Chapter 3
Transform Domain Watermarking

Transform Domain Video Watermarking

The transform domain approach for video watermarking operates on the frequency representation of an image, contrasting with spatial domain techniques. Algorithms based on DCT and DWT are the most commonly used methods in this domain. These techniques transform an image from its spatial domain into the frequency domain, organizing frequency coefficients based on human perception. The coefficients are then modulated to embed watermark data, following three main steps:

1. **Forward Transformation**: Converts the spatial domain image into the frequency domain, producing frequency coefficients.
2. **Coefficient Modification**: Alters these coefficients based on the watermark information.
3. **Inverse Transformation**: Converts the modified frequency domain representation back into the spatial domain.

3.1 Discrete Cosine Transform (DCT)

The DCT converts a 2D spatial domain image into its frequency domain equivalent. It maintains the size of the transformed image equal to the original and positions the DC coefficient, representing low frequencies, in the top-left corner. The remaining coefficients, termed AC coefficients, increase in frequency along a zigzag path. The DC coefficient, always an integer, ranges from -1024 to 1023, while AC coefficients may be integers or non-integers. DCT's ability to distinguish between frequency components makes it an effective tool for watermarking. Most critical information is found in low-frequency components, while mid-frequency components are ideal for watermarking due to their balance of robustness and imperceptibility.

The ability of DCT to differentiate frequency components effectively makes it a highly useful tool for watermarking applications. The mathematical representations for two-dimensional DCT and its inverse are provided in Equations 1 and 2, respectively.

$$\mathbf{F(u,v) = \alpha(u)\alpha(v)\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\cos[\frac{(2x+1)u\pi}{2M}]\cos[\frac{(2y+1)v\pi}{2N}]} \qquad --(3.1)$$

Where $\alpha(u) = \sqrt{1/M}$ for u=0;

$\alpha(u) = \sqrt{2/M}$ for u=1,2,3,…..M-1;

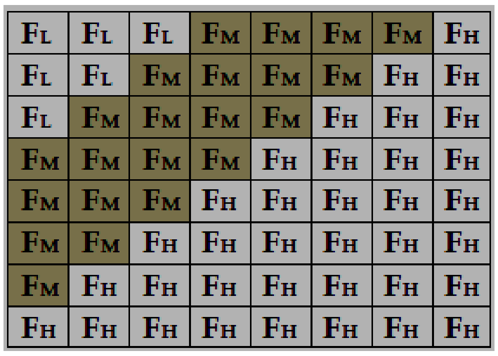
$\alpha(v) = \sqrt{1/N}$ for v=0;

$\alpha(v) = \sqrt{2/N}$ for v=1,2,3,…..N-1;

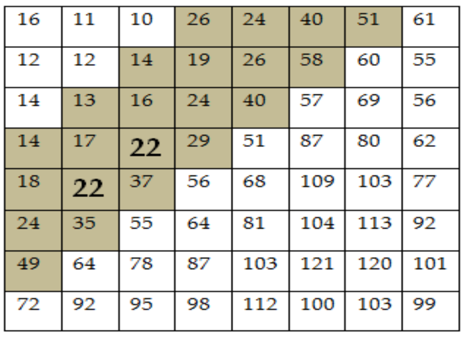
$$\mathbf{f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} \alpha(u)\alpha(v)F(u,v)\ \cos[\frac{(2x+1)u\pi}{2M}]\ \cos[\frac{(2y+1)v\pi}{2N}]} \qquad --(3.2)$$

Where x = 0, 1, 2,……..M-1, y = 0,1,2,……..N-1

Figure 3.1 (a) DCT frequency distribution.



