Statistical Steganography based on a Sensor Noise Model using the Processing Pipeline

Quentin Giboulot

Thèse
de doctorat
de l'UTT

Quentin GIBOULOT

Statistical Steganography based on a Sensor Noise Model using the Processing Pipeline

Champ disciplinaire :
Sciences pour l'Ingénieur

2022TROY0003 **Année 2022**

utt
UNIVERSITÉ DE TECHNOLOGIE
TROYES

THESE

pour l'obtention du grade de

DOCTEUR
de l'UNIVERSITE DE TECHNOLOGIE DE TROYES

en SCIENCES POUR L'INGENIEUR

Spécialité : OPTIMISATION ET SURETE DES SYSTEMES

présentée et soutenue par

Quentin GIBOULOT

le 14 mars 2022

Statistical Steganography based on a Sensor Noise Model using the Processing Pipeline

JURY

Mme Caroline FONTAINE	DIRECTRICE DE RECHERCHE CNRS	Présidente
M. Teddy FURON	CHARGE DE RECHERCHE INRIA - HDR	Rapporteur
M. William PUECH	PROFESSEUR DES UNIVERSITES	Rapporteur
M. Tomas PEVNY	RESEARCHER (DOCENT)	Examinateur
M. Patrick BAS	DIRECTEUR DE RECHERCHE CNRS	Directeur de thèse
M. Rémi COGRANNE	MAITRE DE CONFERENCES	Directeur de thèse

Personnalités invitées

M. Dirk BORGHYS	DOCTOR
Mme Jessica FRIDRICH	FULL PROFESSOR
Mme Véronique SERFATY	DOCTEURE

Remerciements

Cette thèse s'est déroulée au sein de l'Unité de Recherche $LIST3N$ – Computer Science and Digital Society – au sein de l'Université de Technologie de Troyes (UTT). Je tiens à remercier l'ensemble de l'équipe pour son accueil chaleureux et à l'ambiance propice au développement de mes travaux de recherche. Je souhaite remercier tout particulièrement nos deux secrétaires de pôles, Madame Bernadette André et Madame Véronique Banse pour leur aide constante et leurs conseils avisés. De façon plus sporadique, mais non moins importante, cette thèse est également le fruit d'un travail avec les membres de l'équipe Sigma au Centre de Recherche en Informatique, Signal, et Automatique de Lille (CRIStAL). Je garderai d'excellents souvenirs de ces séjours à Lille pour ses nombreuses et stimulantes discussions.

Vient le moment où il est temps de remercier ses directeurs de thèse. Seulement, je pense qu'il est important de souligner le fait que Monsieur Patrick Bas et Monsieur Rémi Cogranne auront été bien plus que des directeurs de thèse. Ils ont été présent dès mes premiers pas dans le monde de la recherche en 2017, m'ont emmené jusqu'à ma première conférence scientifique en 2018 et enfin jusqu'au manuscrit que vous tenez entre les mains, rédigé entre l'année 2021 et 2022, bouclant ainsi cinq années de travail ensemble. A tout le moins, ils auront été de véritables mentors, m'auront beaucoup appris, beaucoup guidé dans la compréhension du monde académique mais m'ont aussi fait confiance en me donnant une large autonomie sur ce travail de thèse tout en étant toujours présents dans les moments de besoin. Pour cela, je leur en suis d'une infinie reconnaissance.

I'll go back to English to express my deepest gratitude to Professor Jessica Fridrich who welcomed me at Binghamton University before the start of my thesis. She taught me a lot about the discipline which is the subject of this manuscript, but also about the scientific practice in general. I can say that she truly motivated me to pursue my career in academia and deeply thank her for that.

Je remercie également Monsieur William Puech, Professeur à l'Université de Montpellier ainsi que Monsieur Teddy Furon, Chargé de Recherche HDR INRIA à Rennes, d'avoir tous deux acceptés d'être les rapporteurs de cette thèse. De même, je remercie Madame Caroline Fontaine, Directrice de Recherche au CNRS ainsi que Tomas Pevny, Associate Professor à la Czech Technical University d'avoir examiné cette thèse.

Cette thèse ne serait évidemment pas ce qu'elle est sans le soutien constant d'un groupe d'amis. Merci à Paul pour cette vie partagée depuis bientôt cinq ans. Merci à Mathilde pour la joie constante qu'elle m'apporte chaque jour. Merci à Pierre-Louis, Tetem, Eloi, Clément, Salomé, Maya-Lune, Eddy, Audrey, Marius et d'autres encore qui se reconnaitront.

Thank you also to Yassine and Jan for their warm welcome back in Binghamton, I hope to continue working with you in the future.

Et bien que cela puisse sembler convenu, j'aimerai clore cette section, en remerciant avec la plus profonde sincérité mes parents, Dominique et Thierry ainsi que mon frère Joris pour ces vingt-sept (et vingt-trois) années passés ensemble.

Contents

Notations

General

smallcase	Scalar
smallcase/boldface	Vector
UPPERCASE/boldface	Matrix

Image representation

$\mathbf{x}$, $\mathbf{X}$	RAW image
$\mathbf{y}$, $\mathbf{Y}$	Precover (unquantized values)
$\mathbf{z}$, $\mathbf{Z}$	Cover
$\boldsymbol{\gamma}$	Prestego (unquantized values)
$\boldsymbol{\zeta}$	Stego
n	Number of elements (photo-site, pixels, DCT coefficients) in an image
m	Number of elements of blocks/macro-blocks of a specified size in an image
i	Used when indexing individual elements of an image
k	Used when indexing vectors of elements of an image

Distributions

$\mathcal{N}$	Univariate or multivariate Gaussian distribution
N	Quantized Gaussian distribution obtained with a uniform quantizer of step 1
$\mathcal{P}$	Poisson distribution

Functions

Φ	Cumulative distribution function of the standard Gaussian distribution
D_{KL}	KL-divergence
Q	Q-function or tail function of the standard Gaussian distribution

1 Basic concepts of steganography and steganalysis

The goal of this opening chapter is to present the main problem underlying steganography and steganalysis. We present the main concepts and provide an historic example of the game that takes place between the two main protagonists of this game: Alice, the steganographer, and Wendy the steganalyst. Chapter 2 and 3 introduce the discipline's main techniques and modern developments

Some researchers also like to call the steganalyst Eve in reference to the analogous antagonist in cryptography.

1.1 The prisoner's problem

The standard narrative which is used to introduce the symmetric problems of steganography and steganalysis is the prisoner's problem [73]:

Alice and Bob have just been arrested and placed in separate cells. Wendy, the warden of the prison, allows communication to occur between the two cells as long as it is performed on a channel where she can access its content. Furthermore, if she is able to infer any criminal intent on the part of either Alice and Bob, such as the communication of an escape plan for example, Wendy will immediately cut all communications between the two parties and put them into solitary confinement in order to prevent further escape attempts.

However, Alice does indeed have a plan of escape and wants to communicate it to Bob. It is assumed that they both had time to share a secret key, unbeknownst to Wendy, before incarceration. Alice's problem is now to find a way to communicate her plan to Bob without getting caught by Wendy.

1.2 Steganography as a solution to the prisoner's problem

A naive solution to Alice's problem would be to simply encrypt the message using the secret key and send the encrypted message to Bob on the public channel. However, such an encrypted message has, by design, no apparent structure and is consequently highly suspicious to Wendy: this not a acceptable solution to Alice's problem. This illustrates two majors points about the prisoner's problem:

- The goal is not to conceal the *content* of the message but to conceal the *act of communication* itself behind another, innocuous, act of communication.
- Corollary to the first item, there is a fundamental social aspect

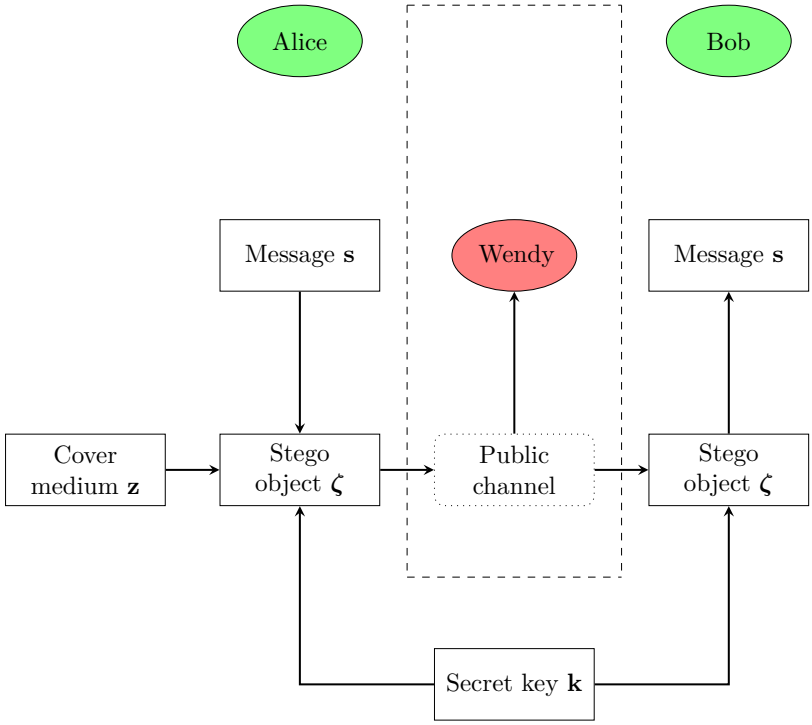


Figure 1.1: The prisoner's problem. The only way Alice can communicate to Bob is through a public channel under surveillance by Wendy. Alice must find a way to make her communication with Bob look innocuous to Wendy.

to the prisoner's problem since solving it necessitates to specify what kind of communication can be considered acceptable in a given context.

These points lead us to the definition of steganography:

Definition 1.2.1 (Steganography)**.** *Steganography is the discipline concerned with the design of techniques which allow to conceal a piece of information inside an innocuous piece of content which has to be communicated through a insecure channel.*

The piece of content chosen to hide the secret piece of information is usually referred to as the cover medium which we abbreviate as cover and denote using the letters $\mathbf{y}$ *(when the image takes values in the reals) and* $\mathbf{z}$ *(when the image takes values in the integers). Once it has been embedded with the secret piece of information, the cover becomes a steganographic object which we abbreviate as stego and denote using the letters* $\boldsymbol{\gamma}$ *(when taking values in the reals) and* $\boldsymbol{\zeta}$ *(when it takes values in the integers).*

There is a large choice of possible cover media in steganography, the only requirement being that it must not be suspicious in the context of interest. For example, sharing the result of a hundred coin tosses on social media might be deemed strange enough to warrant investigation whereas sharing digital images is a normal and accepted behavior in such an environment.

