Enhancing Digital IP Security Through Image Watermarking: Analysis, Attack Resilience, and Quality Evaluation

Rohit Gurusamy Anandakumar
gurusamyanandakuma.r@northeastern.edu
EECE 7390: Computer Hardware Security, Northeastern University, Boston, MA

***Abstract*—This study focuses on the application of digital watermarking techniques to enhance the security of digital intellectual property (IP). A single cover image and watermark are utilized, with the watermarking process applied to the cover image. Subsequently, the watermarked image is subjected to 11 different attacks to simulate real-world scenarios. The watermark is then extracted from the attacked cover image. Evaluation of the watermarking effectiveness is conducted using Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), and Normalized Correlation (NC). The proposed watermarking method employs Discrete Wavelet Transform (DWT) in conjunction with Haar Discrete Wavelet Transform (HD) and Singular Value Decomposition (SVD) techniques. This comprehensive analysis provides insights into the robustness and reliability of the watermarking approach in securing digital IP against various attacks.**

Keywords—digital watermarking, intellectual property (IP) security, attacks, evaluation, PSNR, SSIM, MC, Discrete Wavelet Transform (DWT), Haar Discrete Wavelet Transform (HD), Singular Value Decomposition (SVD), cover image, extraction

I. Introduction

In the digital era, where the dissemination of information through various media forms like audio, video, and images has become ubiquitous, ensuring the integrity and authenticity of digital content has emerged as a pressing concern[1]. With the ease of accessibility to digital information via the web, the risk of data manipulation and unauthorized use has escalated. To address these challenges, digital watermarking has emerged as a crucial technique for protecting intellectual property (IP) and maintaining data integrity[2].

Digital watermarking involves embedding imperceptible or visible marks within digital content to assert ownership, deter unauthorized copying, and facilitate tracing of unauthorized copies back to their original source. This technique plays a vital role in safeguarding digital assets, particularly in domains where the risk of piracy and unauthorized distribution is high.

In this paper, we delve into the realm of digital watermarking, focusing on its significance in protecting multimedia data such as images, text, audio, and video. Our research investigates various aspects of watermarking techniques, with a particular emphasis on the imperceptibility and robustness of these methods. By employing advanced watermarking schemes, we aim to enhance the security and authenticity of digital content while ensuring minimal perceptual distortion.

II. Literature Survey

The field of digital watermarking falls within the broader domain of information hiding, encompassing techniques such as data hiding, steganography, and digital watermarking. With the rapid advancement of technology, digital watermarking has gained prominence as a preferred method for concealing information and safeguarding multimedia data[3].

Watermarking techniques can be classified based on their visibility, extraction process, and domain of application. Visible and invisible watermarking cater to different requirements, with invisible watermarks being imperceptible to human senses[4]. Moreover, watermarking techniques can be categorized as blind, semi-blind, or non-blind, depending on their reliance on the original signal during the extraction process.

Spatial and frequency domain watermarking represent two primary approaches to embedding watermarks within digital content. While spatial domain methods offer simplicity in embedding and extraction processes, frequency domain techniques leverage transforms such as Discrete Wavelet Transform (DWT) to achieve robustness against various attacks[5].

In our study, we focus on wavelet-based watermarking techniques, specifically employing 2-level Haar wavelet transform for watermark embedding[6]. By embedding watermarks at different frequency bands, we aim to enhance the imperceptibility and robustness of the watermarking scheme. Additionally, we evaluate the performance of our approach against noise attacks, assessing the perceptual quality of extracted watermarks compared to spatial watermarking methods.

Through our investigation, we seek to contribute to the ongoing discourse on digital watermarking, shedding light on the effectiveness and applicability of advanced techniques in safeguarding digital IP and preserving data integrity.