F_L	F_L	F_L	F_M	F_M	F_M	F_M	F_H
F_L	F_L	F_M	F_M	F_M	F_M	F_H	F_H
F_L	F_M	F_M	F_M	F_M	F_H	F_H	F_H
F_M	F_M	F_M	F_M	F_H	F_H	F_H	F_H
F_M	F_M	F_M	F_H	F_H	F_H	F_H	F_H
F_M	F_M	F_H	F_H	F_H	F_H	F_H	F_H
F_M	F_H	F_H	F_H	F_H	F_H	F_H	F_H
F_H	F_H	F_H	F_H	F_H	F_H	F_H	F_H



16	11	10	26	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

(a) (b)

Figure 3.1: (a) DCT based classification of Frequency (b) Values of Quantization given in the JPEG compression Scheme

Here F_L = Components with low frequency, F_M = Components having mid-level Frequency and F_H = Components having high-level frequency.

3.1.1 Embedding Process

The following steps outline the procedure for embedding a watermark into video frames using this method:

1. **Frame Extraction**: The original video is divided into individual frames.
2. **Face Detection**: Faces within each frame are detected and localized using algorithms such as Viola-Jones or advanced deep learning-based face detectors.
3. **Color Space Transformation**: The frames undergo a conversion from the RGB color space to the YCbCr color space.
4. **Frame Blocking**: The luminance (Y) component is divided into non-overlapping blocks.

5. **DCT Application**: Discrete Cosine Transform (DCT) is applied to each block to convert spatial data into frequency components.
6. **Frequency Component Modification**: Specific mid-frequency diagonal components, such as (5, 2) and (4, 3), are adjusted based on the watermark properties:
   - For black watermark pixels, ensure (5, 2) > (4, 3). If not, swap their values.
   - For white watermark pixels, ensure (5, 2) < (4, 3). If not, swap their values.
   - Ensure the difference between these two frequency components exceeds the gain factor. If not, adjust the values by adding or subtracting the gain factor to achieve the desired difference.
7. **Inverse DCT**: Apply inverse DCT to the modified blocks to reconstruct the spatial data.
8. **Color Space Reversion**: Convert the modified YCbCr data back to the RGB color space to obtain the updated frame.
9. **Face Image Reinsertion**: Replace the original face region within the frame with the watermarked face image.
10. **Frame Processing Iteration**: Repeat steps 2 to 9 for each subsequent frame until all frames in the video are processed.
11. **Video Reconstruction**: Combine all the modified frames to reconstruct the watermarked video.

This step-by-step process ensures precise embedding of the watermark while maintaining the video's overall perceptual quality.

(a)

Watermarked: 1 **Watermarked: 2** **Watermarked: 3** **Watermarked: 4** **Watermarked: 5**

(b)

Figure 3.2: DCT based Watermarking with K=100 and BS=8 (a) 5 frames of video (b) Watermarked Frames

3.1.2 Extraction Process

1. Divide the watermarked video into frames, Find the faces and convert the color space to YCbCr.
2. Apply DCT on the Y frame's blocks and extract mid-frequency coefficients.
3. Compare coefficients to determine the watermark's binary values.
4. Repeat for all frames to recover the watermark.

Recovered:1 **Recovered:2** **Recovered:3** **Recovered:4** **Recovered:5**

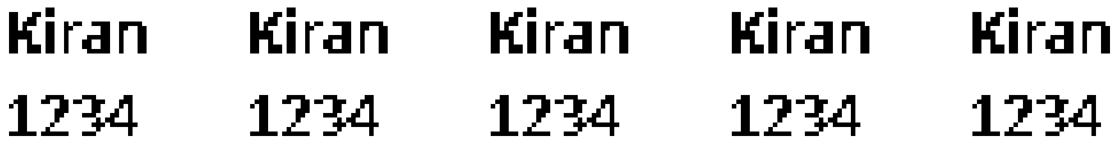


Figure 3.3: Recovered Messages

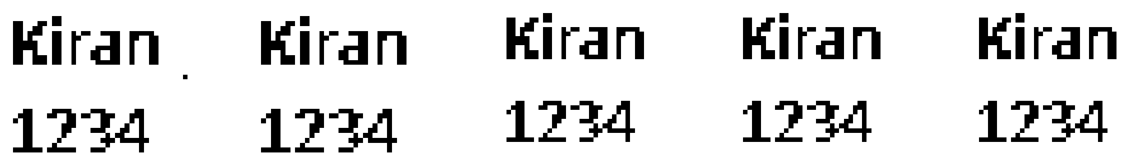
3.1.3 Results

Alpha=10 **Alpha=30** **Alpha=50** **Alpha=70** **Alpha=90**

43.6481 **39.6156** **36.3418** **34.1563** **32.467**

(a)