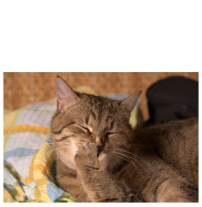
In this manuscript, the cover medium of choice are digital images since they are both plentiful and well accepted as a communication medium in most online interactions. The main problem to solve then

becomes the design of an algorithm which is able to encode a message in a digital image while, at the same time, making this act imperceptible to the Warden.

The notion of imperceptibility is quite nebulous for the moment. In order to develop a useful notion of imperceptibility, we first study the most basic steganographic technique which allows to hide information in digital images: LSB replacement.

1.3 An example of steganography: LSB replacement

At its most fundamental level, a digital image is nothing else but a vector $\mathbf{z}$, whose components encode the light intensity at each image location. In the case of color images, three such vectors of different colors are super-imposed to create the full color images, while a greyscale image uses only one such channel. To facilitate the discussion here, we will consider a greyscale image.



Color image

R, G, B channels

Greyscale image

These intensity values are usually integers ranging from zero to 2^8-1 or $2^{16}-1$ depending on the encoding choice. Now, each intensity value z_i can be written using a binary representation. For example a 8-bit representation of z_i uses a combination of bits $b_l \in \{0,1\}$ such that we can write:

$$z_i = \sum_{l=0}^{8} b_l 2^l \tag{1.3.1}$$

Numbers expressed using this binary system are usually represented as a sequence of 0 and 1 with the rightmost number (bit) representing the lowest power of two:

$$0_{10} = 00000000_2 \tag{1.3.2}$$
$$8_{10} = 00000100_2 \tag{1.3.3}$$
$$9_{10} = 00000101_2 \tag{1.3.4}$$

Any message can be encoded using this scheme, we thus assume that the stego message $\mathbf{s}$ to embed is also a binary sequence of zeros and ones.

The most immediate – and naive – method to hide information in a digital image is to choose randomly a series of indices $\mathbf{i}$ and to replace the rightmost bit of z_{i_j} by the j-th bit of the stego message s_j. This

results in the stego object $\boldsymbol{\zeta}$ constructed as:

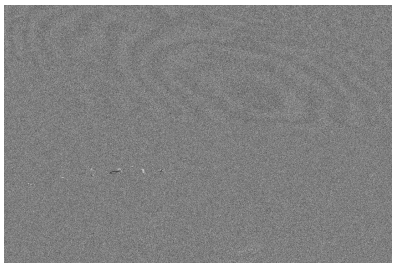
$$\zeta_{i_j} = \begin{cases} z_{i_j} + 1 & \text{if } b_{i_j,0} \neq s_j \text{ and } z_{i_j} \text{ is even} \\ z_{i_j} - 1 & \text{if } b_{i_j,0} \neq s_j \text{ and } z_{i_j} \text{ is odd} \\ z_{i_j} & \text{if } b_{i_j,0} = s_j \end{cases} \tag{1.3.5}$$

for $i \in \mathbf{i}$ and $\zeta_j = z_j, \forall j \notin \mathbf{i}$.

Once the message is embedded – see Algorithm 1 for a pseudo-code – the modified image is then simply sent to Bob through the insecure channel. Assuming that the key provided to Bob is simply $\mathbf{i}$, then Bob can extract the message from the image by reading the bits in the order provided by $\mathbf{i}$ – see Algorithm 2.

Taken at face value, this algorithm indeed solves the prisoner's problem. By modifying only the rightmost bit of randomly chosen values, we ensure that the modifications introduced to the cover are almost invisible to the naked eye. Indeed, if the i-th intensity value of the image was a medium grey with value 127 then the stego value in the worst case is 128 which is a slightly lighter grey. The difference will likely not be discernible:

(a) Cover $\mathbf{z}$ (b) Stego $\boldsymbol{\zeta}$



(c) $\mathbf{z} - \boldsymbol{\zeta}$

Figure 1.2: Difference between a cover and its stego counterpart generated with LSB replacement with a message size equal to the number of pixels in the in the cover ($R = 1$ bpp). The visual difference between the cover and stego image is absent despite pretty clear patterns when computing the difference between the two images.

Now if our chosen notion of imperceptibility would be visual imperceptibility, this would be an acceptable solution to the prisoner's problem. However, this assumes that Wendy is not aware that Alice can use steganography to communicate her plans to Bob. This is not assumed to be the case in practice and as we will see right now by introducing steganalysis to the game, this renders LSB replacement almost useless.

Algorithm 1: LSB replacement – Embedding

Data: $\mathbf{z}$: Cover, $\mathbf{s} \in \{0,1\}^R$: Message to embed, $\mathbf{k}$: Stego key

Result: $\boldsymbol{\zeta}$: Stego object

```
// Initialize a pseudo-random number generator using
   the stego key as the seed
```

$n \leftarrow \text{size}(\mathbf{z})$;

$R \leftarrow \text{size}(\mathbf{s})$;

$\mathbf{i} \leftarrow \text{Perm}(n, \mathbf{k})$;

$\boldsymbol{\zeta} \leftarrow \mathbf{z}$;

for $j \leftarrow 1$ *to* R **do**

  $\zeta_{i_j} = z_{i_j} + s_j - (z_{i_j} \mod 2)$;

Algorithm 2: LSB replacement – Extraction

Data: $\boldsymbol{\zeta}$: Stego object, $\mathbf{k}$: Stego key

Result: $\mathbf{s}$: Extracted message

```
// Initialize a pseudo-random number generator using
   the stego key to obtain the same permutation as
   Alice
```

$n \leftarrow \text{size}(\mathbf{z})$;

$\mathbf{i} \leftarrow \text{Perm}(n, \mathbf{k})$;

for $j \leftarrow 1$ *to* n **do**

  $m_i = \zeta_{i_j} \mod 2$;

```
// The first R elements of m make the hidden
   message
```

1.4 Steganalysis as the adversarial counterpart of steganography

Since Wendy is aware of the existence of steganographic techniques, she will perform an analysis of every piece of content that goes through the communication channel. Her goal is to confirm, to the best of her ability, that no hidden messages are present inside these pieces of medium.

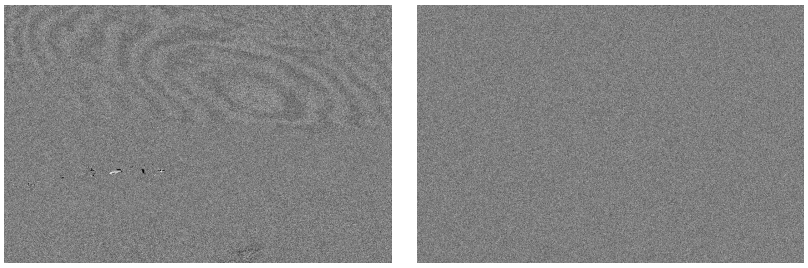
The problem that Wendy is trying to solve is studied by steganalysis, the converse discipline of steganography, which is concerned with the design of techniques able to detect hidden messages in cover media.

In practice, steganalysis tries to leverage artifacts introduced by the use of steganography. These artifacts are usually modeled as deviations from known properties of the distribution of elements of natural images, such as the distribution of the noise, distribution of neighboring pixels in smooth areas, etc... If we go back to our example of LSB replacement, Wendy can indeed exploit the presence of pixel distributions which are characteristics of this algorithm. To see why, it is useful to observe what is called the *Least-Significant Bit plane*, or LSB plane, of an image. It is the matrix built using the least-significant bit of each pixel value. Now observe, in Figure 1.3, that the LSB-bit

It is common to invoke Kerckhoffs's principle in steganography which states that the steganalyst should be aware of everything about the steganography except for the cover and the secret key. As we will see in the next two chapters, this principle is most of the time only partially followed by practionners for the evaluation of steganography. Indeed, in practice, steganography is often able to leverage information unavailable to the steganalyst to improve security. Two majors examples being the knowledge of the processing pipeline used to generate the cover and the rouding errors of DCT coefficients. See Section 2.4.

We will often make the semantic distinction between the steganalyst, which is the actual person performing the steganalysis and the steganalyzer which is the detector employed by the steganalyst to perform the steganalysis of a sample image.

plane of a cover image has a specific structure in certain areas which is completely lost for the stego object:



(a) Cover LSB plane (b) Stego LSB plane

Figure 1.3: Difference between the LSB plane of a cover and its stego counterpart generated with LSB replacement with a message size equal to the number of pixels in the cover ($R = 1$ bpp). Observe that all the structure of LSB in the sky section of the image is lost in the stego image.

The areas which have this specific structure in cover images are smooth and saturated areas where neighboring pixels have a high probability of having the same value, and hence the same LSB. Since LSB replacement is blind to such local structure and will randomly flip bits without regard to the value of neighboring pixels, it is easy in principle to discriminate the two.