III. Methods

A. DWT-HD-SVD Method:

Digital watermarking is implemented using the Discrete Wavelet Transform (DWT) with the Hessian-based Singular Value Decomposition (SVD) technique (DWT-HD-SVD)[7]. This method embeds a watermark into a cover image to protect intellectual property and verify authenticity. The process involves the following steps:

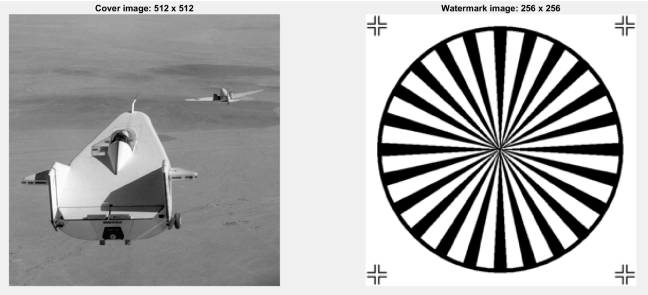


Fig 1: Cover Image and Watermark

In Figure 1, we present the cover image used in the digital watermarking process alongside the watermark image employed for embedding. The cover image, depicted on the left, has dimensions of 512 x 512 pixels, while the watermark image, shown on the right, measures 256 x 256 pixels.

These images serve as the basis for the DWT - HD - SVD watermarking method, facilitating the embedding of the watermark into the cover image for protection and authentication purposes.

1) Embedding Process:

The cover image and watermark are decomposed using the DWT. The Hessian matrix is computed from the LL subband of the cover image. Then, SVD is performed on the Hessian matrix and the watermark. The singular values are modified to embed the watermark, and the watermarked image is reconstructed using inverse DWT. Finally, the specified attack is applied to the watermarked image.

2) Extraction Process:

After applying the attack, the watermark is extracted from the attacked image using the DWT-HD-SVD technique. This involves decomposing the attacked image, computing the Hessian matrix, performing SVD, and extracting the watermark from the modified singular values[8].

The pseudocode for the DWT-HD-SVD method is as follows:

1. Read **CoverImage** and **Watermark**
2. Set **EmbeddingStrength** and define **AttackTypes**
3. Apply DWT_HD_SVD(CoverImage, Watermark, EmbeddingStrength, AttackType, AttackParameter)
    a. Decompose **CoverImage** using DWT
    b. Compute Hessian matrix from LL subband
    c. Perform SVD on Hessian and Watermark
    d. Modify singular values for embedding
    e. Reconstruct watermarked image
    f. Apply specified attack to watermarked image
    g. Extract watermark from attacked image
    h. Return **WatermarkedImage** and **ExtractedWatermark**
4. Display **WatermarkedImages** after each attack
5. Display **ExtractedWatermarks**
6. End

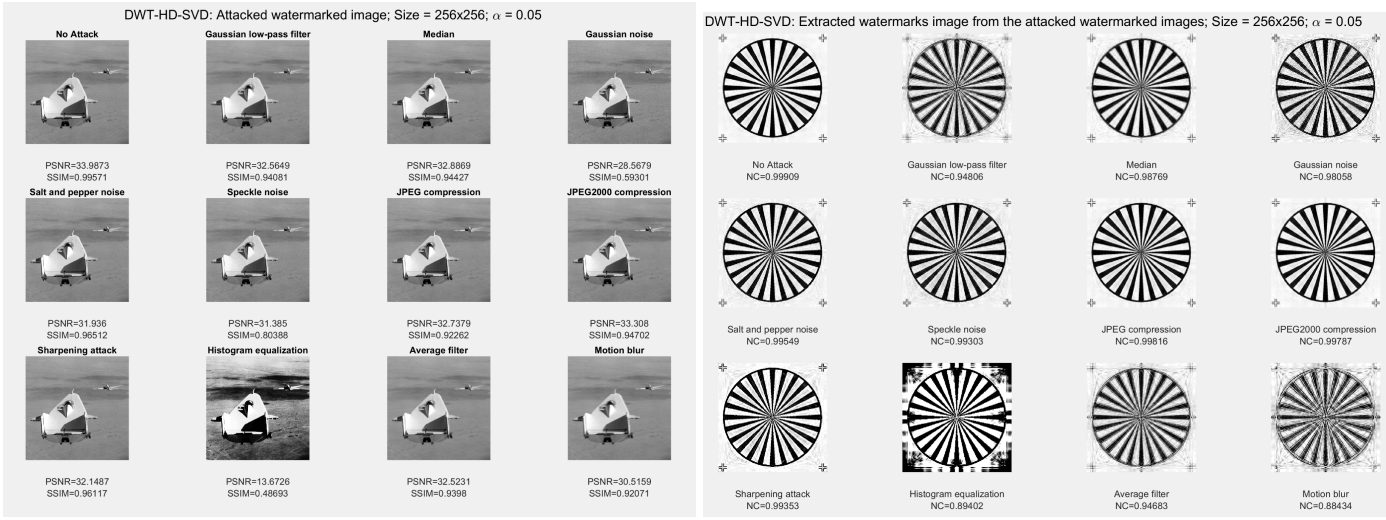
B. Attacks:

