**Watermarking Lab 2 – Embedding Strategies, Attack Localization, and Frequency Analysis**

**1. Embedding Foundations**

To add a watermark, part of the image signal must be slightly modified — “cut away” in the sense of embedding within redundant information.  
Invisible embedding techniques modify **channels with higher redundancy** and lower perceptual sensitivity.

Chrominance channels (**Cb**, **Cr**) are both suitable for invisible watermarking.

If a region has **high entropy** (high information, low redundancy), it should **not** be modified = If a region has **high entropy (i.e., low redundancy / high information content)** do not modify it.

If a region has **high redundancy** it is suitable for embedding.

Note: chroma channels (Cb/Cr) are attractive for invisibility but are vulnerable to chroma subsampling (e.g., 4:2:0) and chroma quantization during JPEG — design and test accordingly.

If a region has **high redundancy**, it is **safe** to embed in it.

**Clarification:**  
Entropy ≈ unpredictability. High-entropy areas (edges, noise) carry unique information.  
Redundancy ≈ repetition. Embedding in redundant channels or regions hides changes more effectively.

**2. Localization and Redundancy**

Localizing both the **embedding** and the **attack** is the best practice.  
Embedding globally often dilutes robustness, while localizing (for example, embedding in a region like 1000 × 1024 pixels) allows better control and evaluation.

In earlier CTMs, the winning approach placed the watermark “on the side,” which worked poorly because it was not adaptive.  
  
A good embedding strategy should follow image structure — edges, textures, and redundancy patterns.

**3. Edge-Based Embedding**

**3.1. Principle**

Edge-based methods start with an **edge detection algorithm** (for example, Canny).  
First, extract **horizontal edges** or combine vertical and diagonal edges as needed.  
Edges correspond to **high-frequency** areas, and many algorithms embed the watermark inside them.  
  
*First, extract horizontal edges or combine vertical and diagonal edges as needed.*  
**with**:

1. First, run an edge detector (for example **Canny**).
2. Extract horizontal, vertical and diagonal edges or a combined edge map as needed; many implementations use morphological or directional filters to focus on a subset (e.g., horizontal).

Although it seems counterintuitive, embedding in textured, high-frequency areas is often beneficial — the watermark becomes **invisible** there because local contrast hides small distortions.

However, if attackers specifically target these regions (for example, with blur or compression), the watermark can be destroyed quickly.  
Therefore, there must always be a **trade-off between robustness and invisibility**.

**Clarification:**

* Canny edge detection is useful for locating good embedding zones.
* The more texture, the more invisible the watermark, but also the less robust it becomes.

**4. Frequency-Domain Interpretation**

The image can be viewed as composed of frequencies:

| **Region type** | **Frequency type** | **Visibility** | **Robustness** |
| --- | --- | --- | --- |
| Flat area | Low frequency | High | High |
| Textured / detailed | High frequency | Low | Low |
| Mixed / mid-detail | Mid frequency | Balanced | Balanced |

Embedding in **mid-frequency regions** achieves the best compromise between imperceptibility and robustness.

High-frequency regions can also be identified in the **frequency domain** (FFT, DCT, DWT).  
Low-frequency areas correspond to smooth regions; high-frequency to textures.

**5. Common Image Attacks**

This section lists common operations used as watermark attacks or robustness tests.  
Each simulates a different degradation scenario.

**5.1. Additive White Gaussian Noise (AWGN)**

Adds random noise uniformly distributed across the image.  
If the amplitude is too high, it can destroy both the image and the watermark.

**Clarification:**  
Use realistic noise levels. AWGN mainly tests robustness of watermarks embedded in low-frequency or mid-frequency regions.

**5.2. Gaussian Filtering**

Removes high-frequency details across the image.  
Be careful with parameters — excessive filtering can erase the watermark or overly blur the image.  
It is particularly effective against **high-frequency watermarks**.

**5.3. Sharpening**

The inverse of blurring.  
It amplifies edges and high-frequency content, which can distort watermarks embedded in low-frequency regions.  
  
Sharpening amplifies high-frequency content (edges and fine detail). It can *either* reveal artifacts of embedding in mid/low-frequency bands or disturb high-frequency embedded watermarks depending on the embedding domain.

**Clarification:**  
Sharpening enhances contrast and can act as a local attack. Use localized sharpening rather than global, for more realistic testing.

**5.4. Median Filtering**

Preserves most frequent pixel values (median of neighborhood).  
It maintains visual quality better than Gaussian blur.  
  
A parameter of **0.1 corresponds to about 10 % false positives (FP)** when evaluating.  
Note: any “0.1 = 10% FP” claim is context-dependent — the false positive rate depends on detector thresholding and test dataset. Median filtering itself does not imply a fixed FP; report FP measured in your ROC experiments.

Median filtering can be applied globally, but resizing and sharpening should be applied **locally** for better control.

**5.5. Resizing**

Downscaling (for example, 4×4 → 2×2) and then upscaling back causes **irreversible information loss**.  
It acts similarly to blurring because small features are averaged out.

**5.6. JPEG Compression**

Divides the image into **8×8 blocks** and converts colors (usually to YCbCr).  
Even converting between RGB and YCbCr can weaken a watermark.  
In fact, converting an image from RGB to YCbCr and back can sometimes be sufficient to partially or completely remove a weak watermark, even without applying resizing or compression.  
  
In particular, converting to YCbCr and applying chroma subsampling (for example 4:2:0) or quantizing chroma channels can significantly degrade or remove watermarks located in Cb/Cr — this is a common, inexpensive way to attack chroma-domain embeddings.

