SECURE DIGITAL CONTENT WITH VERIFICATION USING DFT-BASED WATERMARKING

Project Background

The rapid growth of digital content distribution has increased the need for robust security measures to protect against piracy, unauthorized copying, and tampering. Images are easily replicable and distributable, making them susceptible to unauthorized access, leakage, and malicious manipulation. In various industries such as journalism, entertainment, and advertising, the unauthorized leaking or modification of images can have profound consequences.. With the rise of social media and online platforms, individuals are increasingly vulnerable to the unauthorized sharing and manipulation of their images, highlighting the urgent need for robust image security measures. Therefore, it's imperative for industries and individuals alike to employ effective image protection techniques, such as watermarking to safeguard both professional integrity and personal privacy. , watermarking enhances content integrity and authenticity by providing a mechanism for verifying the originality and source of an image, which is particularly important in contexts where image manipulation or forgery is prevalent

Project Aim

This project aims to develop a secure method for protecting digital content and verifying its authenticity using DFT-based watermarking. By calculating Mean Squared Error (MSE), the system determines the watermark's presence, providing visual feedback on the watermarking process. Through user interaction, users can select an image for watermark embedding and verification, contributing to improved security and integrity of digital content.

Project Objectives

1. Implement a DFT-based watermarking algorithm for embedding watermarks into images.
2. Design a robust verification mechanism utilizing mean squared error (MSE) to accurately determine whether an image has been successfully watermarked or not.
3. Integrate watermarking and verification algorithms into a cohesive secure digital content management system.
4. Optimize the watermarking process to ensure a balance between imperceptibility and robustness, minimizing the visual distortion introduced by the embedded watermark while maintaining its detectability.

Questions related to the project:

Q1: How does DFT-based watermarking enhance the security of digital images?

A: DFT-based watermarking embeds imperceptible identifiers into digital images by altering their frequency domain representation. This technique ensures that the watermark is robust against common image processing operations and compression algorithms, thus enhancing digital content security.

Q2: Why Discrete Fourier Transform is used for watermarking?

A: DFT-based watermarking offers several advantages for image verification, including robustness, invisibility, and computational efficiency. By analyzing the frequency domain representation of watermarked images, verification algorithms can authenticate the presence of the embedded watermark without significantly degrading image quality or requiring extensive computational resources.

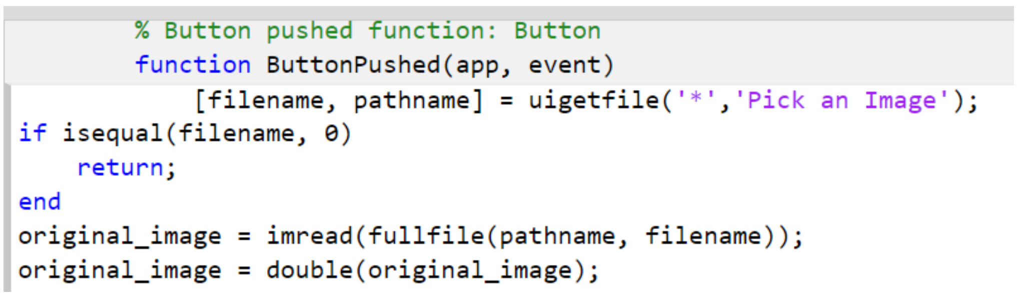
Q3: How does the system utilize mean squared error (MSE) for verification to determine whether an image has been successfully watermarked?

A:The system employs a threshold value within an "if" loop to compare the Mean Squared Error (MSE) of the watermarked image against a predefined threshold. If the calculated MSE falls below the threshold, the system outputs a message indicating that the image is not watermarked; otherwise, it confirms that the image has been watermarked.