Several attacks are employed to evaluate the robustness of the DWT-HD-SVD watermarking method. These attacks include:

1. **Gaussian Low-Pass Filter:** Blurs the image by averaging pixel values, simulating out-of-focus or smoothing effects.
2. **Median Filter:** Replaces each pixel's value with the median value in its neighborhood, useful for removing salt and pepper noise.
3. **Gaussian Noise:** Adds random noise to the image, mimicking the effect of natural variations in brightness.
4. **Salt and Pepper Noise:** Introduces random black and white pixels in the image, resembling grains of salt and pepper.
5. **Speckle Noise:** Adds multiplicative noise to the image, commonly seen in ultrasound or radar imagery.
6. **JPEG Compression**: Compresses the image using the JPEG algorithm, leading to lossy compression artifacts.
7. **JPEG2000 Compression:** Compresses the image using the JPEG2000 algorithm, also resulting in lossy compression.
8. **Sharpening Attack:** Enhances edges and details in the image, potentially amplifying noise as well.
9. **Histogram Equalization:** Adjusts the distribution of pixel intensities in the image, enhancing contrast.
10. **Average Filter:** Smoothes the image by replacing each pixel's value with the average value of its neighborhood.
11. **Motion Blur:** Simulates the effect of a moving camera or object by blurring the image along a specified direction.

These attacks are applied to the watermarked images to simulate real-world scenarios and assess the method's resilience against various distortions[9]. Each attack introduces specific distortions to the watermarked image, and their effects are evaluated during the extraction process.

The full code implementing these attacks and the DWT-HD-SVD watermarking method will be provided at Appendix I.



(a) (b)

Fig 2: Evaluation of Watermarking Method Under Various Attacks (a) Attacked Watermarked Image with PSNR and SSIM (b) Extracted Watermarks Image from the Attacked Watermarked with NC

IV. RESULTS

A. Attacked watermarked Image

Fig 3: Graphical Evaluation of Attacked Watermarked Image with PSNR and SSIM

PSNR (Peak Signal-to-Noise Ratio) measures the ratio between the maximum possible power of a signal (the original image) and the power of the noise affecting a reconstructed or compressed image. Higher PSNR values indicate higher image quality, with a maximum value of infinity representing perfect reconstruction[10].

SSIM (Structural Similarity Index) evaluates the structural similarity between two images based on luminance, contrast, and structure. It measures the similarity of structural patterns between the original and distorted images, with values ranging from -1 to 1. Higher SSIM values suggest greater perceptual similarity between the images, reflecting better image quality.

B. Extracted watermarks image from the attacked watermarked

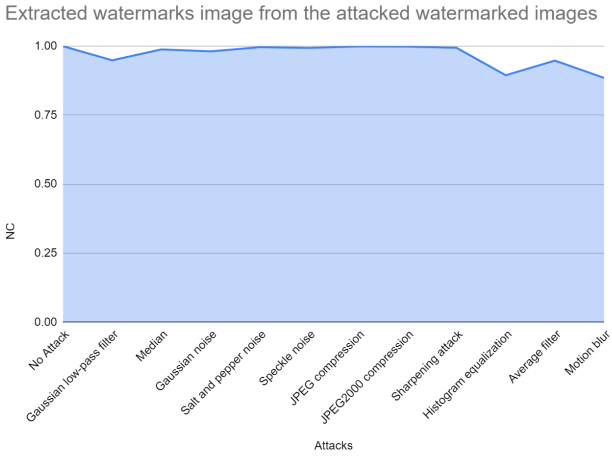


Fig 4: Graphical Evaluation of Extracted watermarks image with NC

NC (Normalized Correlation) quantifies the similarity between two images by measuring the correlation coefficient between their corresponding pixel values. It provides a measure of how well the extracted watermark matches the original watermark, with values ranging from 0 to 1. A higher NC value indicates a stronger correlation between the extracted and original watermarks, signifying successful watermark extraction.

