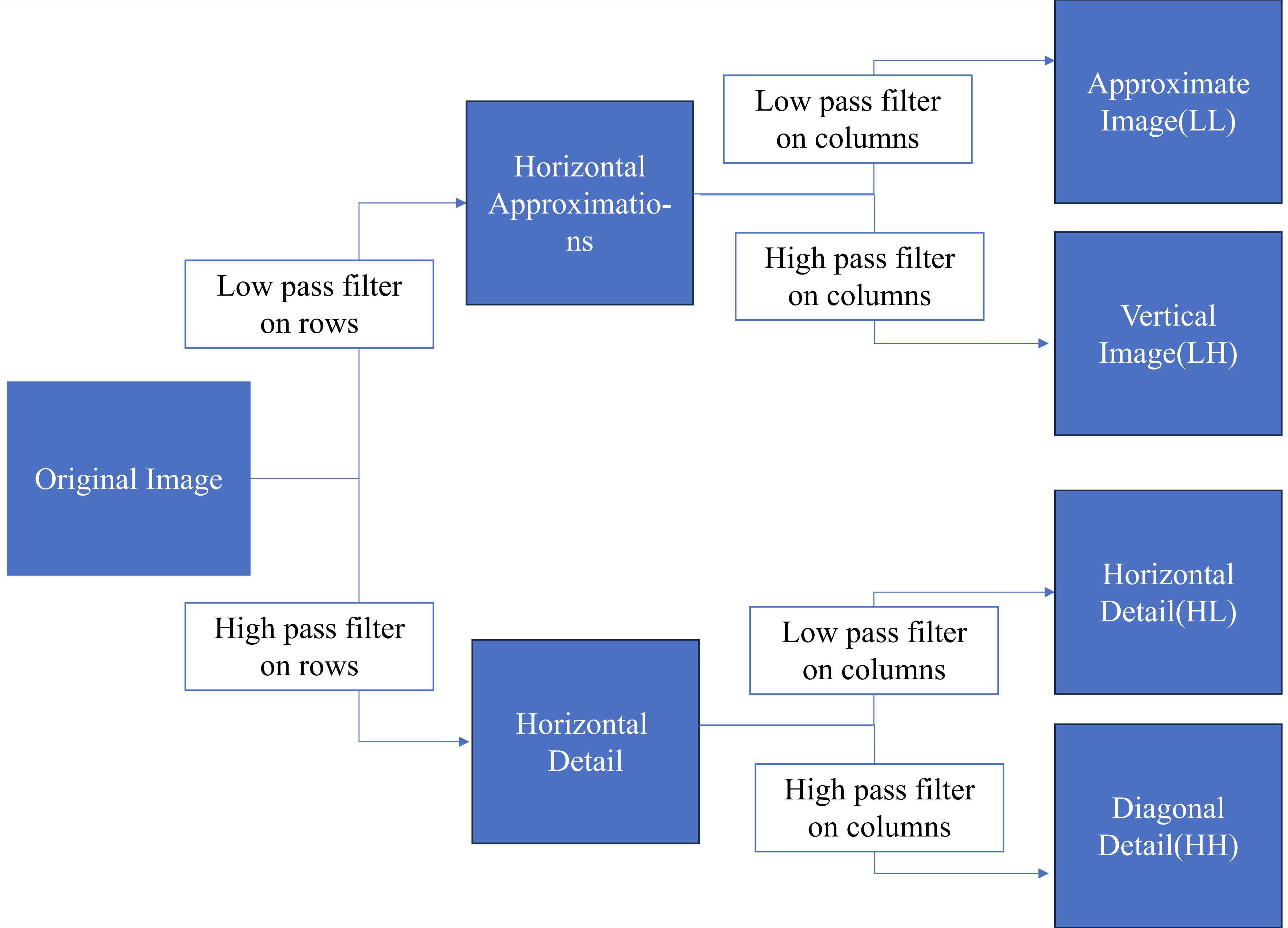
GENRAL MODEL OF DIGITAL WATERMARKING



WAVELET TRANSFORM-METHOD USED



- **Pyramidal decomposition of image:**
- Performing two separate one-dimensional transforms. First, the image is filtered along the x-dimension using low pass and high pass analysis filters and decimated by two.
- Low pass filtered coefficients are stored on the left part of the matrix and high pass filtered on the right.
- Then, it is followed by filtering the sub-image along the y-dimension and decimated by two. Finally, we have split the image into four bands denoted by LL, HL, LH and HH after one-level decomposition.