To make this idea precise, we can follow [72] and construct the histogram of the values of the pixels over the cover and stego images. Let the histogram of the cover image be written as:

$$\mathbf{h}_j = \sum_{i=1}^{n} \delta\left(z_i - j\right), \tag{1.4.1}$$

where $\delta\left(.\right)$ is the Kronecker delta and n is the number of pixels in the image. Assuming the hidden message length is R bits, observe that a given pixel y_i stays the same after embedding if (1) it was not selected for modification, which happens with probability $1 - \frac{R}{n}$ or (2) it was selected for modification but the LSB of the pixel matched the secret message bit, which happens with probability $\frac{R}{2n}$. Consequently, we have, for any j:

$$\mathbb{P}\left(\boldsymbol{\zeta}_i = j | z_i = j\right) = 1 - \frac{R}{2n}, \tag{1.4.2}$$

$$\mathbb{P}\left(\boldsymbol{\zeta}_i \neq j | z_i = j\right) = \frac{R}{2n}. \tag{1.4.3}$$

Using these probabilities, we can compute the expected value of the histogram of the stego object $\mathbf{h}^s$:

$$\mathbb{E}\left[\mathbf{h}^s_{2j}\right] = \left(1 - \frac{R}{2n}\right)\mathbf{h}_{2j} + \frac{R}{2n}\mathbf{h}_{2j+1}, \tag{1.4.4}$$

$$\mathbb{E}\left[\mathbf{h}^s_{2j+1}\right] = \left(1 - \frac{R}{2n}\right)\mathbf{h}_{2j+1} + \frac{R}{2n}\mathbf{h}_{2j}. \tag{1.4.5}$$

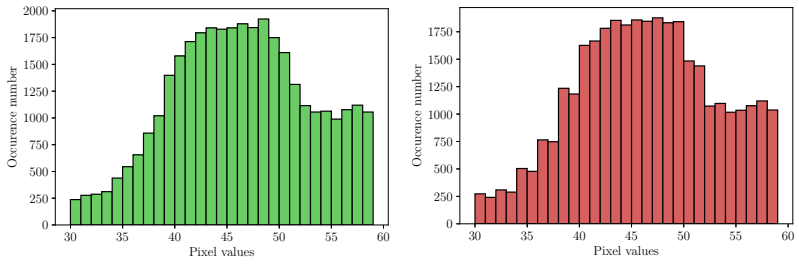
$$\tag{1.4.6}$$

In particular, in the limit where $R = n$, we have that:

$$\mathbb{E}\left[\mathbf{h}^s_{2j}\right] = \mathbb{E}\left[\mathbf{h}^s_{2j+1}\right] = \frac{\mathbf{h}_{2j+1} + \mathbf{h}_{2j}}{2}, \tag{1.4.7}$$

which can be interpreted as the fact that LSB replacement evens out the histograms of cover images – see Figure 1.4. This model of the

statistics of the stego object is exploited further in [28] to build an histogram attack which is able not only to detect the presence of an hidden message but also to quantify its length.



(a) Cover (b) Stego

Figure 1.4: Magnified portion of the histogram of pixel values for the cover and stego image of Figure 1.2. Observe the typical "staircase effect" on the stego histogram: consecutive pixel values tend to have a very close number of occurrence contrary to the cover histogram.

Many more attacks on LSB replacement have been devised during the history of steganalysis showing its inadequacy as a secure steganographic technique. However it is a perfect illustration of what a good steganographic technique must do to evade the steganalyst: (1) minimize the number of modifications to prevent deviating too strongly from the cover model and (2) perform modifications which preserves as much as possible the statistical properties of the cover medium. These two aspects of steganographic design are the main subject of the next chapter.

2 Elements of imperfect steganography

Imperfect steganography is a type of steganography performed by modifying an existing cover medium. For example LSB replacement is a type of imperfect steganography. This is opposed to perfect steganography which is based on either:

- selecting a cover which, incidentally, already contains the message to be communicated,
- generating a stego objects which perfectly imitates a class of cover while, at the same time, containing the message to be communicated.

Perfect steganography is quite difficult to perform for natural images in practice, mainly because it is difficult to define exactly what kind of statistical properties of images one should try to preserve in practice. A common argument in the literature, taken from in [7][Section 3.1.3], states that natural images are fundamentally incognisable, which prevents building a general model describing them.

A notable exception of successful perfect steganography is Natural Steganography [75]. Instead of trying to imitate a full image, it perfectly imitates *the noise of an image* at different ISO and thus allows undetectable embedding. Sadly, Natural steganography can only be used under a very specific context, namely if an image has been developed using a specific linear pipeline.

Due to this difficulty of designing practical algorithm in the perfect steganography paradigm, imperfect steganography has been the strategy of choice for most modern works in steganography using natural images. However, as we have seen, modifying cover elements without a sound strategy can only lead to the steganalyst easily detecting the stego object.

To solve this problem, all modern works[1] using imperfect steganography follow the same core methodology: minimizing a function d related to the empirical or theoretical detectability of the stego object when facing a certain type of steganalyst. This optimization being performed under the constraint that a message of a given size must be embedded in the chosen cover – a constraint which we term the *payload constraint*.

The performance of this strategy relies on two main pillars. The first is the definition of d, that is, how well d captures the actual detectability of the resulting stego object. Second is the coding method. Indeed, most messages can be encoded in order to reduce their size and thus reducing the number of embedding changes performed for a

[1] By modern, I refer to works following the landmark paper of Filler [25] which fully defined the framework presented in Section 2.1 on which most of steganography has been based since.

given payload. We will see in the next subsection that the design of d can be made independently of the coding function and vice-versa.

Before describing the different types of steganography techniques following this paradigm, we fix some notations for the rest of this section. The cover image is modeled as a vector $\mathbf{y} \in \mathbb{U}^n$ of n elements where $\mathbb{U}$ can be either a subset of $\mathbb{R}$ or $\mathbb{N}$. Similarly, the stego image is modeled as $\boldsymbol{\gamma} \in \mathbb{U}^n$. The payload constraint is expressed as a positive integers $R \in \mathbb{N}$ expressed in bits.

2.1 Steganography minimizing a heuristic cost

The first family of imperfect steganography we study is based on a heuristic definition of detectability. Its detectability function d is usually referred to as the distortion function and denoted as D and is a measure of discrepancy between the cover and a stego object created from this cover:

$$\begin{aligned} D : \mathbb{U}^n \times \mathbb{U}^n &\to \mathbb{R}^+ \\ (\mathbf{y}, \boldsymbol{\gamma}) &\mapsto D(\mathbf{y}, \boldsymbol{\gamma}) \end{aligned} \tag{2.1.1}$$

The goal of the steganographer is then to find a stego image that minimizes this quantity. However, D does not *apriori* have a direct link to either empirical or theoretical detectability since its definition is only heuristic. Therefore, embedding schemes employing this method must design and validate their choice of distortion function by testing the resulting stego objects against the best practical steganalyzer available. Consequently, such stego schemes can only be validated empirically.

Despite this limitation, numerous algorithms have been proposed in the literature, with the most important ones being HUGO [66], WOW [41],UED/UERD [39], UNIWARD [42] and HILL [56].

All these schemes have in common that their distortion functions rely on the additive approximation:

Definition 2.1.1 (Additive property of the distortion function)**.** *A distortion function D is said to be additive iff*

$$D(\mathbf{y}, \boldsymbol{\gamma}) = \sum_{i=1}^{n} D(\mathbf{y}, \mathbf{y} + \mathbf{e}_i (\boldsymbol{\gamma} - \mathbf{y})), \tag{2.1.2}$$

where $\mathbf{e}_i$ is a vector of size n containing only zeros except at index i where it contains a 1*.*

In other words an additive distortion function assumes that the distortion can be computed as the sum of distortions obtained when modifying a single element in the cover image. This assumes a steganalyzer which is unable to leverage interactions between embedding changes to improve its detection performance. Even though this model is blatantly incorrect in practice, it allows for tremendous simplification in the design of stego algorithm as well as the derivation of important theorems.

In particular, any additive distortion function can be thought of as associating to each cover element a cost $\rho_i^{(k)}$ of modifying this element by adding some value k:

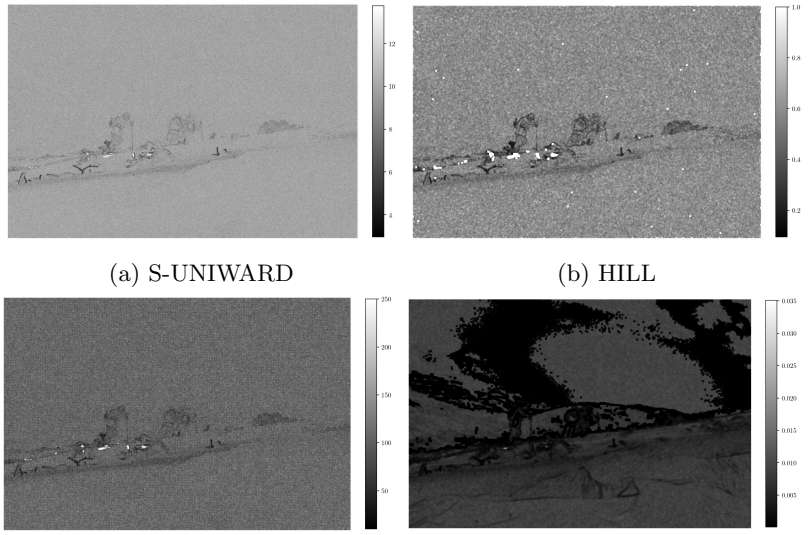
$$\rho_i^{(k)} = D\left(\mathbf{y}, \mathbf{y} + k\mathbf{e}_i\right), \tag{2.1.3}$$

with the convention that not modifying a cover element does not add any distortion:

$$\rho_i^{(0)} = 0, \forall i. \tag{2.1.4}$$

Consequently, the distortion between the cover and a stego can simply be rewritten as the sum of cost of modified cover elements:

$$D\left(\mathbf{y}, \boldsymbol{\gamma}\right) = \sum_{i=1}^{n} \rho_i^{(\gamma_i - y_i)}. \tag{2.1.5}$$



(a) S-UNIWARD (b) HILL

(c) J-UNIWARD (d) UERD

Figure 2.1: Different cost maps for the same image using different state of the art steganographic algorithms in the spatial domain (a-b) and in the JPEG domain (QF100) (c-d). Notice how all the algorithm favor embedding in areas with strong edges (i.e. low cost at the boundary of the skiers) and strongly disfavor smooth areas such as the snow.

Once a distortion function is defined, the steganographer must know how much information a given set of modification allows them to hide. This quantity depends on the coding method used but there exist a universal lower bound on this quantity [2] which depends on the entropy of the distribution of the modification:

Definition 2.1.2 (Binary entropy of a discrete source)**.** *Let* $\mathbf{X}$ *be a discrete random variable taking values in* $\mathcal{X}$*. Then its binary entropy, expressed in bits, is defined as:*

$$H\left(\mathbf{X}\right) = -\sum_{x \in \mathcal{X}} \mathbb{P}\left(\mathbf{X} = x\right) \log_2 \left(\mathbb{P}\left(\mathbf{X} = x\right)\right) \tag{2.1.6}$$

Theorem 2.1.3 (Optimal coding)**.** *The expected length* R *of any binary code for a random variable* $\mathbf{X}$ *is greater than or equal to the entropy:*

$$H\left(\mathbf{X}\right) \leq R \tag{2.1.7}$$

[2] Usually referred to as Shannon's bound coming from his well known Source Coding Theorem.