Methodology

1. Image Loading and Pre-processing:

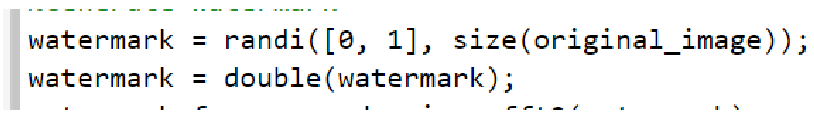
Firstly , we write a small code that allows the user to select an image file using a file dialog window which reads the selected image, converts it to double precision for further processing, and stores it in the variable original_image.



```
        % Button pushed function: Button
        function ButtonPushed(app, event)
            [filename, pathname] = uigetfile('*','Pick an Image');
if isequal(filename, 0)
    return;
end
original_image = imread(fullfile(pathname, filename));
original_image = double(original_image);
```

2. Watermark Generation:

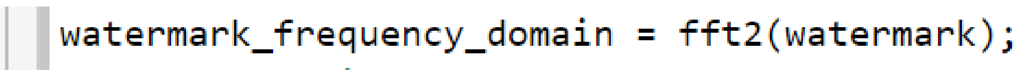
After reading the image we have generated a binary watermark of the same size as the original image.



```
watermark = randi([0, 1], size(original_image));
watermark = double(watermark);
```

3. Frequency Domain Conversion:

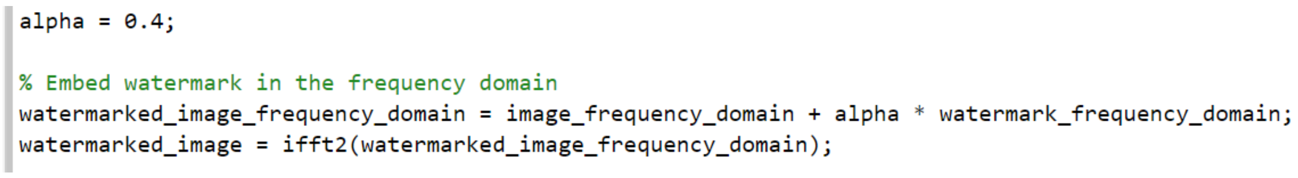
. Both the original image and the watermark are converted to the frequency domain using the 2D Fast Fourier Transform (FFT) function fft2().



```
watermark_frequency_domain = fft2(watermark);
```

4. Watermark Embedding:

After generating the watermark and converting the images in the frequency domain we embed the watermark into the frequency domain of the original image using an alpha blending factor (alpha). It adds the scaled watermark frequency domain to the original image frequency domain. Then, we apply the inverse FFT (ifft2()) to obtain the watermarked image in the spatial domain.



```
alpha = 0.4;

% Embed watermark in the frequency domain
watermarked_image_frequency_domain = image_frequency_domain + alpha * watermark_frequency_domain;
watermarked_image = ifft2(watermarked_image_frequency_domain);
```

5. Displaying Images and Frequency Domain Representation:

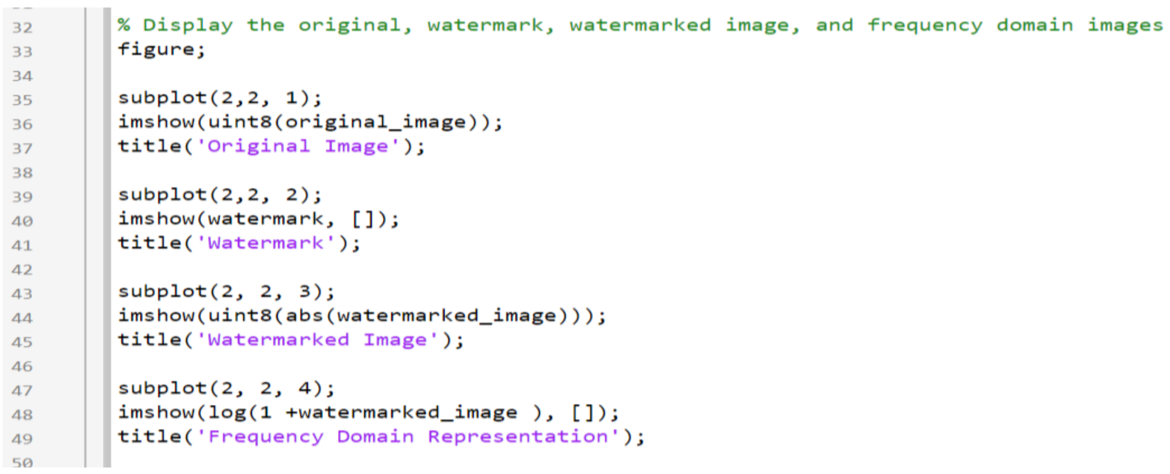
Then we display images to show the results:

Original Image: Displays the original image.