Original Image
Low pass filter on rows
High pass filter on rows
Horizontal Approximatio-ns
Horizontal Detail
Low pass filter on columns
High pass filter on columns
Low pass filter on columns
High pass filter on columns
Approximate Image(LL)
Vertical Image(LH)
Horizontal Detail(HL)
Diagonal Detail(HH)

ALGORITHM

- **Preprocessing:**
  - Load the original image and watermark image.
  - Convert both images to RGB mode to ensure they have three color channels (R, G, and B).
  - Resize the watermark image to match the dimensions of the original image (if not already the same size).
- **Initialize Parameters:**
  - Set the watermark strength parameter (alpha), which controls the visibility of the watermark.
- **Channel-wise Processing:**
  - For each color channel (R, G, B):
  - Extract the channel data from the original image and the watermark image.
  - Perform two level Discrete Wavelet Transform (DWT) on both the original image channel and the watermark image channel, decomposing each into:
    - Low-frequency component (LL).
    - High-frequency components (LH, HL, HH).
  - Embed the watermark by modifying the low-frequency (LL) component of the original image channel:
    - LL_embedded = LL_original + alpha * LL_watermark.
  - Reconstruct the channel using the Inverse Discrete Wavelet Transform (IDWT) with the embedded low-frequency component and the unmodified high-frequency components.

ALGORITHM

- **Recombine Channels:**
  - Combine the watermarked channels (R, G, and B) into a single RGB image.
- **Postprocessing:**
  - Clip the pixel values of the watermarked image to the valid range (0-255).
  - Convert the image data back to a format suitable for saving (e.g., uint8 format).
- **Output:**
  - Save the resulting watermarked image to the specified output path.

WAVELET USED

- **Simplicity**
  - The Haar wavelet is the simplest wavelet, involving only basic addition and subtraction operations.
- **Computational Efficiency**
  - Haar wavelet computations are fast, which is essential for processing large images or performing real-time applications.
- **Image Compression and Subband Focus**
  - Haar wavelets provide a straightforward way to decompose an image into low-frequency (LL) and high-frequency (LH, HL, HH) components.
  - The low-frequency subband (LL) contains the most important image features (e.g., general structure and luminance), making it an ideal target for embedding a watermark that should be perceptually invisible but robust.

WAVELET USED

- **Good for Watermarking**
  - The Haar wavelet is effective for embedding watermarks because:
  - The low-frequency subband is less sensitive to noise and distortions, ensuring the watermark remains robust during compression, resizing, or transmission.
  - High-frequency components remain unchanged, preserving edge details and texture of the image.
- **Wide Use in Image Processing**
  - Haar wavelets are widely used in applications such as image compression (e.g., JPEG 2000) and denoising, making them a natural choice for watermarking.
  - Their widespread use ensures compatibility with existing image processing frameworks and algorithms.

LIBRARIES USED

- **pywt (PyWavelets):**
    - Purpose: Enables the use of wavelet transforms, which decompose images into frequency components (subbands).
    - Usage:
    - DWT (pywt.dwt2): Splits an image into low-frequency (LL) and high-frequency (LH, HL, HH) components.
    - IDWT (pywt.idwt2): Reconstructs the image from these components after modification.
- **numpy:**
    - Purpose: Performs fast numerical operations on image arrays.
    - Usage:
    - Converts images into arrays for mathematical operations.
    - Adds the watermark to the low-frequency subband.
    - Clips values to the valid pixel range (0-255) for image reconstruction.
- **Pillow:**
    - Purpose: Handles image input, output, and basic processing.
    - Usage:
    - Opens images (Image.open) and converts them to compatible modes (RGB).
    - Resizes the watermark to match the original image size.
    - Converts processed arrays back into image format for saving (Image.fromarray).