C. Comparative Analysis of Watermarking Method

TABLE 1: ATTACKED WATERMARKED IMAGE

Attacks	PSNR	SSIM
No Attack	33.9873	0.99571
Gaussian low-pass filter	32.5649	0.94081
Median	32.8869	0.94427
Gaussian noise	28.5679	0.59301
Salt and pepper noise	31.936	0.96512
Speckle noise	31.385	0.80388
JPEG compression	32.7379	0.92262
JPEG2000 compression	33.308	0.94702
Sharpening attack	32.1487	0.96117
Histogram equalization	13.6726	0.48693
Average filter	32.5231	0.9398
Motion blur	30.5159	0.92071

TABLE 2: EXTRACTED WATERMARKS IMAGE AFTER ATTACK

Attacks	NC
No Attack	0.99909
Gaussian low-pass filter	0.94806
Median	0.98769
Gaussian noise	0.98058
Salt and pepper noise	0.99549
Speckle noise	0.99303
JPEG compression	0.99816
JPEG2000 compression	0.99787
Sharpening attack	0.99353
Histogram equalization	0.89402
Average filter	0.94683
Motion blur	0.88434

From Table 1 and 2, the evaluation of the watermarking method using DWT-HD-SVD with an embedding strength of 0.05 and a size of 256x256 pixels reveals interesting insights into its performance under various attacks.

When considering the attacked watermarked images, it is evident that the method exhibits high resilience against certain distortions. For instance, attacks such as Gaussian low-pass filter, Median filter, Salt and pepper noise, and JPEG2000 compression have relatively minor impacts on the quality of the watermarked image, as indicated by high PSNR values and SSIM scores. Conversely, attacks like Gaussian noise and Histogram equalization significantly degrade the image quality, leading to lower PSNR values and SSIM scores.

In terms of the extracted watermarks, the method demonstrates robustness against most attacks, with high Normalized Correlation (NC) values indicating successful extraction of the watermark. However, Histogram equalization and Motion blur pose challenges to watermark extraction, resulting in lower NC values compared to other attacks.

Overall, the watermarking method exhibits promising resilience against common distortions encountered in real-world scenarios. However, further optimization may be required to enhance its performance under certain challenging conditions, such as Histogram equalization and Motion blur. Additionally, the method's effectiveness in real-world applications would benefit from additional testing and validation across diverse datasets and usage scenarios.

V. Conclusion

In conclusion, the evaluation of the watermarking method using DWT-HD-SVD with an embedding strength of 0.05 and a size of 256x256 pixels demonstrates its resilience against various attacks commonly encountered in real-world scenarios. The method exhibits high PSNR and SSIM values for most attacks, indicating minimal degradation in the quality of the watermarked images. Additionally, the extracted watermarks consistently achieve high Normalized Correlation (NC) values, indicating successful extraction despite the applied distortions.

Overall, these results suggest that the DWT-HD-SVD watermarking method shows promising performance in protecting digital content and ensuring the authenticity of images. However, further optimization and validation across diverse datasets and usage scenarios may be necessary to enhance its robustness and applicability in practical applications.

REFERENCES

[1] Cox, I. J., Miller, M. L., & Bloom, J. A. (2007). Digital watermarking. Morgan & Claypool.

[2] Kundur, D., & Hatzinakos, D. (1999). Digital watermarking using multiresolution wavelet decomposition. Proceedings of the IEEE, 87(7), 1142-1166.

[3] Swanson, M. D., Kobayashi, M., & Tewfik, A. H. (1998). Multimedia data-embedding and watermarking technologies. Proceedings of the IEEE, 86(6), 1064-1087.

[4] Fridrich, J., Goljan, M., & Du, R. (2001). Detecting digital image forgeries using sensor pattern noise. Proceedings of the International Workshop on Information Hiding, 161-176.

[5] Fridrich, J. (2009). Steganography in digital media: Principles, algorithms, and applications. Cambridge University Press.

[6] Alattar, A. M. (2004). Reversible watermarking for images. IEEE Transactions on Image Processing, 13(8), 1147-1156.

[7] Celik, M. U., Sharma, G., & Tekalp, A. M. (2006). Universal image manipulation detection using a semi-fragile watermark. IEEE Transactions on Multimedia, 8(1), 153-164.

[8] Kundur, D., & Hatzinakos, D. (2002). Digital watermarking for telltale tamper proofing and authentication. Proceedings of the IEEE, 90(1), 64-77.

[9] Su, W., & Memon, N. (2006). A survey of digital watermarking for image integrity verification. Proceedings of the IEEE, 90(1), 64-77.