Watermark: Displays the generated binary watermark.

Watermarked Image: Displays the watermarked image.

Frequency Domain Representation: Displays the log-scaled magnitude of the watermarked image in the frequency domain.



```
% Display the original, watermark, watermarked image, and frequency domain images
figure;

subplot(2,2, 1);
imshow(uint8(original_image));
title('Original Image');

subplot(2,2, 2);
imshow(watermark, []);
title('Watermark');

subplot(2, 2, 3);
imshow(uint8(abs(watermarked_image)));
title('Watermarked Image');

subplot(2, 2, 4);
imshow(log(1 +watermarked_image ), []);
title('Frequency Domain Representation');
```

6. Watermark Presence Verification:

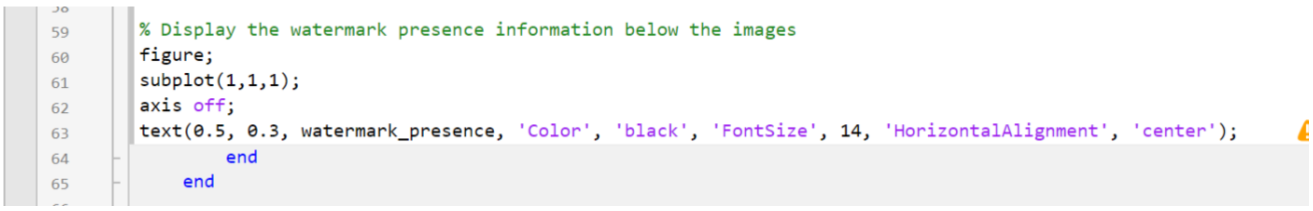
To verify whether the watermark is present or not we calculate the Mean Squared Error (MSE) between the original and watermarked images. If the MSE exceeds a threshold (1e-6), the watermark is considered present; otherwise, it is considered not present.

```matlab
% Check if watermark is present
mse = norm(original_image(:) - watermarked_image(:), 'fro')^2 / numel(original_image);
if mse > 1e-6
    watermark_presence = 'Watermark is present';
else
    watermark_presence = 'Watermark is not present';
end

% Display the watermark presence information below the images
```

7. Displaying Watermark Presence Information:

We create a new figure with no axes and add a text annotation indicating whether the watermark is present or not.



```matlab
% Display the watermark presence information below the images
figure;
subplot(1,1,1);
axis off;
text(0.5, 0.3, watermark_presence, 'Color', 'black', 'FontSize', 14, 'HorizontalAlignment', 'center');
        end
    end
```