CODE

```
def embed_watermark(image_path, watermark_path,
output_path):

    # Load the original image and watermark

    image = Image.open(image_path)

    watermark = Image.open(watermark_path)

    # Ensure the image and watermark are in RGB mode

    image = image.convert('RGB')

    watermark = watermark.convert('RGB')

    # Resize the watermark to match the image size

    watermark = watermark.resize(image.size,
Image.ANTIALIAS)

    # Convert the images to numpy arrays

    image_array = np.array(image)

    watermark_array = np.array(watermark)

    # Initialize an array for the watermarked image

    watermarked_array = np.zeros_like(image_array)
```

```
# Perform DWT and watermark embedding for each channel (R, G, B)
    alpha = 0.1  # Watermark strength parameter
    for channel in range(3):  # 0: R, 1: G, 2: B
        # Perform DWT on the image channel
        coeffs = pywt.dwt2(image_array[:, :, channel], 'haar')
        LL, (LH, HL, HH) = coeffs

        # Perform DWT on the watermark channel
        coeffs1 = pywt.dwt2(watermark_array[:, :, channel], 'haar')
        LL1, (LH1, HL1, HH1) = coeffs1

        # Embed the watermark in the LL subband
        watermark_embedded = LL + alpha * LL1

        # Reconstruct the watermarked channel
        coeffs_embedded = (watermark_embedded, (LH, HL, HH))
        watermarked_channel = pywt.idwt2(coeffs_embedded, 'haar')

        # Clip values to the valid range and convert to uint8
        watermarked_channel = np.clip(watermarked_channel, 0, 255).astype(np.uint8)

        # Store the watermarked channel
        watermarked_array[:, :, channel] = watermarked_channel

    # Convert the watermarked image array back to an image
    watermarked_image = Image.fromarray(watermarked_array, 'RGB')
    # Save the watermarked image
    watermarked_image.save(output_path)
```