JPEG compression is among the **strongest attacks**, as it removes high-frequency details.

* Quality parameter should be as high as possible (near 100) for moderate attacks.
* JPEG **quality** controls attack strength: lower quality → stronger attack (more frequency quantization). For testing, sweep quality values (e.g., 90, 70, 50, 30) to measure robustness; use high values (≥90) for light attacks and low values (≤50) for strong attacks.
* JPEG can be applied globally.

**Clarification:**  
JPEG directly challenges watermarking in the transform domain, especially DCT-based methods, since quantization discards frequency coefficients.

**6. Global vs Local Application**

* **Global attacks** (e.g., full-image compression, blur) test overall robustness.
* **Localized attacks** (e.g., patch blur, regional sharpening) test resilience of spatially embedded watermarks.
* Redundancy can help offset local damage.
* Balance between **redundancy** and **entropy** is key: redundant zones hide data, high-entropy zones should remain untouched.

**7. Transform-Domain Embedding**

Embedding in the transform domain allows frequency-selective control and higher robustness.

**7.1. Fourier Transform (FFT)**

Transforms the spatial image into frequency components.  
Typically, **high frequencies** appear at the center and **low frequencies** near the corners (depending on implementation).  
Using fftshift can invert this mapping (moving high frequencies to corners).

* Contains **real and imaginary** parts.  
  Modify only **real** components to avoid visual distortion = modify **magnitude** or apply small perturbations to selected real parts; avoid large arbitrary phase changes. Always test inverse transform visually.
* Do not modify the **bright central area** (low frequencies).
  + The DC coefficient reflects average block brightness; altering it causes noticeable luminance shifts — avoid it. Embed in mid-frequency AC coefficients within 8×8 blocks (for JPEG/DCT domains).
* Embed towards the edges or mid-regions.
* fftshift → high frequencies to corners.  
  fft → high frequencies at center.
* **Inverse FFT** reconstructs the image back.

FFT can also be used to **locate or visualize attacks** by comparing magnitude spectra.  
  
The 2D discrete Fourier Transform (FFT) produces complex coefficients. By convention the zero frequency (DC) is located at the origin index; calling fftshift moves the DC (low frequencies) to the center of the array and moves high frequencies toward the corners. When inspecting or embedding, state whether you operate on the shifted spectrum or the unshifted one. Coefficients are complex (magnitude and phase). Typical safe modifications adjust magnitudes or small real-part perturbations; large arbitrary changes to phase (imaginary components) can produce visible artefacts. Avoid changing the DC (low-frequency / bright central region after fftshift). Prefer mid/outer bands for embedding. Use inverse FFT to reconstruct the image.

**7.2. Discrete Cosine Transform (DCT)**

More efficient than FFT, widely used for image compression (including JPEG).  
It is also ideal when detection must run quickly.

* The **DC component** (top-left) represents the image average — **do not modify** it.
* Embed watermark in **AC components** (medium frequencies).
* Always perform **Inverse DCT (IDCT)** after embedding.

**Clarification:**  
DCT strikes the best balance between efficiency, robustness, and invisibility; it is the most practical domain for digital watermarking.

**7.3. Wavelet Transform (DWT)**

Decomposes the image into multiple subbands:  
**LL, HL, LH, HH** (low-low, high-low, low-high, high-high).  
Each subband represents different directional frequencies.

* Helps localize regions for watermark embedding without blurring the image.
* Allows **multi-level embedding** — the watermark can be embedded several times across scales.
* If embedded only once, a single successful attack can remove it.  
  Embedding redundantly increases robustness but decreases invisibility.
* Practical recommendation: embed redundantly across **2–4** subbands/levels (e.g., HL and LH at two scales) and test perceptual impact; more than 4 redundant embeddings usually yields noticeable artefacts.

**Clarification:**  
Embedding 2–3 times across DWT subbands (for example HL and LH) is a good compromise between invisibility and robustness.

**8. Summary and Practical Guidelines**

1. **Choose embedding zones wisely:**
   * Modify redundant, low-entropy regions.
   * Avoid unique, high-entropy data.
2. **Channels:** use **Cb/Cr**, **not luminance**, for invisible embedding.
3. **Frequency strategy:**
   * Low-freq → robust, visible.
   * High-freq → invisible, fragile.
   * Mid-freq → optimal compromise.
4. **Localization:** embed in edges or selected regions, not uniformly.
5. **Transform preference:**
   * FFT: analysis and visualization.
   * DCT: efficient embedding.
   * DWT: multiresolution, high robustness.
6. **Redundancy:** embed watermark more than once in different zones or subbands.
7. **Evaluate using attacks:**
   * AWGN, Gaussian, median, sharpening, resizing, JPEG compression.
8. **Parameter discipline:**
   * Avoid excessive filtering (loss of watermark).
   * Keep WPSNR > 35 dB for visual acceptability.
9. **Always analyze the trade-off:** robustness ↑ → invisibility ↓.
10. Suggested parameter ranges for experiments (examples):  
    • AWGN σ: 1–10;  
    • Gaussian blur σ: 0.5–3.0 (kernel sizes 3–11);  
    • Median filter kernel: 3–9;  
    • Resizing scale: 0.5–0.95 downscale then upsample;  
    • JPEG quality: 30–100 (test 30, 50, 70, 90);  
    Randomize parameter selection per trial and log seeds.
11. **Final rule:** localized embedding + redundancy + mid-frequency placement = most stable outcome.