Note that we express this lower bound only for the discrete case, even though we assumed covers to be able to take values in $\mathbb{R}$ or $\mathbb{N}$. In practice, a digital image will always be shared using discrete values. As such the payload constraint must always be expressed in the discrete domain. However, as we will see in Part II of this manuscript, we can design algorithms which compute their detectability function in the continuous domain even though their payload constraint is expressed in the discrete domain.

Proof. See [17, Theorem 5.3.1] □

Consequently, even when using the best possible code, there exists no distribution of the embedding changes which allows to hide less than $H(\mathbf{X})$ bits of information.

Now if we assume that the steganographer has access to the optimal code, two possible strategies can be chosen;

Payload-Limited Sender (PLS) Finding the distribution π of embedding changes that minimizes the distortion under a given payload constraint R:

$$\text{PLS}: \begin{cases} \min\limits_{\pi} \mathbb{E}_{\pi}[D] & = \sum\limits_{\boldsymbol{\gamma} \in \mathbb{U}^n} \pi(\boldsymbol{\gamma}) D(\mathbf{y}, \boldsymbol{\gamma}) \\ R & = H(\pi) \end{cases} \tag{2.1.8}$$

Distortion-Limited Sender (DLS) Finding the distribution π of the embedding changes that maximizes the payload size under a given distortion constraint D^*:

$$\text{DLS}: \begin{cases} \max\limits_{\pi} & H(\pi) \\ D^* = & \mathbb{E}_{\pi}[D] \end{cases} \tag{2.1.9}$$

Both these problems can be solved using the technique of Lagrange multipliers which leads to the following fundamental theorem for steganography based on heuristic costs:

Theorem 2.1.4 (Optimal distribution and separation principle)**.** *Let $\mathbf{y} \in \mathbb{U}^n$ be a cover and $\boldsymbol{\gamma}(\mathbf{y})$ the set of possible stegos obtainable from this cover using a given embedding scheme. The solution π^* to the PLS and the DLS problem is given by:*

$$\pi^*(\boldsymbol{\gamma}) = \frac{e^{-\lambda D(\mathbf{y}, \boldsymbol{\gamma})}}{Z} \tag{2.1.10}$$

where Z is the normalization factor:

$$Z = \sum_{\boldsymbol{\gamma} \in \boldsymbol{\gamma}(\mathbf{y})} e^{-\lambda D(\mathbf{y}, \boldsymbol{\gamma})} \tag{2.1.11}$$

and λ is obtained from the payload or distortion constraint for the PLS an DLS problem respectively.

Proof. See [25, Section II] □

A direct consequence of this theorem is the *separation principle*: it provides the optimal distribution of the embedding changes of a stego with respect to (1) a given distortion function and (2) the optimal coding method. In practice, imagine we use a practical embedding scheme with a given distortion function and that we choose to use the optimal distribution π^*. Let us then measure the empirical average distortion $\mathbb{E}[D]$. If it is higher than the expected theoretical distortion, then we know the sub-optimality is caused by the coding method.

Such a point of view allows to separate the design of the distortion function, for which we can always obtain the optimal distribution of embedding changes, from the design of the coding method.

In a research setting, in order to evaluate the performance of a steganographic algorithm or of a steganalyzer we usually only *simulate* an optimal embedding by making changes over an image by following the optimal distribution of changes π^*. To do so, we use a useful corollary of Theorem 2.1.4 for additive distortion function:

A consequence of this corollary is that, in the case of an additive distortion, the embedding probabilities are determined independently of the others. Furthermore, the embedding modifications are sampled independently.

Corollary 2.1.5 (Optimal distribution for additive distortion function)**.** *Let D be an additive distortion function and $\mathcal{A}$ the set of possible embedding changes. Then the optimal distribution π^* for the PLS and DLS problem is the product of the marginal probabilities of each embedding change in $\mathcal{A}$:*

$$\pi^*(\boldsymbol{\gamma}) = \prod_{i=1}^{n} \frac{e^{-\lambda \rho_i^{\gamma_i - y_i}}}{\sum_{k \in \mathcal{A}} e^{-\lambda \rho_i^k}} \tag{2.1.12}$$

This corollary allows to compute the probability of making a given change at each pixel or DCT coefficient of an image using only the cost map, provided the distortion function is additive.

2.2 Almost-optimal coding for practical embedding

Even though we only use simulations of embedding schemes in this manuscript, it is important to understand how far current research is in practice from the optimal coding bound. We only present a short overview of the current state of the art method as a complete introduction would necessitate lengthy developments in coding theory which are out of the scope of this manuscript.

The landmark paper of Thomas filler in 2011 has proposed a coding technique, termed Syndrom-Treillis Code [24] or STC for short, which allows to find the stego object which contains a chosen message while also providing a trade-off between computational complexity and coding optimality. That is, in theory, the STC allows to get as close as one wants to the theoretical bound in Theorem 2.1.3.

The goal of the STC is to find $\boldsymbol{\gamma}$ such that the message $\mathbf{s}$ to be communicated is a *syndrome* of a parity-check matrix $\mathbf{P} \in \mathcal{A}^{l \times L}$ with $l < L$:

$$\mathbf{P}\,\boldsymbol{\gamma} = \mathbf{m}. \tag{2.2.1}$$

Since we have $l < L$, there exists multiple possible solutions to Eq (2.2.1), but following the separation principle, $\boldsymbol{\gamma}$ should be chosen to minimize the distortion of the cover image $\mathbf{y}$.

Now, to solve this problem, the STC uses the classical Viterbi algorithm – to find the best stego image $\mathbf{y}$ – combined with convolutional encoding. Assuming that the distortion function is additive, the STC uses a block-diagonal parity check matrix $\mathbf{P}$ built out of sub-matrices $\tilde{\mathbf{P}} \in \mathcal{A}^{h \times w}$. The choice of w is directly constrained by L and l. On

the other hand, h – termed the constraint length – is freely chosen by the user.

An important result in convolutional codes is that the code approaches exponentially the optimal coding bound as the constraint length increases. However, the complexity of the decoding operation will also increase with the constraint length. Consequently, there is a trade-off between optimality of the coding scheme and computational complexity.

For a long time, the performance of the STC has been considered to be so good that research on the subject of coding method in steganography has, for all intent and purpose, stopped in favor of research on the design of cost functions. However, a recent series of papers by Kin-Cleaves [46, 47] has highlighted the importance of the coding loss of STC with respect to empirical detectability, showing that some gains are still possible and that this coding loss should be taken into account by the researcher when simulating embedding schemes.

2.3 Steganography minimizing statistical detectability

Despite the fact that heuristic methods in steganography led to embedding schemes with state-of-the-art performance against empirical steganalyzers, the approach is plagued by two fundamental issues.

First of all, there is no direct link between empirical or theoretical detectability and the distortion function. For example, two stego images with the same distortion against a given cover can have wildly different empirical detectability against a given steganalyzer. This lacks of interpretability of the distortion function makes it difficult to understand what design choice leads to better performance in the first place.

This leads us to the second, even more concerning issue. The performance of these embedding schemes is highly dependent on the experimental setting as was shown in [71]. In this paper, the authors show that when tested on different datasets and when using different steganalyzers, the ranking of tested steganographic schemes differed wildly. This is exacerbated by the fact that distortion function parameters are usually optimized with respect to a given experimental setting[3]. Therefore, heuristic algorithms cannot provide any guarantee of performance.

To solve these issues, another framework of imperfect steganography as been devised, namely steganographic schemes minimizing statistical detectability. This trend began with the design of the Multivariate Gaussian algorithm [29] which assumed cover elements to follow an independent but not-identically distributed Gaussian model [4]. The authors then minimized the KL-divergence between the cover distribution and the stego distribution. As we show in this section, and as was known at the time through the work of Cachin [11], the KL-divergence is directly related to the performance of some type of optimal detector. This work was subsequently refined up to the landmark paper [69] de-

[3] HILL is the paradigmatic steganographic algorithm of this strategy. It was designed purely heuristically by searching for the parameters of a given set of filters which maximized security against the Spatial Rich Model steganalyzer.

[4] Contrary to what the name suggests, no dependencies between cover elements were taken into account.

scribing the MiPOD algorithm. In this work, the steganalyst problem is cast as a statistical hypothesis testing problem and an optimal detector is derived that solves this problem. The steganographer's goal is then to minimize the power of this optimal detector while taking into account the payload constraint.

The main limitation of these works is that their state-of-the-art performance do not carry over to JPEG compressed image which are the images of interest to the steganographer due to their ubiquity. One of the main contributions of this thesis is to construct algorithms following this framework with state-of-the-art performance in the JPEG domain – see Chapter 9 and 10.

In the rest of this section, we formalize this framework and provide a series of theorems which will be heavily used in Part II of this manuscript where we present our steganographic algorithms. We begin by formulating the steganalyst decision problem as an hypothesis testing problem between two simple hypotheses and derive the optimal detector under this setting by following the classic Neyman-Pearson framework. We then derive some properties of this optimal detector which will be useful throughout this manuscript. Finally, we cast the problem of the steganographer as a constrained optimization problem where the goal is to minimize the power of the optimal detector under a given payload constraint.

2.3.1 *Simple hypothesis testing*

During the stego game, an image is presented to Wendy, the steganalyst, $\boldsymbol{\xi} \in \mathbb{U}^n$. She assumes that $\boldsymbol{\xi}$ is generated by a random variable with distribution $P_{\boldsymbol{\theta}}$ where $\boldsymbol{\theta} \in \Omega \subset \mathbb{R}^p$ is an unknown vector of p parameters.

Her problem is to decide if $\boldsymbol{\xi}$ is either a cover or a stego image. Therefore, she must choose between two hypotheses: $\mathcal{H}_0$, the image is cover or $\mathcal{H}_1$ it is a stego object. In other words, she has to build a *test* which discriminates between these two hypotheses. Let us first formalize these two concepts:

Definition 2.3.1 (Hypothesis)**.** *Let $\Omega_k \subset \Omega$. A parametric hypothesis $\mathcal{H}_k$ is a proposition such that when it is true, there exist a vector of parameter $\boldsymbol{\theta}_k \in \Omega_k$ such that:*

$$\boldsymbol{\xi} \sim P_{\boldsymbol{\theta}_k}. \tag{2.3.1}$$

If Ω_k is a singleton, that is, if under $\mathcal{H}_k$ $\boldsymbol{\xi}$ can only follow a unique distribution $P_{\boldsymbol{\theta}_k}$, then the hypothesis is said to be simple otherwise it said to be composite.