Alpha=10 **Alpha=30** **Alpha=50** **Alpha=70** **Alpha=90**



0.9814 **0.9839** **0.9839** **0.9839** **0.9839**

(b)

Figure 3.4: With Various values of K (a) Frame 1 - Watermarked (b) Messages at the receiver end

Frame No.	**PSNR (db)**	**MSE**	**Correlation**
1	29.1135	36.8455	0.9839
2	29.3435	35.235	0.9839
3	29.4035	36.0056	0.9839
4	28.3435	37.1340	0.9839
5	29.5221	35.8411	0.9839

Table 3.1: DCT based watermarking with K=100 & BS=8

Alpha	**PSNR (db)**	**MSE**	**Correlation**
10	43.6481	2.8072	0.9814
20	41.5132	4.5894	0.9839
30	39.6156	7.1042	0.9839
40	37.8874	10.5765	0.9839
50	36.3418	15.0973	0.9839
60	35.1874	19.6945	0.9839
70	34.1563	24.9716	0.9839
80	33.2444	30.8064	0.9839
90	32.467	36.8455	0.9839
100	31.8645	42.3286	0.9839

Table 3.2: DCT based watermarking with various values of K

3.1.4 Observations and Results

1. Increasing the gain factor reduces perceptibility but enhances robustness.
2. Frames with a PSNR above 28 dB are visually acceptable, while correlation values above 0.50 ensure identifiable messages.
3. The method is robust against some attacks (e.g., color reduction, histogram equalization) but vulnerable to others (e.g., average filtering, rotation).

3.1.5 Observations

Following are the observations after successfully implementing both embedding & extraction algorithms, assuming K=100 for analysis in case of visual metrices:

1. As the gain factor increases, the perceptibility of the method decreases.
2. The robustness of the method improves with an increase in K.
3. When the PSNR value exceeds 28 dB, the frames appear visually acceptable. Additionally, if the correlation value exceeds 0.50, the embedded message is distinguishable.
4. The method shows limited robustness against attacks such as average filtering, median filtering, and image rotation.
5. The method demonstrates partial robustness against attacks like Gaussian low-pass filtering, compression, Gaussian noise, salt-and-pepper noise, and speckle noise.
6. The method is highly robust against , histogram equalization, , linear camera motion, color reduction, cropping, and high-pass filtering attacks.

3.1.6 Comparison - Correlation based watermarking Method:

• For the same gain factor, there is a significant difference in perceptibility, while the variation in robustness is minimal.

• Spackle noise and Compression attacks result in lower robustness, whereas higher robustness is observed against Gaussian low-pass filtering, linear camera motion, and attack such as high-pass filtering

3.2 DWT in Image Processing

3.2.1 Introduction

The wavelet transform is useful in almost all image processing applications, including compression, signal analysis, digital watermarking, and general signal processing. These applications have become increasingly effective and practical over the last few decades due to the use of wavelet transforms.

Unlike periodic waves, which maintain consistent oscillations over time or space, wavelets are localized waves that concentrate their energy within specific time or spatial regions. This characteristic makes wavelets highly effective for signal analysis. The Discrete Wavelet Transform (DWT) works by convolving signals, either 1D or 2D, with wavelets across various time scales and positions. This approach is computationally efficient, requires fewer resources, and is straightforward to implement.

DWT operates on the principle of sub-band coding, where signals are subjected to filtering and sub-sampling. Filtering determines the resolution by retaining specific amounts of signal information, while sub-sampling adjusts the signal's scale. Using a combination of low-pass and high-pass filters, DWT performs multi-level decomposition. At each decomposition level, the frequency resolution is doubled, reducing frequency uncertainty by half while halving the time resolution. For instance, a signal with 500 samples will be reduced to 250 samples after the first decomposition level.

