School of Engineering and Applied Science (SEAS), Ahmedabad University

ECE501: Digital Image Processing
WEEKLY REPORT-2

Date:11/10/2025

Project: **8. Digital Image Watermarking and Extraction Embed a watermark in an image and later extract or detect it**

Group Name: Pixels

Name	Enrollment Number
Bhavya Surati	AU2340215
Devang Parmar	AU2340217
Shubham Mehta	AU2340210
Pranel Agrawal	AU2340209

Main Problem Statement:

Digital Image Watermarking and Extraction Embed a watermark in an image and later extract or detect it

Our Learnings:

One of the ways to do watermarking is spatial domain watermarking. The spatial domain watermarking is the way to hide the watermark inside the image. The idea is to take our original image, and the watermarked image, here the watermarked image is the grayscale image and for spatial domain we are considering only the least significant bits (LSB), now for each bit, say we take a part of the original image, then we will reduce the bits of those parts with the corresponding watermarked image and in this way we create a watermark and we hide it so that it is not visible.
Our secret key was also used to randomly choose the pixels within the image in which the watermark bits will be carried. This makes the embedding a process that is more secure because it will be possible only to a person who has the same secret key to locate and retrieve the hidden signature

Work done till Week-1:

1. Selection and Knowledge of the Topic:

- This week, we applied digital image watermarking in Python as the LSB (Least Significant Bit) algorithm.

- We applied a cover image and a watermark image, changed the watermark to grayscale, resized and binarized the watermark and applied it to the LSBs of the cover image using a random selection secret key.
- We proceeded to remove the watermark with the same key and checked that this was the same.
- It is through this that we got to know how the process of LSB watermarking conceals data without any apparent distortion and how randomization makes it more secure.
- We also contrasted its performance with that of frequency-domain methods (DCT and DWT) learnt in the previous section.

Python

```python
import cv2
import numpy as np
import random
import matplotlib.pyplot as plt

# ---------------- CONFIG ----------------
cover_image_path = "cover.png"        # your cover image file in Colab
watermark_image_path = "watermark.png"  # your watermark image file
secret_key = 12345
# ----------------------------------------

# Load images
cover = cv2.imread(cover_image_path)
watermark = cv2.imread(watermark_image_path, cv2.IMREAD_GRAYSCALE)

if cover is None or watermark is None:
    raise Exception(" Could not read one or both images. Check file paths.")

# Resize watermark (¼ size of cover for safety)
wm_resized = cv2.resize(watermark, (cover.shape[1]//4, cover.shape[0]//4))
_, wm_bin = cv2.threshold(wm_resized, 127, 1, cv2.THRESH_BINARY)

# Flatten watermark bits
wm_bits = wm_bin.flatten().astype(np.uint8)
total_bits = len(wm_bits)

# Flatten cover and ensure it's uint8
flat_cover = cover.flatten().astype(np.uint8)

# Generate random pixel positions using secret key
random.seed(secret_key)
indices = random.sample(range(len(flat_cover)), total_bits)
```

```
# Embed watermark bits into LSBs safely
for i, bit in enumerate(wm_bits):
    pixel_value = int(flat_cover[indices[i]])
    pixel_value = (pixel_value & ~1) | int(bit)
    flat_cover[indices[i]] = np.uint8(pixel_value)

# Reshape back into image
watermarked = flat_cover.reshape(cover.shape)
print("Watermark embedded successfully!")

# ---------------- EXTRACT WATERMARK ----------------
flat_received = watermarked.flatten().astype(np.uint8)

# Extract bits using same secret key
extracted_bits = np.array([(flat_received[idx] & 1) for idx in indices],
dtype=np.uint8)
extracted_image = extracted_bits.reshape(wm_bin.shape) * 255

print("Watermark extracted successfully!")

# ---------------- SHOW RESULTS ----------------
plt.figure(figsize=(15,5))
plt.subplot(1,3,1)
plt.title("Original Cover Image")
plt.imshow(cv2.cvtColor(cover, cv2.COLOR_BGR2RGB))
plt.axis('off')

plt.subplot(1,3,2)
plt.title("Watermarked Image")
plt.imshow(cv2.cvtColor(watermarked, cv2.COLOR_BGR2RGB))
plt.axis('off')

plt.subplot(1,3,3)
plt.title("Extracted Watermark")
plt.imshow(extracted_image, cmap='gray')
plt.axis('off')

plt.show()
```

Original Cover Image

Watermarked Image

Extracted Watermark

2. Tool and Library Selection:

- Language chosen: Python
- Libraries to be used:
    - OpenCV - for image processing operations
    - NumPy - for mathematical computations
    - Matplotlib - for visualization
    - PyWavelets - for DWT-based watermark embedding

3. Plan for Next Week:

- Defining Scope and Goals:
    - We will either modify the code for spatial domain so there is a scope to modify to increase robustness or we will work on frequency domain for optimization.
- Implementation:
    - Also create datasets for template images and watermark images.
    - Implement DWT based watermarking for better robustness.
    - Implement verification methods using diverse image set for comprehensive validation
- Future Goals:
    - Comparing different methods using PSNR metrics.
    - Develop a complete watermarking system with GUI.
    - Modify the code for spatial domain so there is a scope to modify to increase robustness