Definition 2.3.2 (Binary hypothesis test)**.** *Let $\mathcal{H}_0$ and $\mathcal{H}_1$ be two hypotheses. Then a binary hypothesis test is a surjective (and measurable) mapping δ such that:*

$$\delta : \mathbb{U} \to \{\mathcal{H}_0, \mathcal{H}_1\}. \tag{2.3.2}$$

Obviously, many such tests exist, but Wendy wants to build an *optimal test*. However, to determine optimality, we must first define

what quantities we will measure to determine the performance of a test.

The performance of an hypothesis test can be measured using the notion of error. In our case, a test makes an error when it classifies an image as stego even though it was really a cover – a false alarm – and when it fails to detect a stego object – a non-detection error.

Formally such errors are defined as follows:

Definition 2.3.3. *The error of the k-th kind of a test δ, happens when a test rejects hypothesis $\mathcal{H}_k$ when it is true. For simple hypothesis tests, the probability of error of the k-th kind, denoted as $\alpha_k(\delta)$, is defined as the probability that the test rejects $\mathcal{H}_k$ when $\mathcal{H}_k$ is the true hypothesis:*

$$\mathbb{P}_{\theta_k}\left[\delta(\boldsymbol{\xi}) \neq \mathcal{H}_k\right] = \alpha_k(\delta). \tag{2.3.3}$$

In this manuscript we use the following conventions: $\alpha_0(\delta)$ is named the probability of false alarm and sometimes will be written P_{FA}, $\alpha_1(\delta)$ is named the probability of non-detection and sometimes will be written P_{MD}. Finally, $\beta(\delta) \triangleq 1 - \alpha_1(\delta)$ is named the power of δ and sometimes will be written as P_D.

Wendy must now choose a criterion of optimality with regard to these two types of error. In a real-life situation, the cost of falsely accusing someone of misdeed is considered very high for Law Enforcement Agencies. That is, the cost of a false alarm far outweighs the cost of non-detection. Furthermore, the probability of false alarm should not only be small but also known with certainty and, if possible, chosen *apriori* by the agent. Assuming Wendy works under these principles, the optimality criterion can be formalized following one of the most classic frameworks in decision theory, the Neyman-Pearson framework:

Definition 2.3.4 (Neyman-Pearson criterion of optimality)**.** *Let $\alpha_0 \in (0,1)$ be a fixed false alarm probability and let Δ be the set of possible tests. Then the optimal test in the sense of Neyman-Pearson is the most powerful test in the set of test for which the false-alarm is bounded by α_0. Formally let this set be written as:*

$$\mathcal{K}_{\alpha_0} = \{\delta \in \Delta | \alpha_0(\delta) \leq \alpha_0\}. \tag{2.3.4}$$

Then the optimal test δ^* is such that:

$$\forall \delta \in \mathcal{K}_{\alpha_0}, \beta(\delta^*) \geq \beta(\delta). \tag{2.3.5}$$

This criterion leads to a systematic method for building the optimal test. This method is given by one of the most celebrated result in the field of hypothesis testing:

Theorem 2.3.5 (Neyman-Pearson Lemma)**.** *Let $P_{\boldsymbol{\theta}_0}$ and $P_{\boldsymbol{\theta}_1}$ be two distributions with density $f_{\boldsymbol{\theta}_0}$ and $f_{\boldsymbol{\theta}_1}$ respectively. Let $\alpha_0 \in (0,1)$ and $\mathcal{K}_{\alpha_0} = \{\delta \in \Delta | \alpha_0(\delta) \leq \alpha_0\}$. Let the likelihood ratio $\Lambda(\boldsymbol{\xi})$ be defined as:*

$$\Lambda(\boldsymbol{\xi}) \triangleq \frac{f_{\boldsymbol{\theta}_1}(\boldsymbol{\xi})}{f_{\boldsymbol{\theta}_0}(\boldsymbol{\xi})}. \tag{2.3.6}$$

Then the most powerful test δ^* which discriminates between $\mathcal{H}_0$ and $\mathcal{H}_1$ is the likelihood ratio test (LRT):

$$\delta^*(\boldsymbol{\xi}) = \begin{cases} \mathcal{H}_0 & \text{if } \Lambda(\boldsymbol{\xi}) < \tau \\ \mathcal{H}_1 & \text{if } \Lambda(\boldsymbol{\xi}) \geq \tau \end{cases} \tag{2.3.7}$$

where τ is a threshold such that:

$$\mathbb{P}_{\boldsymbol{\theta}_0}[\Lambda(\boldsymbol{\xi}) \geq \tau] = \alpha_0. \tag{2.3.8}$$

Proof. See the original paper [61, Section III] □

We now provide several useful results on the properties of the LRT that will be useful in the second part of this manuscript.

2.3.2 *Properties of the likelihood ratio test*

As a ratio of probability distribution functions, the LRT might lead to somewhat difficult mathematical manipulation. Thankfully, it is possible to transform the LRT into forms more amenable to mathematical analysis without losing any of its power by using the following result:

Proposition 2.3.6 (Invariance of LRT under bijective transformation)**.** *Let g be any bijective and strictly increasing function. Then under the same settings as Theorem 2.3.5, we have that:*

$$\delta_g^*(\boldsymbol{\xi}) = \begin{cases} \mathcal{H}_0 & \text{if } g(\Lambda(\boldsymbol{\xi})) < g(\tau) \\ \mathcal{H}_1 & \text{if } g(\Lambda(\boldsymbol{\xi})) \geq g(\tau) \end{cases} \tag{2.3.9}$$

has the same power as δ^*.

In particular, due to the heavy use of exponential families in statistical modeling, the log function is often used.

Another important property of the LRT is its direct link to the KL-divergence as shown by the following proposition:

Proposition 2.3.7 (LRT and KL-divergence)**.**

$$\mathbb{E}_{\mathcal{H}_0}[\log(\Lambda(\boldsymbol{\xi}))] = -\mathrm{D}_{KL}(f_{\boldsymbol{\theta}_0} \,||\, f_{\boldsymbol{\theta}_1}) \tag{2.3.10}$$

$$\mathbb{E}_{\mathcal{H}_1}[\log(\Lambda(\boldsymbol{\xi}))] = \mathrm{D}_{KL}(f_{\boldsymbol{\theta}_1} \,||\, f_{\boldsymbol{\theta}_0}) \tag{2.3.11}$$

Proof. This result is trivially obtained by the following observation:

$$\begin{aligned} \mathbb{E}_{\mathcal{H}_0}[\log(\Lambda(\boldsymbol{\xi}))] &= \mathbb{E}_{\mathcal{H}_0}\left[\log\left(\frac{f_{\boldsymbol{\theta}_1}}{f_{\boldsymbol{\theta}_0}}\right)\right] \\ &= -\mathrm{D}_{KL}(f_{\boldsymbol{\theta}_0} \,||\, f_{\boldsymbol{\theta}_1}). \end{aligned} \tag{2.3.12}$$

And similarly for $\mathcal{H}_1$. □

Now, one is often interested in computing the exact power of the log-LRT as a function of the model's parameters. The exact distribution of the LRT under either hypothesis often don't make this a simple endeavor. However, it is often possible to invoke the central limit theorem to compute the power of the LRT in the asymptotic regime as the number of samples (usually DCT coefficients) tends towards infinity.

Theorem 2.3.8 (Linderberg's central limit theorem)**.** *Suppose* $(\xi_i)_{1\leq i\leq n}$ *is a sequence of independent random variables with finite expectation* $\mathbb{E}[\xi_i]$ *and finite variance* $\mathrm{Var}[\xi_i]$*. Denote* $s^2 = \lim_{n\to\infty}\sum_{i=1}^n \mathrm{Var}[\xi_i]$*. Then under Lindeberg's conditions:*

$$\forall \epsilon > 0, \lim_{n\to\infty} \frac{1}{s^2}\sum_{i=1}^{n} \mathbb{E}\left[(\xi_i - \mathbb{E}[\xi_i])^2 \cdot \mathbf{1}_{\{\xi_i : |\xi_i - \mathbb{E}[\xi_i]|^2 > \epsilon s^2\}}\right] = 0 \quad (2.3.13)$$

where $\mathbf{1}$ *is the indicator function.*

We have that:

$$\sum_{i=1}^{n} \xi_i - \mathbb{E}[\xi_i] \rightsquigarrow \mathcal{N}(0, s^2), \quad (2.3.14)$$

with $\rightsquigarrow$ *denoting convergence in distribution as* $n \to \infty$*.*

Proof. See [23][Section 3.4.2]. □

A direct corollary for the power of the LRT can be easily derived:

Corollary 2.3.9 (Asymptotic power of the LRT)**.** *Suppose* $(\Lambda(\xi_i))_{1\leq i\leq n}$ *is a sequence of independent random variables such that* $\Lambda(\xi_i))$ *is defined in the same way as in Theorem 2.3.5. Suppose each* $\Lambda(\xi_i)$ *has finite expectation* $\mathbb{E}_{\mathcal{H}_0}[\Lambda(\xi_i)]$ *and finite variance* $\mathrm{Var}_{\mathcal{H}_0}[\Lambda(\xi_i)]$ *under hypothesis* $\mathcal{H}_0$ *and similarly under* $\mathcal{H}_1$*. Then the asymptotic power of the LRT is given by:*

$$P_D \simeq Q\left(\frac{Q^{-1}(P_{FA})\sqrt{\sum_{i=1}^{n}\mathrm{Var}_{\mathcal{H}_0}[\log\Lambda(\xi_i)]} + \sum_{i=1}^{n}\mathbb{E}_{\mathcal{H}_0}[\log\Lambda(\xi_i)] - \sum_{i=1}^{n}\mathbb{E}_{\mathcal{H}_1}[\log\Lambda(\xi_i)]}{\sqrt{\sum_{i=1}^{n}\mathrm{Var}_{\mathcal{H}_1}[\log\Lambda(\xi_i)]}}\right) \quad (2.3.15)$$