One of the key strengths of DWT is its ability to balance time and frequency resolutions. It provides high time resolution for higher frequencies and enhanced frequency resolution for lower frequencies, making it an adaptable and versatile tool for signal processing.

Understanding the DWT

For a one-dimensional signal, the process involves passing the signal through both a low-pass filter (capturing low-frequency components, or "information") and a high-pass filter (capturing high-frequency components, or "edges"). The low-frequency part undergoes further decomposition to extract additional low and high-frequency components, a process that can continue for multiple levels based on the application requirements. Commonly, applications like compression or watermarking use up to five decomposition levels.

The original signal can be reconstructed from its DWT coefficients using the Inverse Discrete Wavelet Transform (IDWT).

Figure 3.7 illustrates the basic filtering and decomposition process. Low-frequency components typically carry the most significant information in a signal, while high-frequency components are less critical. For example, in human speech, removing high-frequency components may change the sound's clarity but still retain intelligibility. Removing low-frequency components, however, can render the speech incomprehensible.

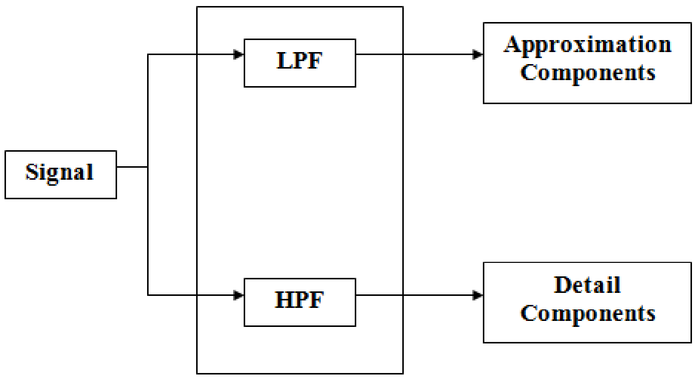


Figure 3.5: Filtering or decomposition process at its most basic level

Approximations and Details in Wavelet Analysis

Wavelet analysis focuses on two key elements:

1. **Approximations** – High-scale, low-frequency components.
2. **Details** – Low-scale, high-frequency components.

The decomposition process applies a signal to low-pass and high-pass filters, producing outputs twice the size of the input. To maintain consistency, the outputs are down-sampled by a factor of 2, reducing each output's size to match the original signal. This concept is depicted in Figure 3.8.

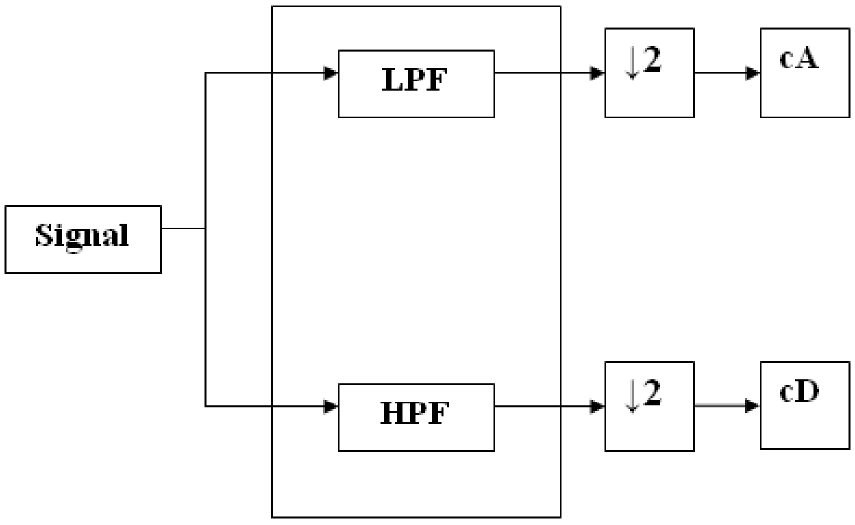


Figure 3.6: Analysis with down sampling

Mathematically, the equations for low-pass and high-pass filters are represented as:

$$\mathbf{H\,(w)} = \sum_{\mathbf{k}} \mathbf{hk\ e^{-jkw}} \qquad ----(3.3)$$

$$\mathbf{H\,(w)} = \sum_{\mathbf{k}} \mathbf{hk\ e^{-jkw}} \qquad ----(3.4)$$

These equations determine how signals are divided into successive approximation components through iterative decomposition, forming a wavelet decomposition tree. Figure 3.9 illustrates this multi-level decomposition.

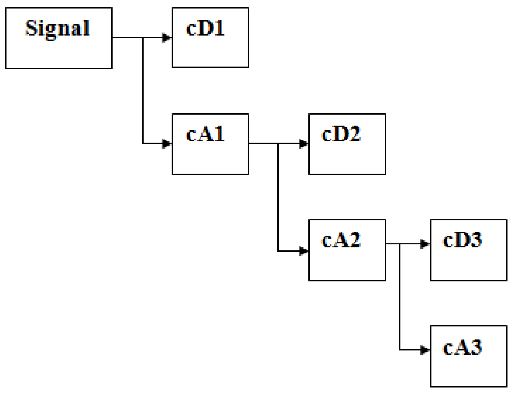


Figure 3.7: Multiple-level decomposition or analysis

2D DWT for Images

For a two-dimensional image F(x,y), the DWT and IDWT processes are applied first along the x-dimension and then along the y-dimension. This decomposition results in a pyramidal representation of the image, dividing it into four components:

1. **Approximation Component** – Captures the core image structure.
2. **Horizontal Detail Component** – Contains horizontal edge information.
3. **Vertical Detail Component** – Contains vertical edge information.
4. **Diagonal Detail Component** – Contains diagonal edge information.

This method of decomposition is highly effective for various image-processing applications, providing robust capabilities for analysis and reconstruction.

Significance of Low-Frequency and High-Frequency Components

In the realm of wavelet analysis, low-frequency components, also known as approximations, hold the majority of the signal's essential information. These components are crucial for tasks like image reconstruction and noise reduction. High-frequency components, or details, typically represent finer details such as edges and noise. While they are less critical for representing the primary structure of the signal, they are essential for enhancing clarity and emphasizing boundaries in image processing tasks.

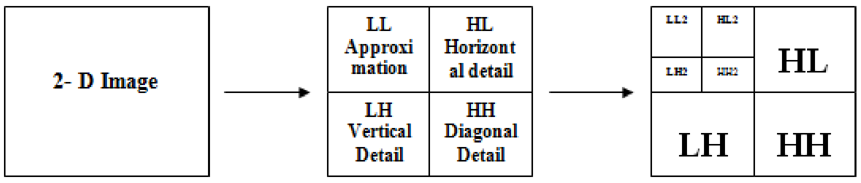
For instance, in audio signals, removing high-frequency components may alter the tonal quality but still leave the main message understandable. Conversely, if low-frequency components are removed, the result will lack coherence, rendering the message incomprehensible. This principle applies to images as well, where the low-frequency approximations define the broader structure, and the high-frequency details contribute to the finer textures and edges.