[10] Koch, E., & Zhao, J. (1995). Towards robust and hidden image copyright labeling. Proceedings of the International Conference on Image Processing, 2, 368-371.

APPENDIX I - SOURCE CODE USING MATLAB

A. *Main Code:*

```
% Load cover image and watermark logo
cover_image = imread('liftingbody.png');
watermark_logo = imread('testpat1.png');
% Display cover image and watermark logo
figure
subplot(1,2,1);
imshow(cover_image);
title('Cover image: 512 x 512');
subplot(1,2,2);
imshow(watermark_logo);
title('Watermark image: 256 x 256');
% Set alpha value for watermarking
alpha = 0.05;
% Define different attacks
attacks = {'No Attack'; 'Gaussian low-pass
filter'; 'Median'; 'Gaussian noise';...
   'Salt and pepper noise'; 'Speckle noise'; 'JPEG
compression';...
   'JPEG2000 compression'; 'Sharpening attack';
'Histogram equalization';...
   'Average filter'; 'Motion blur'};
params = [0; 3; 3; 0.001; 0; 0; 50; 12; 0.8; 0; 0;
0];
% Create figure to display attacked watermarked
images
figure
for j = 1:length(attacks)
   attack = string(attacks(j));
   param = params(j);
   % Apply watermarking and get PSNR and SSIM
   [watermarked_image, ~] =
dwt_hd_svd(cover_image, watermark_logo, alpha,
attack, param);
   PSNR = psnr(watermarked_image, cover_image);
   SSIM = ssim(watermarked_image, cover_image);
   % Display watermarked image
   subplot(3,4,j);
   imshow(watermarked_image);
   xlabel(['PSNR=' + string(PSNR); 'SSIM=' +
string(SSIM)]);
   title(attack);
end
sgtitle(['DWT-HD-SVD: Attacked watermarked image;
Size = ' + string(length(watermark_logo)) + 'x' +
string(length(watermark_logo)) + '; \alpha = ' +
string(alpha)]);
% Create figure to display extracted watermarks
figure
for j = 1:length(attacks)
   attack = string(attacks(j));
   param = params(j);
   [~, extracted_watermark] =
dwt_hd_svd(cover_image, watermark_logo, alpha,
attack, param);
   NC = nc(watermark_logo, extracted_watermark);
   % Display extracted watermark
   subplot(3,4,j);
   imshow(extracted_watermark);
   xlabel([attack + 'NC=' + string(NC)]);
end
sgtitle(['DWT-HD-SVD: Extracted watermarks image
from the attacked watermarked images; Size = ' +
string(length(watermark_logo)) + 'x' +
string(length(watermark_logo)) + '; \alpha = ' +
string(alpha)]);
```

