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

**ECE501: Digital Image Processing**
**WEEKLY REPORT-4**

***Date:25/10/2025***

**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 |

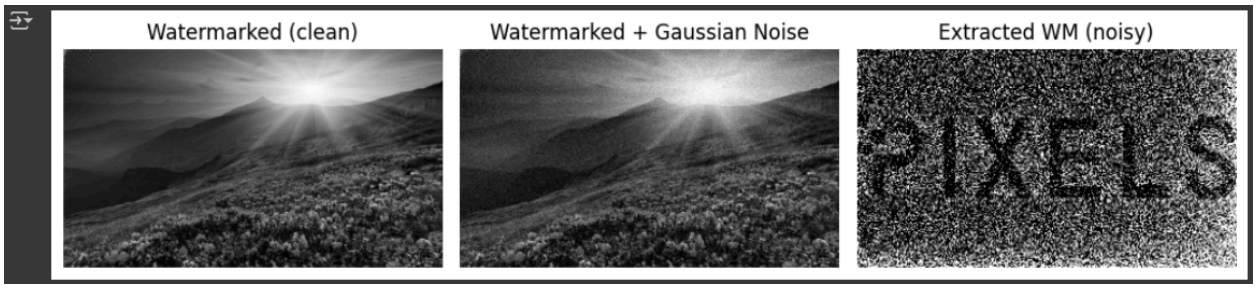
## Main Problem Statement:

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

## Our Learnings:

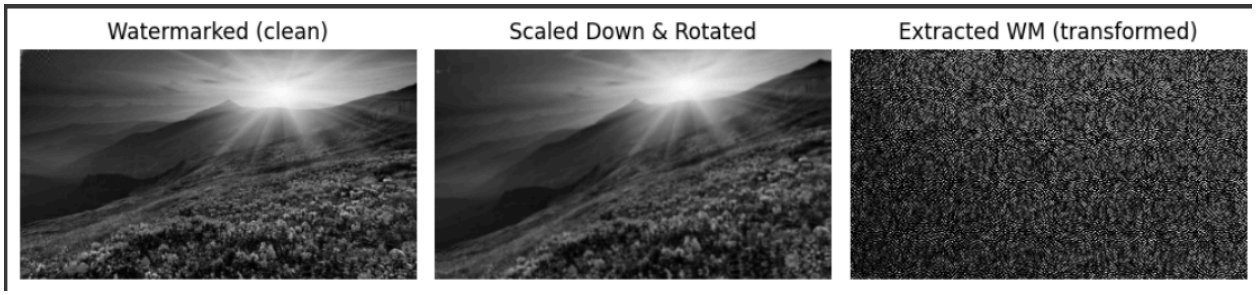
We already know what DWT and DCT methods do individually. The drawback in DWT includes the tricky process of choosing between the subbands and it can cause minor tuning and noise in the edges. In DCT, It is tougher to adapt the watermark strength in some regions. It is also much lesser robust. The hybrid DWT-DCT method provides with a fairly-good robustness as We have to apply the DCT method in a particular subband given from the DWT method. These are the results:

But when simulated for actual noisy changes that happen in real life (compression,resizing,rotation) the results are not up to the mark:

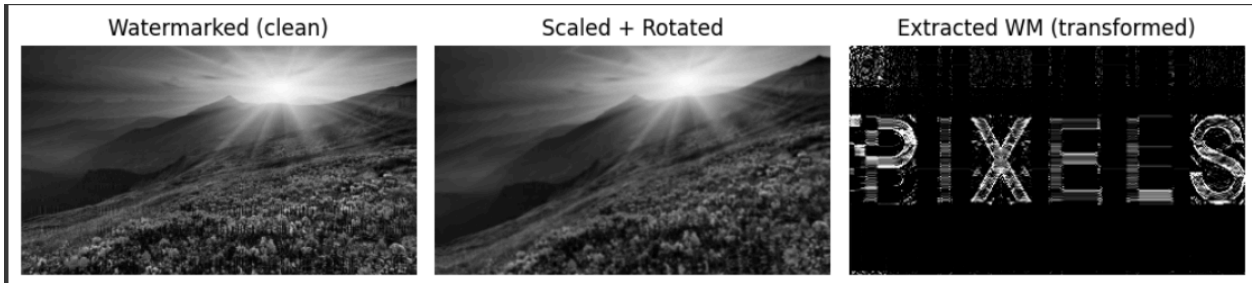



And these results for for higher range of alpha values too.