Multi-Level Decomposition in DWT

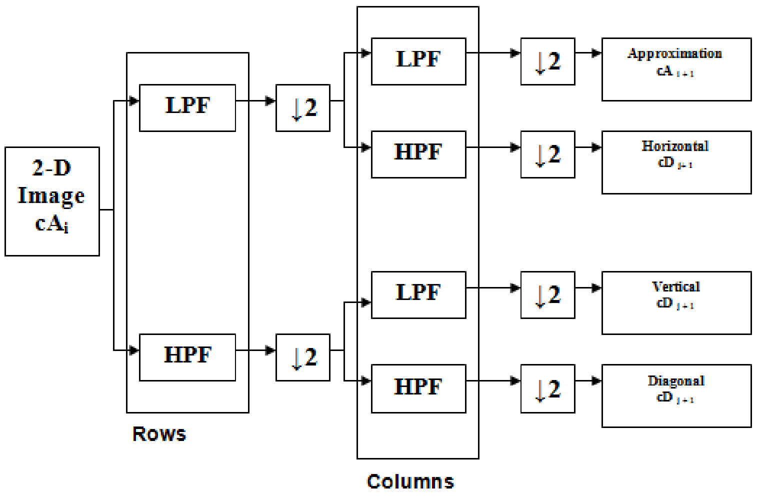
One of the powerful aspects of DWT is its ability to perform multi-level decomposition. By iteratively applying the wavelet transform to the approximation components, an image or signal can be decomposed into multiple levels, with each level offering finer resolution.

This process generates a hierarchical structure known as the wavelet decomposition tree, as shown in Figure 3.9. At each successive level, the signal is split into four components:

1. **LL (Approximation)** – The low-frequency component containing the majority of the image information.
2. **LH (Horizontal Detail)** – Captures horizontal edges in the image.
3. **HL (Vertical Detail)** – Captures vertical edges in the image.
4. **HH (Diagonal Detail)** – Represents diagonal edges and finer textures.



(a)



(b)

Figure 3.8: Basic decomposition steps for images

The decomposition process effectively isolates the frequency content at different scales, enabling targeted analysis and manipulation. For instance, compression algorithms can prioritize the LL component while selectively discarding or compressing the detail components (LH, HL, HH) to reduce file size. Similarly, watermarking applications embed hidden data in these detail components without significantly altering the image's visual quality.

Reconstruction with IDWT

The Inverse Discrete Wavelet Transform (IDWT) reconstructs the original signal or image from its decomposed components. By combining the approximation and detail coefficients from each level in reverse order, the original data can be accurately restored.

This reconstruction relies on the perfect reconstruction property of wavelet transforms, ensuring no loss of information during the decomposition and synthesis processes. This makes wavelets highly effective for lossless compression and signal recovery applications.

3.2.2 Applications of DWT

The DWT has found extensive applications in various fields, including:

1. **Image Compression** – The JPEG2000 standard, for instance, employs wavelet-based compression techniques to achieve high compression ratios without significant loss in image quality.
2. **Digital Watermarking** – By embedding watermark data into high-frequency components, wavelet-based watermarking techniques ensure robustness against attacks like compression and noise addition.
3. **Noise Reduction** – The ability to isolate noise in high-frequency components allows DWT to be used in denoising signals and images while preserving essential information.
4. **Feature Extraction** – In machine learning and pattern recognition, DWT helps extract features by isolating different frequency bands, aiding in classification tasks.
5. **Signal Analysis** – DWT's multi-resolution property enables the analysis of non-stationary signals, making it ideal for applications in audio processing, biomedical signal analysis, and more.

3.2.3. Advantages of DWT

1. **Multi-Resolution Analysis** – DWT allows simultaneous time and frequency domain analysis, making it versatile for signals with varying frequency content over time.
2. **Efficient Computation** – The sub-band coding approach and down-sampling reduce computational complexity, making DWT suitable for real-time applications.

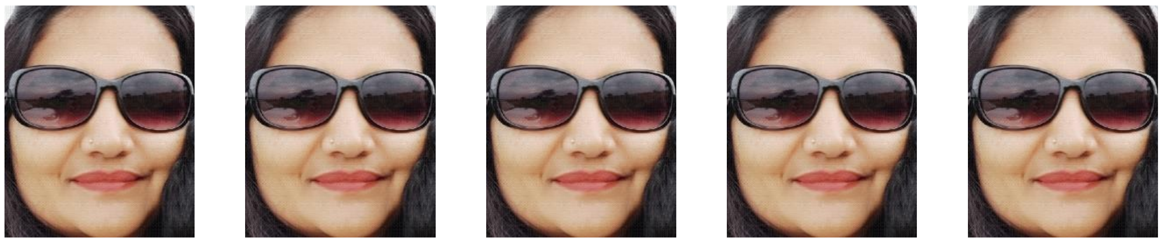
3. **Compact Representation** – DWT provides a sparse representation of signals, reducing storage requirements while preserving critical information.
4. **Scalability** – The hierarchical decomposition facilitates analysis at different resolutions, enabling scalability in applications like image compression.