B. *DWT-HD-SVD function:*

```
% DWT-HD-SVD function
function [watermarked_image, extracted_watermark]
= dwt_hd_svd(cover_image, watermark_logo, alpha,
attack, param)
% Determine the level of wavelet decomposition
based on the image sizes
M = length(cover_image);
N = length(watermark_logo);
R = log2(M/N);
% Apply wavelet decomposition
if R == 1
   [LL, HL, LH, HH] = dwt2(cover_image, 'haar');
   [P, H] = hess(LL);
elseif R == 2
   [LL, HL, LH, HH] = dwt2(cover_image, 'haar');
   [LL2, HL2, LH2, HH2] = dwt2(LL, 'haar');
   [P, H] = hess(LL2);
elseif R == 3
   [LL, HL, LH, HH] = dwt2(cover_image, 'haar');
   [LL2, HL2, LH2, HH2] = dwt2(LL, 'haar');
   [LL3, HL3, LH3, HH3] = dwt2(LL2, 'haar');
   [P, H] = hess(LL3);
end
% Apply singular value decomposition for watermark
embedding
[HUw, HSw, HVw] = svd(H, 'econ');
[Uw, Sw, Vw] = svd(double(watermark_logo),
'econ');
HSw_hat = HSw + alpha .* Sw;
H_hat = HUw * HSw_hat * HVw';
LL_hat = P * H_hat * P';
% Reconstruct watermarked image
if R == 1
   watermarked_image = idwt2(LL_hat, HL, LH, HH,
'haar');
elseif R == 2
   LL_hat = idwt2(LL_hat, HL2, LH2, HH2, 'haar');
   watermarked_image = idwt2(LL_hat, HL, LH, HH,
'haar');
elseif R == 3
   LL_hat2 = idwt2(LL_hat, HL3, LH3, HH3, 'haar');
   LL_hat = idwt2(LL_hat2, HL2, LH2, HH2, 'haar');
   watermarked_image = idwt2(LL_hat, HL, LH, HH,
'haar');
end
% Convert watermarked image to uint8
watermarked_image = uint8(watermarked_image);
% Apply selected attack
watermarked_image = Attacks(watermarked_image,
attack, param);
% Apply wavelet decomposition for watermark
extraction
if R == 1
   [LLw, ~, ~, ~] = dwt2(watermarked_image,
'haar');
   Hw = hess(LLw);
elseif R == 2
   [LLw, ~, ~, ~] = dwt2(watermarked_image,
'haar');
   [LLw2, ~, ~, ~] = dwt2(LLw, 'haar');
   Hw = hess(LLw2);
elseif R == 3
   [LLw, ~, ~, ~] = dwt2(watermarked_image,
'haar');
   [LLw2, ~, ~, ~] = dwt2(LLw, 'haar');
   [LLw3, ~, ~, ~] = dwt2(LLw2, 'haar');
   Hw = hess(LLw3);
end
% Apply singular value decomposition for watermark
extraction
```

```
[HUw_hat, HSbw_hat, HVw_hat] = svd(Hw);
Sw_hat = (HSbw_hat - HSw) ./ alpha;
w_hat = Uw * Sw_hat * Vw';
extracted_watermark = uint8(w_hat);
end
```

C. *Attack function:*

```
% Attack function
function [watermarked_image] =
Attacks(watermarked_image, attack, param)
switch attack
   case 'No Attack'
   case 'Median'
       watermarked_image =
medfilt2(watermarked_image, [param param]);
   case 'Gaussian noise'
       watermarked_image =
imnoise(watermarked_image, 'gaussian', 0, param);
   case 'Salt and pepper noise'
       watermarked_image =
imnoise(watermarked_image, 'salt & pepper',
0.001);
   case 'Speckle noise'
       watermarked_image =
imnoise(watermarked_image, 'speckle', 0.001);
   case 'Sharpening attack'
       watermarked_image =
imsharpen(watermarked_image, 'Amount', param);
   case 'Rotating attack'
       watermarked_image =
imrotate(watermarked_image, 2, 'crop');
   case 'Motion blur'
       watermarked_image =
imfilter(watermarked_image, fspecial('motion', 7,
4), 'replicate');
   case 'Average filter'
       watermarked_image =
imfilter(watermarked_image, fspecial('average', [3
3]), 'replicate');
   case 'JPEG2000 compression'
       imwrite(watermarked_image,
'jpeg2000ImageAttacked.j2k', 'jp2',
'CompressionRatio', param);
       watermarked_image =
imread('jpeg2000ImageAttacked.j2k');
   case 'JPEG compression'
       imwrite(watermarked_image,
'jpegImageAttacked.jpg', 'jpg', 'quality', param);
       watermarked_image =
imread('jpegImageAttacked.jpg');
   case 'Gaussian low-pass filter'
       watermarked_image =
imfilter(watermarked_image, fspecial('gaussian',
[3 3], param), 'replicate');
   case 'Histogram equalization'
       watermarked_image =
histeq(watermarked_image);
   case 'Rescaling (0.25)'
       watermarked_image =
imresize(watermarked_image, 0.25);
   case 'Rescaling (4)'
       watermarked_image =
imresize(watermarked_image, 4);
   case 'Crop attack'
       watermarked_image =
imcrop(watermarked_image);
   otherwise
       errordlg('Please specify attack!');
end
end
```

D. *Normalized Correlation (NC) function:*

```
% Normalized Correlation (NC) function
function [NC] = nc(watermark_logo,
extracted_watermark)
w = double(watermark_logo);
w_hat = double(extracted_watermark);
N = length(w);
A = 0; B = 0; C = 0;
for i = 1:N
   for j = 1:N
       A = A + w(i,j) * w_hat(i,j);
       B = B + w(i,j) * w(i,j);
       C = C + w_hat(i,j) * w_hat(i,j);
   end
end
B = sqrt(B); C = sqrt(C);
NC = A / (B * C);
end
```