These results were just for added noise. For actual compression and resizing noise, the results are even worse.




Hence, we realised that we need to make changes. This is where SVD(Singular Value Decomposition) method comes in.
Adding noise to Hybrid dwt-dct model disrupts the coefficient alignment for the watermarked image. SVD isolates the not only the image but the particular frequency component of that particular subband into its components and embedding the image there. Making it much more stable for smaller rescaling and resizing changes.
The results:




These results are atleast better than the hybrid DCT-DWT Method.

## Work done in Week:

**1.Complete Hybrid DWT-DCT-SVD implementation and steps along the way:**

- This week, we implemented the hybrid dwt-dct method and implemented it for actual real-life changes and noise.
- We also introduced one more method to add into our model-SVD.
- We studied SVD through this paper.

- Implemented hybrid DWT-DCT-SVD in code to get good results.
- Compared both results from this new method and DWT-DCT method.
- We also did trials on the alpha-value coefficient that balances the robustness and visibility of watermark in the watermarked image.

Python

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

def embed_svd_watermark(cover_img, watermark_img, alpha=0.05):
    coeffs2 = pywt.dwt2(cover_img, 'haar')
    LL, (LH, HL, HH) = coeffs2
    HL = HL.astype(np.float32)

    dct_sub = cv2.dct(HL)
    U, S, Vt = np.linalg.svd(dct_sub, full_matrices=False)

    wm_float = cv2.resize(watermark_img, (HL.shape[1],
HL.shape[0])).astype(np.float32)
    Uw, Sw, Vwt = np.linalg.svd(wm_float, full_matrices=False)

    S_emb = S + alpha * Sw
    dct_embedded = np.dot(U, np.dot(np.diag(S_emb), Vt))

    HL_wm = cv2.idct(dct_embedded)
    watermarked_coeffs = (LL, (LH, HL_wm, HH))
    watermarked_img = pywt.idwt2(watermarked_coeffs, 'haar')
    watermarked_img = np.uint8(np.clip(watermarked_img, 0, 255))
    return watermarked_img, S, U, Vt, Uw, Vwt, Sw, HL.shape

def extract_svd_watermark(watermarked_img, S_orig, U, Vt, Uw, Vwt, alpha=0.05,
shape=None):
    coeffs2_w = pywt.dwt2(watermarked_img, 'haar')
    LL_w, (LH_w, HL_w, HH_w) = coeffs2_w
    if shape is not None:
        HL_w = HL_w[:shape[0], :shape[1]]
    dct_w = cv2.dct(HL_w.astype(np.float32))
    Uw_w, Sw_w, Vwt_w = np.linalg.svd(dct_w, full_matrices=False)
    Sw_extracted = (Sw_w - S_orig) / alpha
    wm_extracted = np.dot(Uw, np.dot(np.diag(Sw_extracted), Vwt))
    wm_extracted = np.uint8(np.clip(wm_extracted, 0, 255))
    return wm_extracted
```

```
# Load images
cover_img = cv2.imread("cover.png", cv2.IMREAD_GRAYSCALE)
watermark_img = cv2.imread("watermark.png", cv2.IMREAD_GRAYSCALE)

# Show original images
plt.figure(figsize=(8,4))
plt.subplot(1,2,1)
plt.title("Cover Image")
plt.imshow(cover_img, cmap='gray'); plt.axis('off')
plt.subplot(1,2,2)
plt.title("Watermark Image")
plt.imshow(watermark_img, cmap='gray'); plt.axis('off')
plt.tight_layout()
plt.show()
```

This code is the SVD implementation on watermarked image from DWT-DCT method.

### 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 and DCT-based watermark embedding

## 3. Plan for Next Week:

- Better models:
    - We will try to bring in more metrics and choose better values for coefficients to improve on the results.
- Implementation:
    - Apply the chosen Kaggle dataset on the final chosen model.
    - Implement verification methods like SNR to compare the extracted and original watermarks for accuracy.

- Future Goals:
    - Comparing different methods using PSNR metrics.
    - Develop a complete watermarking system with GUI.
    - Modify the code so there is a scope to modify to increase robustness

## 4. References:


- *Large-scale Common Watermark Dataset*. (2022, June 20). Kaggle. https://www.kaggle.com/datasets/kamino/largescale-common-watermark-dataset
- Akshay. (n.d.). *View of hybrid digital watermarking using DWT, DCT, and SVD for robust image protection*. https://theaspd.com/index.php/ijes/article/view/3065/2340