3.2.4 Embedding Process

The process of video watermarking using Discrete Wavelet Transform (DWT) is outlined as follows:

1. The original video is divided into individual frames.
2. Face detection is performed within each frame, utilizing algorithms such as Viola-Jones or deep learning-based methods to locate and identify faces.
3. Two random sequences are generated, referred to as *pn_sequence_one* and *pn_sequence_zero*.
4. The color space of the frame is converted from RGB to YCbCr format.
5. The Discrete Wavelet Transform (DWT) is applied to the Y channel, and the HL (horizontal-low) component is selected for embedding.
6. The HL component is split into non-overlapping blocks for further processing.
7. For each block, if the message bit is zero, the watermark block is filled with *pn_sequence_zero*; otherwise, it is filled with *pn_sequence_one*. This process is repeated for all blocks.
8. The original HL component is combined with the weighted watermark block, where the weight is controlled by a parameter known as the gain factor.
9. Inverse DWT is applied to reconstruct the Y frame with the embedded watermark.
10. The color space is converted back from YCbCr to RGB, resulting in a modified RGB frame.
11. The watermarked face image, containing the embedded watermark, is placed back into its original position within the frame.
12. Steps 3 through 11 are repeated for each subsequent frame until the final frame is processed.
13. The watermarked video is generated by combining all the watermarked frames.

(a)



(b)

Figure 3.9: DWT based watermarking method with K=100 and BS=8 (a) 5 frames of video (b) Watermarked Frames

3.2.5 Extraction Process

1. The watermarked video is divided into multiple individual frames.
2. Facial regions within each frame are identified and localized using algorithms like Viola-Jones or deep learning-based face detection methods.
3. Two pseudo-random sequences, named pn_sequence_one and pn_sequence_zero, are generated. These sequences must match those used during the embedding process.
4. The color space of each frame is converted from RGB to YCbCr format.
5. A discrete wavelet transform (DWT) is applied to the Y channel, and the HL sub-band is selected for further processing.
6. The HL sub-band is divided into non-overlapping blocks for analysis.
7. Each block is correlated with both pseudo-random sequences. If the correlation is higher with pn_sequence_zero, the extracted bit is set to 0; otherwise, it is set to 1. This process is repeated for all blocks to retrieve the embedded watermark.
8. Steps 3 to 7 are repeated for subsequent frames until the entire video has been processed and the watermark is fully extracted.

Recovered:1 **Recovered:2** **Recovered:3** **Recovered:4** **Recovered:5**

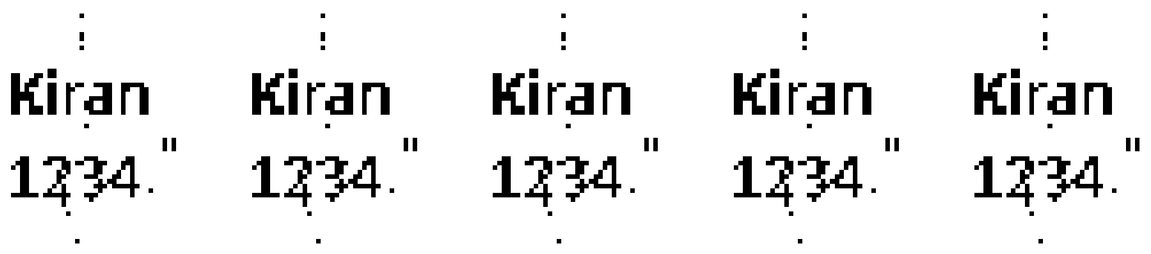
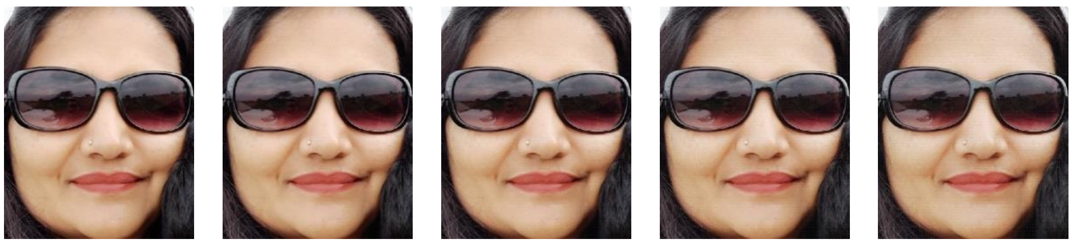