Results

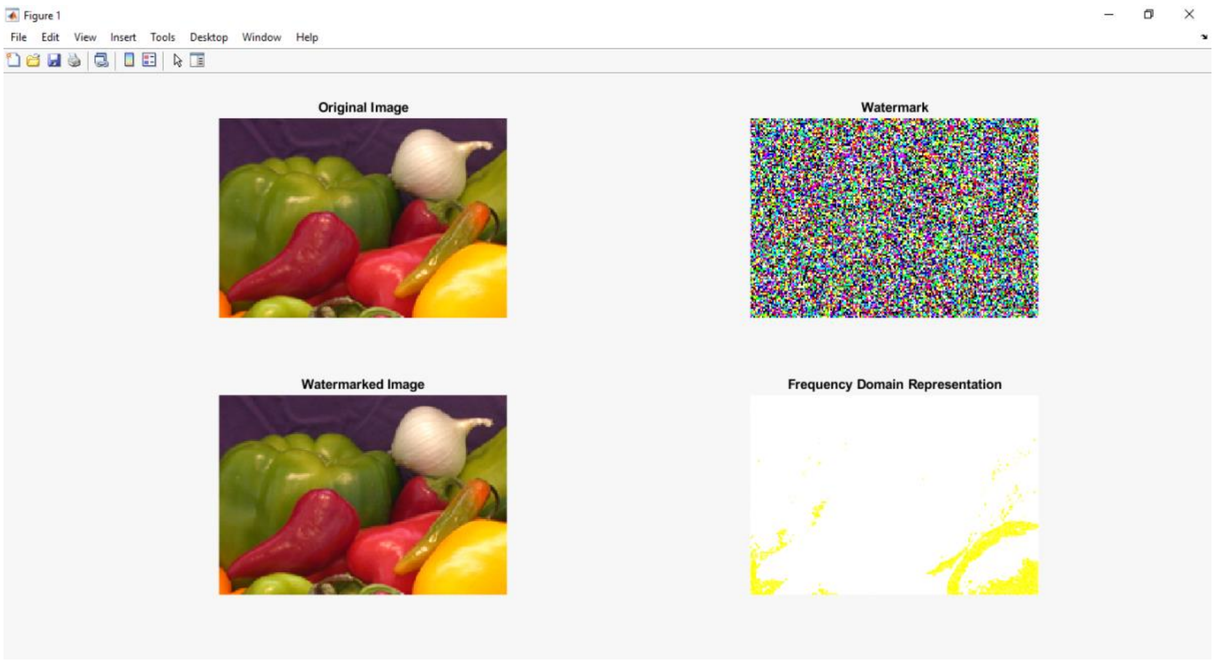
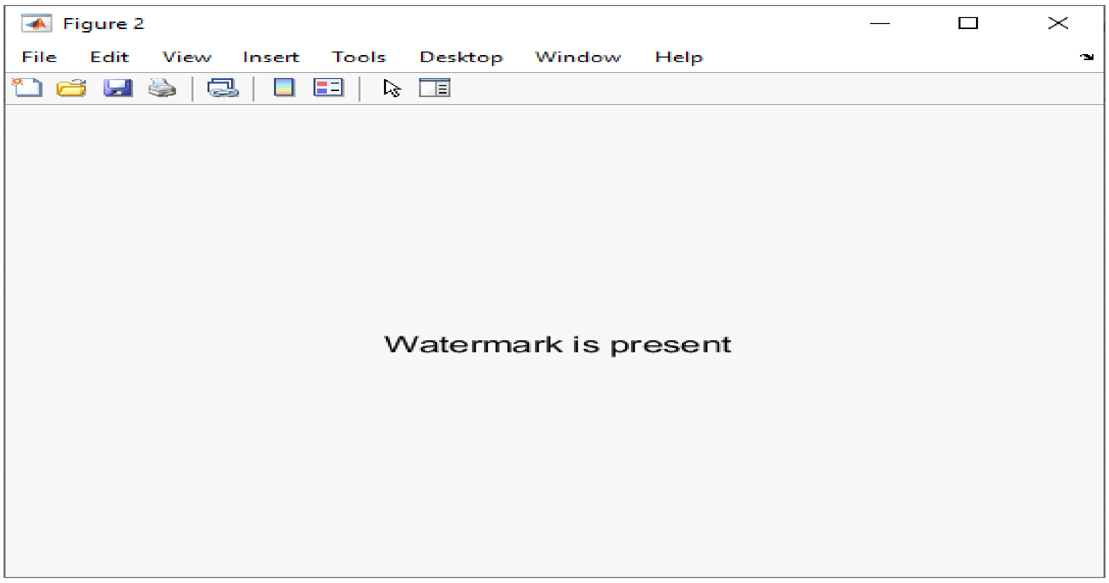


Figure 1
File Edit View Insert Tools Desktop Window Help
Original Image
Watermark
Watermarked Image
Frequency Domain Representation



Watermark is present

Future Work

In the future, application can be optimized by enhancing the security and effectiveness of our DFT-based image watermarking system. Techniques such as spread spectrum or error correction coding can be applied to enhance the security of the watermark. Additionally, we can implement the reversible watermarking methods which could enable the extraction of the original image from the watermarked version without any loss of information. Moreover, we can use the techniques for embedding multiple watermarks into the same image that could enhance the versatility of the app.