Multimedia Security and Privacy

TP4: Watermark Detection

Prof. Sviatoslav Voloshynovskiy,
Roman Chaban <roman.chaban@unige.ch>,
Yury Belousov <yury.belousov@unige.ch>.

Stochastic Information Processing Group

April 19, 2023

UNIVERSITÉ DE GENÈVE
FACULTÉ DES SCIENCES
Département d'informatique

SIP stochastic information processing

Submission

Please send your submissions in either PDF or Jupyter Notebook to "`Assignments/TP4: Watermark Detection`" on `https://moodle.unige.ch` before Wednesday, May 3rd 2023, 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 modelling

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 uniformly 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:

$$N = (N_1 \cdot N_2) \cdot \theta_N \tag{1}$$

  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 $\mathbf{y}$:

$$\mathbf{y} = \mathbf{x} + \mathbf{w} \tag{2}$$

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

$$\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \sigma^2_{noise}\mathbf{I}) \tag{3}$$

  where $\boldsymbol{\mu} = 0$ and $\sigma^2_{noise} = 1$.
- Add the AWGN matrix $\mathbf{z}$ to the watermarked image $\mathbf{y}$, resulting in attacked image $\mathbf{v}$:

$$\mathbf{v} = \mathbf{y} + \mathbf{z} = \mathbf{x} + \mathbf{w} + \mathbf{z} \tag{4}$$

- 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 of exercises you will attack a watermarked image leveraging the 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 $\mathbf{x}$:

$$\hat{\mathbf{w}}_{non-blind} = \mathbf{v} - \mathbf{x} \tag{5}$$

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

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

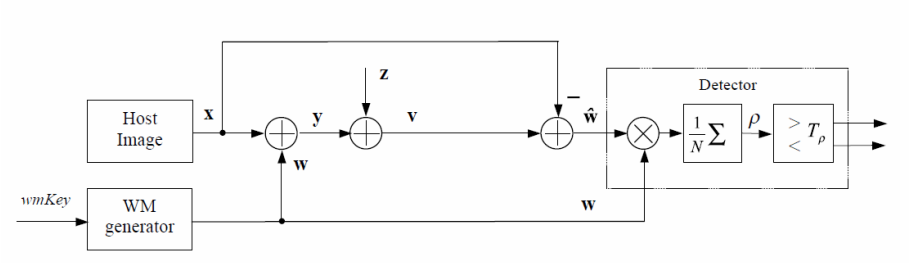

Figure 1 – Non-blind watermark detection

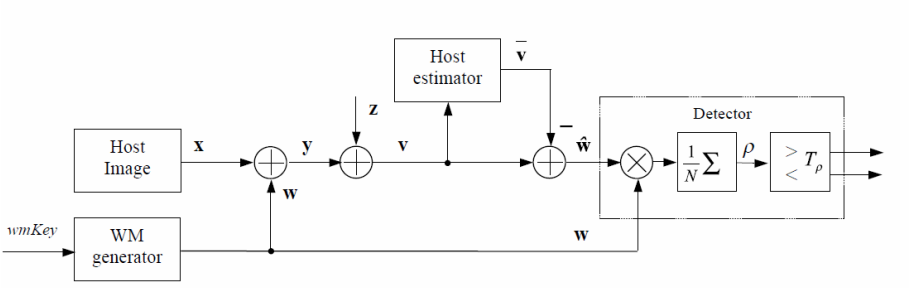

Figure 2 – Blind watermark detection

3 Blind watermark detection using the Maximum Likelihood estimate

This exercise will introduce a simple method to blindly estimate 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:

$$\hat{\mathbf{w}}_{blind} = \mathbf{v} - \bar{\mathbf{v}} \tag{7}$$

  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 dot product ρ_{blind} between the original watermark $\mathbf{w}$ and the blindly estimated watermark $\hat{\mathbf{w}}_{blind}$:

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

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