Figure 3.10: Recovered Messages

3.2.6 Results

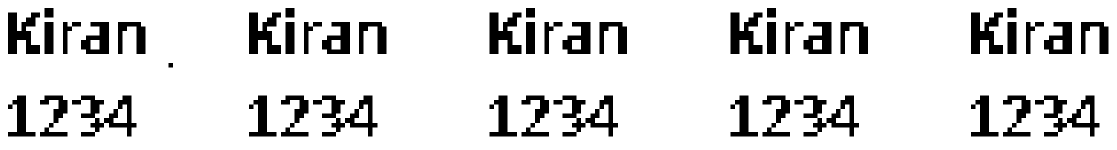
Alpha=10 **Alpha=30** **Alpha=50** **Alpha=70** **Alpha=90**



46.7082 **40.7597** **36.887** **34.6422** **33.1722**

(a)

Alpha=10 **Alpha=30** **Alpha=50** **Alpha=70** **Alpha=90**



0.5678 **0.8012** **0.8741** **0.8996** **0.9285**

(b)

Figure 3.11: Results with various gain factors (a) Watermarked Frame 1 (b) Recovered Messages

Table 3.3: DWT with K=100 & BS=8

Frame No.	**PSNR (db)**	**MSE**	**Correlation**
1	32.9991	35.8512	0.9351
2	32.5858	35.1212	0.9310
3	31.4432	35.6785	0.9364
4	32.9010	34.8070	0.9333
5	32.5321	35.9807	0.9384

Alpha	PSNR (db)	MSE	Correlation
10	46.7082	1.3876	0.5678
20	43.4634	2.9291	0.7371
30	40.7597	5.459	0.8012
40	38.5981	8.9799	0.8432
50	36.887	13.3163	0.8741
60	35.6228	17.8154	0.8935
70	34.6422	22.3286	0.8996
80	33.8469	26.8156	0.8996
90	33.1722	31.3231	0.9285
100	32.5858	35.8512	0.9351

Table 3.4: DWT with Various values of K

3.2.7 Observations

Below are the observations recorded after successfully implementing the embedding and extraction algorithms. For these observations, a gain factor of 100 was used to evaluate both perceptibility and robustness. A higher PSNR value indicates better perceptibility, while a higher correlation value reflects stronger robustness.

1. Increasing the gain factor results in a reduction in perceptibility.
2. Robustness improves as the gain factor increases.
3. Frames appear visually acceptable when the PSNR exceeds 28 dB, and the embedded message becomes distinguishable when the correlation value is above 0.50.
4. This approach is vulnerable to attacks such as average filtering, median filtering, rotation, and high-pass filtering.
5. Partial robustness is observed against attacks like Gaussian low-pass filtering, compression, color reduction, Gaussian noise, salt-and-pepper noise, and speckle noise.
6. The method demonstrates complete robustness against histogram equalization, linear camera motion, and cropping.

3.2.8 Comparison - Correlation Method & DCT Method:

1. Among the three methods, this approach offers the highest perceptibility at the same gain factor, albeit with a significant difference in robustness.

2. Compared to the other two methods, this approach exhibits lower robustness against most types of attacks.