Proof. We first express the threshold τ as a function of the P_{FA}:

$$\begin{aligned} P_{FA} &= \mathbb{P}[\delta^*(\boldsymbol{\xi}) = \mathcal{H}_1 \mid \mathcal{H}_0] \\ &= \mathbb{P}\left[\sum_{i=1}^{n}\log\Lambda(\xi_i) > \log\tau \mid \mathcal{H}_0\right] \\ &\simeq Q\left(\frac{\log\tau - \sum_{i=1}^{n}\mathbb{E}_{\mathcal{H}_0}[\log\Lambda(\xi_i)]}{\sqrt{\sum_{i=1}^{n}\mathrm{Var}_{\mathcal{H}_0}[\log\Lambda(\xi_i)]}}\right) \\ \iff \log\tau &= Q^{-1}(P_{FA})\sqrt{\sum_{i=1}^{n}\mathrm{Var}_{\mathcal{H}_0}[\log\Lambda(\xi_i)]} + \sum_{i=1}^{n}\mathbb{E}_{\mathcal{H}_0}[\log\Lambda(\xi_i)], \end{aligned} \quad (2.3.16)$$

where $\simeq$ here means that we use the asymptotic distribution of the LRT. We then plug it directly in the expression of P_D:

$$\begin{aligned} P_D &= \mathbb{P}[\delta^*(\boldsymbol{\xi}) = \mathcal{H}_1 \mid \mathcal{H}_1] \\ &= \mathbb{P}\left[\sum_{i=1}^{n}\log\Lambda(\xi_i) > \log\tau \mid \mathcal{H}_1\right] \\ &\simeq Q\left(\frac{\log\tau - \sum_{i=1}^{n}\mathbb{E}_{\mathcal{H}_1}[\log\Lambda(\xi_i)]}{\sqrt{\sum_{i=1}^{n}\mathrm{Var}_{\mathcal{H}_1}[\log\Lambda(\xi_i)]}}\right). \end{aligned} \quad (2.3.17)$$

□

Corollary 2.3.9 is quite useful as it allows to reduce the study of the LRT to the study of only its first two moments under each hypothesis which is further simplified by knowing that the expectation under each hypotheses are given by the KL-divergence between each of the pdf.

Sadly, it is not always possible to invoke the central limit theorem, as will be seen in Chapter 10. In these cases, we can still bound the power of the LRT by the KL-divergence using a simple data processing inequality:

Proposition 2.3.10 (Data processing inequality of hypothesis testing [11])**.** *Let $\mathcal{H}_0$ and $\mathcal{H}_1$ be two simple hypotheses with respective pdf of their distributions denoted as $f_{\boldsymbol{\theta}_0}$ and $f_{\boldsymbol{\theta}_1}$. Let δ be a test discriminating between $\mathcal{H}_0$ and $\mathcal{H}_1$. Also, let $\epsilon > 0$. Now, suppose that:*

$$\mathrm{D}_{\mathrm{KL}}\left(f_{\boldsymbol{\theta}_0} \,||\, f_{\boldsymbol{\theta}_1}\right) \leq \epsilon. \tag{2.3.18}$$

Then the following inequality holds:

$$\alpha_0(\delta) \log_2\left(\frac{\alpha_0(\delta)}{\beta(\delta)}\right) + (1-\alpha_0(\delta)) \log_2\left(\frac{1-\alpha_0(\delta)}{1-\beta(\delta)}\right) \leq \epsilon. \tag{2.3.19}$$

In particular, when the probability of false alarm is zero, that is, when $\alpha_0(\delta) = 0$:

$$\beta(\delta) \leq 1 - 2^{-\epsilon} \tag{2.3.20}$$

Proof. See [11][Section 3]. □

A particularly useful application of this inequality for steganography is the fact that minimizing the KL-divergence allows controlling the maximum power of a given test.

2.3.3 *Minimizing the power of the optimal detector*

With all these new elements now defined, let us go back to the steganographer, Alice, who wants to design a steganographic algorithm with security guarantees. To do so, she can leverage the results of the previous subsections. First of all, we assume again that the covers available to Alice follow a unique distribution $P_{\boldsymbol{\theta}_0}$. Alice also assumes the worst-case adversary against her steganography, that is, for each stego object Alice generates, she assumes that Wendy is aware of all the parameters relevant to the cover and to the generation of this stego object: the distribution of the cover image $P_{\boldsymbol{\theta}_0}$, the distribution of the stego signal $P_{\boldsymbol{\theta}_1}$ as well as the size of the payload R. Under these assumptions, Alice knows that the optimal detector Wendy can use is the LRT δ^* with power $\beta(\delta^*)$

The problem of the steganographer can thus be formalized as the following optimization problems:

$$\mathrm{PLS}: \begin{cases} \min_{\boldsymbol{\theta}_1} & \beta(\delta^*) \\ R = & H(f_{\boldsymbol{\theta}_1}) \end{cases} \tag{2.3.21}$$

$$\text{DeLS} : \begin{cases} \max_{\boldsymbol{\theta}_1} & H(f_{\boldsymbol{\theta}_1}) \\ D^* = & \beta\left(\delta^*\right) \end{cases} \tag{2.3.22}$$

We will see in Part II of this manuscript that these problems are often solvable using the technique of Lagrange multipliers.

2.4 Steganography using side-information

Up until now, we have considered that the steganalyst had access to the same information as the steganographer. This is hardly true in practice since the steganalyst often has only access to the final image and possibly the metadata contained within. The steganographer, on the other hand, can use RAW images taken with her own camera. They can consequently have access to the RAW image, the processing pipeline used to transform it into the final image and a lot of information between these two states which are unavailable to the steganalyst.

We call any such information, which is available to the steganographer yet not recoverable perfectly by the steganalyst, *side-information*.

Side-information can take many forms in steganography but the most popular form, and incidentally the one which is of most interest to us in this manuscript, is the access to the *JPEG cover image before the rounding operation*. Such an image, usually referred to as the *precover* and which we will denote as $\mathbf{y}$, can be used to compute the rounding errors $\mathbf{r}$:

A very creative use of side-information can be found in [19] where the authors use multiple JPEG images of the same scene to improve their steganography.

$$\mathbf{r} = \mathbf{y} - [\mathbf{y}] = \mathbf{y} - \mathbf{z}. \tag{2.4.1}$$

Now, how these rounding errors can be leveraged to improve steganography ?

First, one has to imagine a setup where the steganographer captures a RAW image $\mathbf{x}_1$ of a static scene. Now, imagine the steganographer captures a second image $\mathbf{x}_2$ in exactly the same conditions. These two images certainly have the same semantic content but they are not strictly identical because of numerous sources of noise introduced both by the quantum nature of light and by the camera sensor electronic components. Therefore, for a given set of acquisition parameters, a given static scene, and a given processing pipeline, there actually exists a whole distribution of possible cover images.

Chapter 5 discusses at length these different sources of noise and provides a statistical model.

There are at least two strategies to leverage this fact. Note that whatever the strategy chosen, side-information has been shown to provide considerable increase in steganographic security.

2.4.1 *Heuristic cost modulation*

The first strategy is intuitive and heuristic: unrounded DCT coefficients which are on average close to the boundary of an integer bin are more "unstable" in the sense that there is a high probability that the rounded value between $\mathbf{y}_1$ and $\mathbf{y}_2$ differ because of small differences due to the noise – see Figure 2.2 for a visual explanation of this phenomenon.

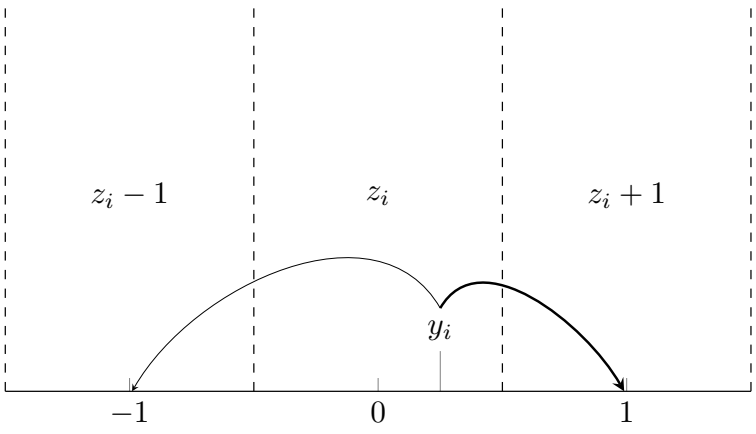


Figure 2.2: Without any more information on the model of the noise, we expect the unrounded DCT coefficient y_i to have a higher probability in falling into the bin closest to its value if the image was to be slightly perturbed. In the setting of this illustration, we expect that modifying the rounded DCT coefficient z_i by $+1$ will be less detectable than modifying it by -1.

A natural idea is then to improve cost-based steganography algorithm by modulating their cost depending on (1) the direction in which the DCT coefficient is most likely to be rounded to and (2) the distance of the DCT coefficient to the bin boundary. This direction led to a myriad of modulation strategies which usually fall into one of two categories: binary SI embedding and asymmetric-cost SI embedding.

Binary SI embedding This strategy effectively forbids modification which are not made in the direction of the rounding of the DCT coefficient while also favoring modifications of costs close to bin boundary. Assuming that the i-th DCT coefficient has a cost of modification ρ_i, a classic example of this strategy is given by the following modulation rule:

$$\begin{cases} \rho_i^{\operatorname{sign}(r_i)} & = (1 - 2|r_i|)\rho_i, \\ \rho_i^{-\operatorname{sign}(r_i)} & = \infty. \end{cases} \tag{2.4.2}$$

An example of this strategy can be found in [14].

Asymmetric-cost SI embedding In this strategy, no modification is *apriori* forbidden, the cost is just modulated differently depending on the direction of the modification. A classic example is the ternary embedding strategy which is defined by the following rule:

$$\begin{cases} \rho_i^{\operatorname{sign}(r_i)} & = (1 - 2|r_i|)\rho_i, \\ \rho_i^{-\operatorname{sign}(r_i)} & = \rho_i. \end{cases} \tag{2.4.3}$$

This is effectively the same rule as the binary embedding strategy except that $\rho_i^{-\operatorname{sign}(r_i)}$ is not modulated instead of being forbidden. Notice that the modulation is always less than 1,therefore the embedding modification in the direction of the rounding will always be preferred on average.

