

Multimedia Security and Privacy

TP4: Watermark Detection

Prof. SVIATOSLAV VOLOSHYNOVSKIY,
ROMAN CHABAN <roman.chaban@unige.ch>,
DENIS ULLMANN <denis.ullmann@unige.ch>.

Stochastic Information Processing Group

April 21, 2021

Submission

Please archive your report and codes in “Name.Surname.zip” (replace “Name” and “Surname” with your real name), and upload to “Assignments/TP4: Watermark Detection” on <https://moodle.unige.ch> before **Wednesday, May 5th 2021, 23:59 PM**. Note, **the assessment is mainly based on your report, which should include your answers to all questions and the experimental results.**

1 Watermark Embedding and Channel modeling

1.1 Exercise

- Read in a gray scale image \mathbf{x} , for example `liftingbody.png`. It will serve as the host image
- Generate a matrix \mathbf{w}' of size \mathbf{x} with uniform distributed values $\{-1, 1\}$. The magnitude of these two values governs the watermark strength.
- Randomly sample from matrix \mathbf{w}' with a given density $\theta_N = 0.5$. The result is watermark \mathbf{w} . The watermark length N is defined as:

$$N = (N_1 \cdot N_2) \cdot \theta_N \quad (1)$$

where N_1 and N_2 are the dimensions of host image \mathbf{x} .

- Embed the watermark \mathbf{w} in to the host image \mathbf{x} forming watermarked image \mathbf{y} :

$$\mathbf{y} = \mathbf{x} + \mathbf{w} \quad (2)$$

- Generate a matrix \mathbf{z} with Additive White Gaussian Noise (AWGN):

$$\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \sigma_{noise}^2 \mathbf{I}) \quad (3)$$

where $\boldsymbol{\mu} = 0$ and $\sigma_{noise}^2 = 1$.

- Add the AWGN matrix \mathbf{z} to the watermarked image \mathbf{y} , resulting in attacked image \mathbf{v} :

$$\mathbf{v} = \mathbf{y} + \mathbf{z} = \mathbf{x} + \mathbf{w} + \mathbf{z} \quad (4)$$

- Show the original image \mathbf{x} , the watermarked image \mathbf{y} and the attacked watermarked image \mathbf{v} on the screen. What do you see?

2 Non-blind watermark detection

In this set exercises you will attack a watermarked image leveraging knowledge of the original image. See Figure 1.

2.1 Exercise

- Extract the estimated watermark $\hat{\mathbf{w}}$ from the marked image \mathbf{v} using the original image \mathbf{x} :

$$\hat{\mathbf{w}}_{non-blind} = \mathbf{v} - \mathbf{x} \quad (5)$$

- Determine the linear correlation $\rho_{non-blind}$ between the original watermark \mathbf{w} and the estimated watermark $\hat{\mathbf{w}}_{non-blind}$:

$$\rho_{non-blind} = \frac{1}{N} \sum_{k=1}^N \hat{w}_{non-blind}[k] \cdot w[k] \quad (6)$$

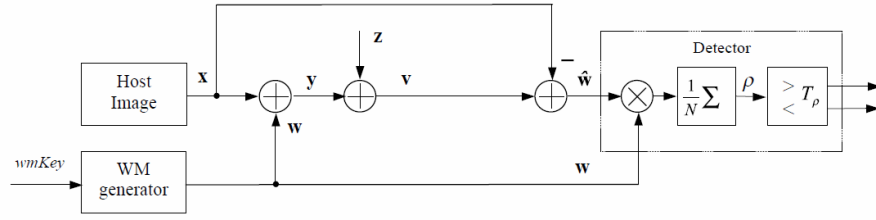


Figure 1 – Non-blind watermark detection

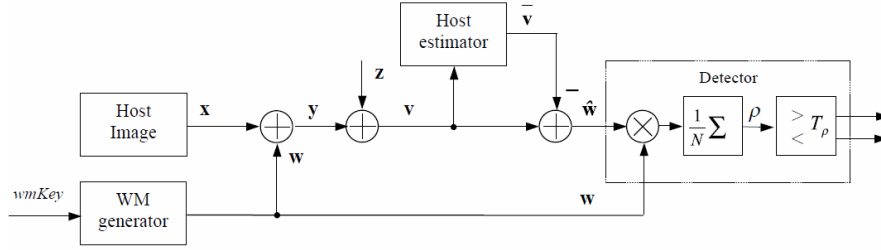


Figure 2 – Blind watermark detection

3 Blind watermark detection using the Maximum Likelihood estimate

This exercise will introduce a simple method to blindly estimated a watermark when one only has access to the marked image. See Figure 2.

3.1 Exercise

- Given only the marked image \mathbf{v} , blindly estimate the watermark $\hat{\mathbf{w}}_{blind}$ using the following formula:

$$\hat{\mathbf{w}}_{blind} = \mathbf{v} - \bar{\mathbf{v}} \quad (7)$$

where $\bar{\mathbf{v}}$ is the local mean of marked image \mathbf{v} .

- Why can you assume that $\bar{\mathbf{v}} = \mathbf{x}$?

Remark 1. You could properly use Python-numpy functions `numpy.reshape`, `numpy.mean`, `numpy.var`, or the Python-skimage function `skimage.block_reduce`.

Matlab offers a number of functions to facilitate region based processing, amongst them are `blockproc`, `blkproc` and `colfilt`.

3.2 Exercise

- Determine the linear correlation ρ_{blind} between the original watermark \mathbf{w} and the blindly estimated watermark $\hat{\mathbf{w}}_{blind}$:

$$\rho_{blind} = \frac{1}{N} \sum_{k=1}^N \hat{w}_{blind}[k] \cdot w[k] \quad (8)$$

- What can you say about the difference between blind and non-blind watermark detection in terms of the linear correlations ρ_{blind} and $\rho_{non-blind}$?