Results

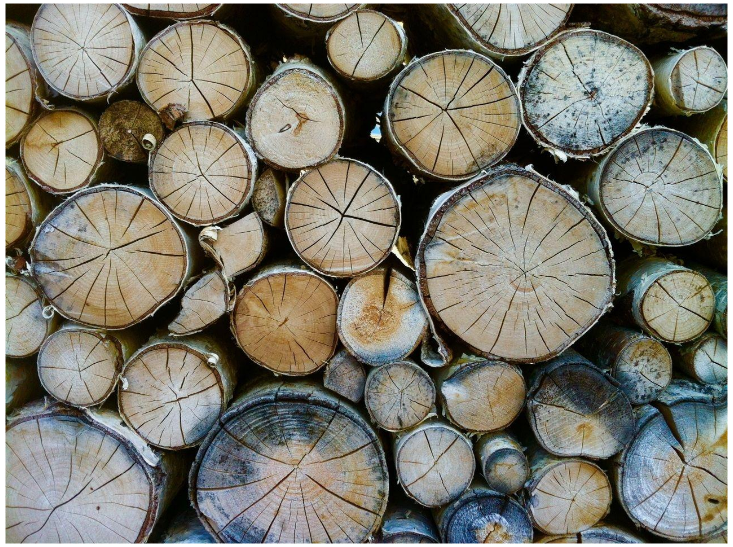


Fig. 1. Cover Image

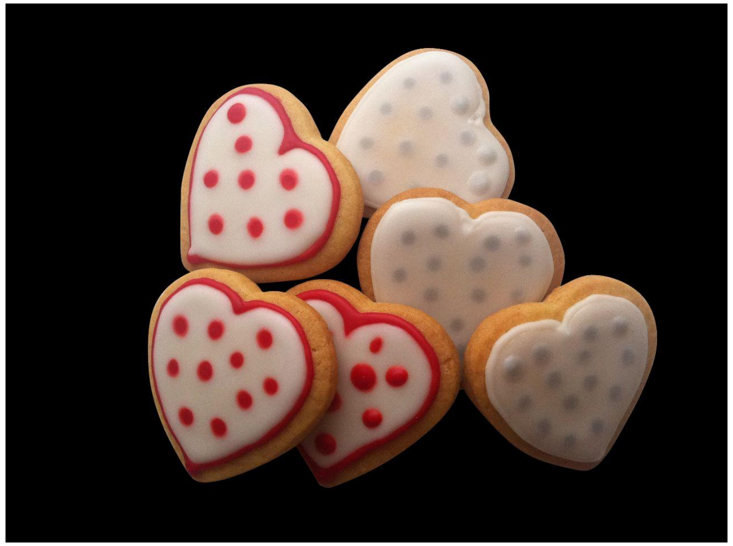


Fig.2. Image used as watermark

Results

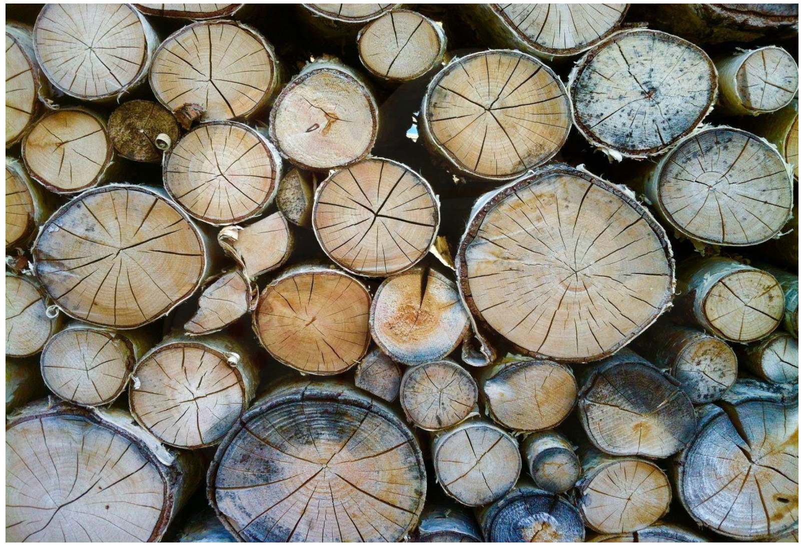


Fig. 3. Watermarked image when alpha=0.1

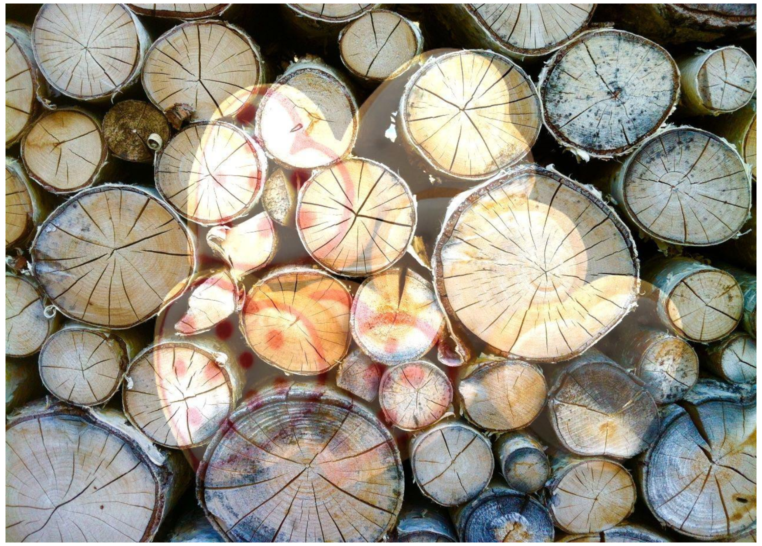


Fig. 4. Watermarked image when alpha=0.5

OTHER WATERMARKING TECHNIQUES

Least Significant Bit (LSB) Substitution

- **Key Idea:** Watermark bits are embedded directly into the least significant bits of the image's pixel values.

Unique Strengths:

- **Simple and Fast:** No complex transformations are required, making it one of the fastest watermarking techniques.
- **High Data Capacity:** Can embed large amounts of watermark data due to the direct modification of pixel bits.
- **Low Computational Cost:** No need for frequency transforms like DWT, DCT, or SVD.

DISCRETE FOURIER TRANSFORM (DFT)

- **Key Idea:** Watermark is embedded in the Fourier frequency domain, which is rotation, scaling, and translation invariant.

Unique Strengths:

- **Resistant to Geometric Attacks:** Unlike DCT or DWT, DFT is robust to rotation, scaling, and translation, making it perfect for environments where such transformations are likely.
- **Global Embedding:** The watermark is distributed globally across the entire image, making it difficult for an attacker to remove it without affecting the entire image.
- **High Robustness:** Strong resistance to most image processing attacks like cropping, filtering, and compression.

SINGULAR VALUE DECOMPOSITION (SVD)

Key Idea: Embeds the watermark by modifying the singular values of the image's matrix, ensuring robustness.

Unique Strengths:

- **High Robustness:** Singular values of an image are resistant to noise, compression, and filtering.
- **Low Perceptual Distortion:** Embedding in singular values maintains the image's perceptual quality since singular values are stable.
- **Flexible Hybridization:** Can be combined with DWT or DCT to enhance overall performance.