Further examples of this strategy can be found in [18, 42, 10] with special mention to the minimum perturbation strategy presented

in [10] which is the best performing heuristic known as of today.

2.4.2 *Preservation of the statistical distribution of the precover*

As is the case for the non side-informed case, the heuristic strategy is problematic in the sense that the modulation of its cost cannot be directly linked to empirical or theoretical detectability. A better way would then be to derive mathematically a strategy of modification which is directly linked to a statistical measure of detectability such as the power of an optimal detector under a given model or the KL-divergence between the cover and stego distribution.

Up until the work of this manuscript, this direction has seen very few proponents. The only work, to the best of our knowledge, which tackled this problem comprehensively is [21]. This work minimizes the KL-divergence between the cover and the stego image using a model of the rounding errors under a Gaussian model of the precover.

For technical reasons owing to the properties of the KL-divergence and to the choice of the rounding-error model, the authors had to limit themselves to a binary embedding strategy but found mathematically the following strategy of modulation:

$$\begin{cases} \rho_i^{\operatorname{sign}(r_i)} & = (1-2|r_i|)^2 \, \rho_i \\ \rho_i^{-\operatorname{sign}(r_i)} & = \infty \end{cases}. \tag{2.4.4}$$

This strategy led to very small empirical improvements compared to the still current state-of-the-art SI-UNIWARD which, by default, uses the ternary SI embedding strategy for modulating its costs, which might explain the lack of interest in strategies of this type.

2.5 Conclusion

In this chapter, we have presented the two main strategies in designing steganographic algorithms: the heuristic strategy and the statistically informed strategy. We have shown that the heuristic strategy, though quite successful in a laboratory setting, is limited in its real-world applications by its lack of a strong link to either empirical or theoretical detectability. The statistically informed strategy is, on the other hand, able to provide security guarantees at the cost of specifying both a cover and stego model that are accurate enough in a real-world setting. It also has the advantage of providing both a systematic design methodology and a theoretically sound way of assessing *apriori* the performance of a steganographic algorithm through the derivation of the optimal detector under the model of interest.

Up until the work presented in this manuscript, the main setback of this method which prevented its adoption in the JPEG domain was the lack of an accurate model of the noise of natural images. The goal of Part I of this manuscript is to derive such a model, starting from first principles, by studying how an image is captured and transformed into a JPEG image. The second part of this manuscript leverages this model in order to design steganographic algorithms which are able to

provide security guarantees for a class of images far larger than what was possible using the original MiPOD model in [69].

Finally, we have also shown how the steganography can leverage some of the knowledge which is unavailable to the steganalyst, in particular the knowledge of the unrounded DCT coefficients. This knowledge, termed side-information, if used correctly, allows tremendous gains in security for steganography. Once again, there exist several heuristic ways to utilize side-information, usually by modulating the cost map depending on the rounding errors. However, a competitive and statistically sound method had yet to be proposed.

The work in Part II of this manuscript tries to solve this problem by finding the optimal modulation strategy under a certain model of the cover and stego. This is done by minimizing the power of the optimal detector as described in Section 2.3, but this detector is expressed in the continuous domain, that is, in the domain of the precover, instead of minimizing it in the domain of the cover which is usually quantized. On the other hand, the payload constraint is expressed in the quantized domain. This choice of methodology automatically provides embedding probabilities that take into account the rounding errors without having to state explicitly a modulation rule – we refer the reader to Chapter 9 for more information.

We now go on to present the other side of the steganography game, the detection of hidden data in cover media.

3 Elements of modern steganalysis

As we have seen in Chapter 2, modern steganography, compared to the naive LSB replacement algorithm of Chapter 1, has developed more and more powerful techniques in order to reduce the number of embedding changes and to make those changes the least detectable possible. These developments began to take place during the year 2010 and continue today with the advent of powerful cost learning techniques based on neural networks [6]. As can be expected, the development of steganalysis mirrors this history and is fueled by the development of new steganography techniques.

The technique which marks the entry of steganalysis into its modern phase is the combined use of the SPAM [65] steganalysis features with the use of a Support Vector Machines (SVM). The SPAM features, developed in 2010, are based on Markov-chain model of dependencies between pixels and were the basis on which the HUGO steganographic algorithm was built. Though largely obsolete nowadays, the technique contains the two main elements that would be refined until 2016: (1) the use of a heuristic, high-dimensional model of pixel dependencies to handcraft features which are then fed to (2) a classifier trained in a supervised learning framework. This set of techniques is presented in Section 3.1.

The year 2015 marks the entry of steganalysis into a new phase as it begins to leverage the recent advances in computer vision to build the first convolutional neural network designed explicitly for steganalysis – the Gaussian Neuron CNN [68]. However, its poor performance compared the high-dimensional models of the time made it so that the deep-learning approach did not become popular before the advent of Xu-Net [84] in 2016 and its extension to the JPEG domain in 2017 [83]. The first iterations of Xu-Net merely incorporated the best practice of neural-network design: the use of batch-normalization [44] and of the ReLU activation function [59]. The extension to the JPEG domain added shortcut connections between layers to prevent the vanishing gradient problem, a common problem in deep networks where the gradient at one point of the network becomes so small that the network stops converging. However, Xu-Net retained one design peculiarity of the Gaussian neuron CNN, namely the use of a pre-processing layer. This layer consists of a series of high-pass filters reminiscent of those used to build the high-dimensional feature sets in steganalysis. They are meant to remove image content to facilitate detection of a stego-signal. This design choice was entirely abandoned by the next big step in steganalysis based on neural-network: SRNet [8]. The rationale behind SRNet was to have a fully automatic, universal architecture

for steganalysis where older architecture still relied on expert knowledge. Consequently SRNet is mainly based on a classic ResNet architecture [40] without any pre-processing layer. Its main design choice is to prevent any downsampling in the first layers of the network in order to retain as much information as possible on local variations in the image. This choice is motivated by the fact that contrary to classic problem in computer vision, steganalysis has to deal with the detection of very weak and localized signals. As of the writing of this manuscript, the latest development is a result of the steganalysis competition ALASKA2 [12] which took place in 2020. During the competition, most of the leading teams used the EfficientNet architecture [76] which, contrary to every other neural network architecture used in steganalysis until this time, was not specifically designed for steganalysis. The main novelty was the use of transfer learning. EfficientNet was first initialized with weights obtained by training it on ImageNet for other computer vision tasks. It was then refined by training it to discriminate between cover and stego images. This led to outstanding performance, outperforming SRNet. Further improvements were obtained by specific modifications to the architecture – see [86]. We present the deep learning methodology in steganalysis in more detail in Section 3.2.

3.1 Steganalysis based on handcrafted image features

A recurring theme in steganography and steganalysis is the difficulty of modeling images. In particular, for steganography, it is difficult to model the impact of several embedding changes on the statistics of an image. As we have seen in Section 2.1, this difficulty led to heuristic steganographic techniques to approximate their distortion function as additive, thus foregoing the modeling of dependencies between cover elements. Steganalysis based on handcrafted features leverages exactly this weakness of the steganographic schemes. The main idea behind the methodology is to build a model, not of the image itself which would be too difficult, but of the dependencies between neighboring cover elements once image content has been removed. As we will see in detail in Part I of this manuscript, these dependencies are introduced by the different steps of the processing pipeline used to go from the RAW file to the final developed image.

The details of the methodology differ if the image is in the spatial or the JPEG domain, but the main steps are as follows:

1. *Residual extraction*: the image is convolved by a series of high-pass filters to obtain the so-called *residual image* defined as the difference between a denoised and the original image. This is performed because the stego signal is a noise-like, weak signal hidden among image content. Detection should thus be easier once the content has been removed.

2. *Feature construction*: co-occurence matrices of different orders are built using the residuals with each bin corresponding to a single

feature.

3. *Classification*: a classifier is trained on pairs of cover-stego images to discriminate between these two classes.

We develop each of these steps in the rest of this section and provide examples.

3.1.1 *Residual extraction*

The extraction of residuals $\mathbf{R}_k$ of an image $\mathbf{Y}$ of size $h \times w$ is performed using a series of convolutions with different linear filters $\mathbf{K}_k$:

$$\mathbf{R}_k = \mathbf{Y} \star \mathbf{K}_k \tag{3.1.1}$$

Some feature sets also use some non-linear operations such as the minimum or the maximum. Here are some examples:

SRM The Spatial Rich model is a feature set built for the steganalysis of uncompressed images (spatial domain). It is built from 78 different submodels built from high-pass linear filters and combination of these filters, for example by selecting the maximum or minimum ouptput of two different filters.

DCTR The Discrete Cosine Transform Residuals is a feature set built for steganalyzing JPEG images. It uses a set of 64 filters corresponding to the 64 DCT kernels as defined in Eq (6.1.26). Each of these filters are then applied on the decompressed image of interest to obtain the residuals.

GFR The Gabor Filter Residuals is an improvement on the DCTR feature set which uses exactly the same algorithm except with different filters. Instead of using DCT kernels, GFR uses a bank of so-called Gabor filters [74] with different scales and orientations – see Figure 3.1.

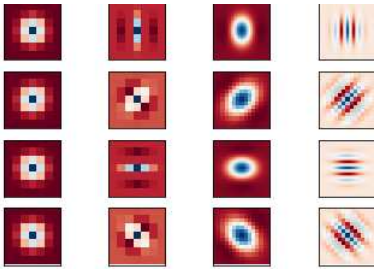


Figure 3.1: Bank of 2D-Gabor filters with different scales and orientation.

3.1.2 *Feature construction*

Once the residuals have been extracted, the feature set is constructed by computing co-occurrence matrices of these residuals. Such statistics are prone to have many under-populated bins for signals with a large dynamic range, consequently, the residuals are usually quantized and truncated before constructing these co-occurrence matrices:

$$\bar{\mathbf{R}}_k = \mathrm{Trunc}_T\left(\left[\frac{\mathbf{R}_k}{q}\right]\right) \tag{3.1.2}$$

with $q \in \mathbb{R}$, $T \in \mathbb{N}$ and

$$\mathrm{Trunc}_T(x) = \begin{cases} x & \text{if } -T < x < T, \\ -T & \text{if } x \leq -T \\ T & \text{if } x \geq T \end{cases}. \tag{3.1.3}$$

A matrix of co-occurrence $\mathbf{C}$ of n dimensions is then built by selecting two offset vectors $\boldsymbol{\delta}_x$ and $\boldsymbol{\delta}_y$, then computing the entry at the set of coordinates $\mathbf{i} \in \{-T, \dots, T\}^n$ as:

$$C_{\mathbf{i}} = \sum_{x=1}^{h} \sum_{y=1}^{w} \begin{cases} 1 & \text{if } \mathbf{R}_k\left(x + \boldsymbol{\delta}_x, y + \boldsymbol{\delta}_y\right) = \mathbf{i} \\ 0 & \text{otherwise} \end{cases} \tag{3.1.4}$$

leading to a co-occurrence matrix with $(2T+1)^n$ bins that correspond to the final feature set once repeated values corresponding to image symmetries are removed.

SRM The Spatial Rich Model uses four dimensional co-occurrence matrices with a horizontal and vertical neighborhood, that is, for each residual it computes two couples of vectors. For the horizontal neighborhood, it uses $\boldsymbol{\delta}_x = \begin{pmatrix} 0 & 1 & 2 & 3 \end{pmatrix}$ and $\boldsymbol{\delta}_y = \begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix}$ and vice-versa for the vertical one.

DCTR The feature set of DCTR only builds histogram for each DCT kernel and for each relative position of the pixel within an 8×8 DCT block. Some histograms are merged and symmetries are leveraged to further reduce the dimensionality of the feature set.

3.1.3 *Classification through supervised learning*

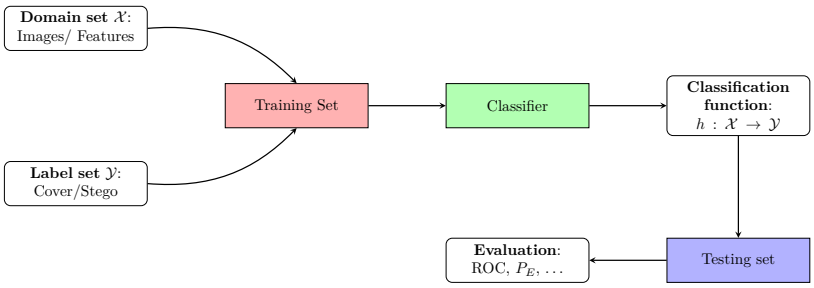


Figure 3.2: Diagram summarizing the process of supervised learning. A training set made out of labeled examples is fed to classifier which outputs a classification function which is then evaluated on a testing set containing samples not present in the training set.

Once a feature set is associated to each image, the steganalyst must devise a method to discriminate between feature sets coming from cover images and those coming from stego objects. This is performed in the framework of statistical supervised learning.

Following [72], the statistical supervised learning framework can be formulated as follows:

The detector's input The detector is assumed to have access to the following elements/knowledge:

- *Domain set:* The set of objects the detector wishes to label. In our case, the detector wishes to label images. These images might be provided as is, in which case the domain set is $\mathbb{U}^n$ or they can be provided as a feature set, for example, the rich models we have just studied, in which case the domain set is $\mathbb{R}^d$ where d is the dimension

of the feature set. To take into account all the possibilities, we refer to the domain set in this subsection as $\mathcal{X}$.

- *Label set* The set of possible labels or classes of the objects in the domain set. In our case, the detector wants to discriminate between cover and stego images. As such the label set is denoted as $\mathcal{Y}$ and is equal to $\{\text{Cover}, \text{Stego}\}$ or more simply and equivalently $\{0, 1\}$.

- *Training data* A finite sequence of n_{TRN} pairs of labeled objects $\text{TRN} = ((x_i, y_i))_{1 \leq i \leq n_{\text{TRN}}}$ where each pair belongs to $\mathcal{X} \times \mathcal{Y}$. In our case this would be a dataset of images which are known as cover and another datasets with images which are all stego objects. We will often refer to TRN as the training set.

The detector's ouput From its inputs, the detector has to learn a method of classification, that is, a function $h : \mathcal{X} \to \mathcal{Y}$ associating to each image a class, cover or stego. This function is meant to be used to label new data which does not come from the training set.

A data generation model In this framework, each class is assumed to be generated by a given probability distribution with pdf denoted as $\mathbb{P}$ over the domain set $\mathcal{X}$. We also assume that the data is generated such that it is possible to define the true labeling function $f : \mathcal{X} \to \mathcal{Y}$[1]. This function always outputs the label which corresponds to the true class of the image. Note that the detectors has access to neither the probability distribution nor to the true labeling function.

[1] Obviously this is too stringent of an assumption. The theory usually compares the learned function h to the Bayes Optimal Predictor, which is the detector providing the lowest possible classification error under the distribution which generates the data. See [72][Section 3.2.1] for more information.

Performance metrics during training The performance of the detector is theoretically measured by the probability of (classification) error $\mathbb{P}[h(x) \neq f(x)]$ This metric is usually referred to as the *generalization error* or *risk* in the literature. However, computing this error necessitates access to $\mathbb{P}$ and $f(x)$ which are not available to the detector. Instead, the detector can rely on the empirical risk, which is the rate of error of the detector on the training set:

$$L_{\text{TRN}}(h) = \frac{|\{1 \leq i \leq n_{\text{TRN}} | h(x_i) \neq f(x_i)\}|}{n_{\text{TRN}}} \tag{3.1.5}$$

Sadly, minimizing the empirical risk is known to lead to overfitting, that is, to a classifier overly specialized on the provided training examples but with poor performance on unseen data. To remedy this problem, the main idea is to restrict the set of learnable functions $\mathcal{H}$. Such a restriction is referred to as *inductive bias*. The detector then minimizes the empirical risk under this restricted set of functions. A classic example of choice for $\mathcal{H}$ is given by linear classifiers which only select functions which separate the domain set using an hyperplane. Another standard way of introducing an inductive bias is by the use of *regularization*.

Assessment of the generalization of the performance Once a detector has been constructed using the training dataset, its performance

must be evaluated on unseen data to evaluate its generalization performance. In an ideal world, the probability distribution and the true labeling function would be known by the steganalyst and the generalization could be computed exactly. Sadly this is not the case for steganalysis since even when a good model of certain aspects of an image is available, its parameters cannot be estimated perfectly.

Consequently, a standard set of metrics have been chosen along the years for steganalysis which are used to evaluate steganalysis performance [2].

[2] For the reasons given in Chapter 2, these metrics also serve to empirically evaluate steganography security.

These metrics are computed on a set of images not present in the training set, termed the *testing set* or *held-out set*. The labels of these images are known to the steganalyst but not provided to the detector. The performance of the detector can then be assessed in a variety of ways. Steganalysis usually uses metrics derived from the false-alarm rate $\hat{P}_{FA}$ (cover classified as stego) and the non-detection rate $\hat{P}_{MD}$ (stego classified as cover):

$$\hat{P}_{FA} = \frac{|\{1 \leq i \leq n_{\text{TST}} \mid h(x_i) = 1 \wedge f(x_i) = 0\}|}{2n_{\text{TST}}}, \tag{3.1.6}$$

$$\hat{P}_{MD} = \frac{|\{1 \leq i \leq n_{\text{TST}} \mid h(x_i) = 0 \wedge f(x_i) = 1\}|}{2n_{\text{TST}}}, \tag{3.1.7}$$

where n_{TST} is the number of images in the testing set and where we assume that they are as many stego images as cover images. Observe that we denote the false-alarm rate and the non-detection rate analogously to the probability of false alarm and the probability of non detection as defined in Definition 2.3.3. This choice has been made to highlight that $\hat{P}_{FA}$ is nothing else than an empirical estimation of P_{FA} of the detector under the true data-generation model (and respectively for P_{MD}).

The false alarm rate of a detector can often be controlled by a threshold chosen by the steganalyst, in which case, the Receiving-Operating characteristic curve, or ROC curve, can be computed by plotting the graph of $1 - \hat{P}_{MD}$ as a function of $\hat{P}_{FA}$ – see Figure 3.3.

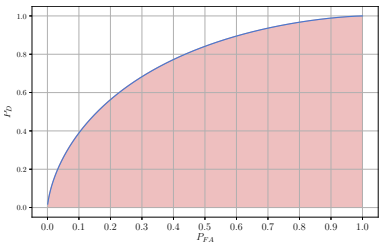


Figure 3.3: Example of a ROC curve in blue. The area in red correspond to the Area Under the Curve or AUC and can be used as a metric of the performance of a detector.

Two main quantities are often used in steganalysis:

- AUC : The area under the ROC curve, which is computed as:

$$\text{AUC} = \int_0^1 (1 - P_{MD}) dP_{FA} \tag{3.1.8}$$

  The closer to 1, the better the detector.

- P_E: The minimum probability of error under equal priors (of an image being cover or stego):

$$P_E = \min_{P_{FA}} \frac{1}{2} (P_{FA} + P_{MD}) \tag{3.1.9}$$

  The lower, the better the detector.

It should be noted that P_E is usually chosen as the metric of choice in steganalysis and in steganography though this might only be a question

of habit on the part of the community and the fact that it is easily computed. However, other quantities which will not be used in this manuscript but which are currently gaining some interest are the FP_{50} and MD_5 quantities. FP_{50} is the probability of detection at a false alarm rate of 50% and MD_5 the number of missed detection at 5% alarm rate. These quantities give a greater emphasis on the ability of a detector to perform well at low false alarm rates.

The classifiers of choice for steganalysis before the advent of neural networks were the Ensemble Classifier [49] and the Low-Complexity Linear Classifier [13]. The reason for the success of these two classifiers is a direct consequence of the peculiarities of steganalysis as a classification problem. Indeed, steganalysis usually operates on datasets of images containing between 10,000 and 100,000 images. Before neural networks, a high-dimensional feature set had to be extracted for each of these image, an already quite costly endeavor. The two aforementioned classifier both used "tricks" as to allow extremely fast training and testing steps. The Ensemble Classifier builds a set of simple classifiers which operates only on a subset of the dimensions of the feature set. The Low-Complexity Linear Classifier leverages cross-validation to find a regularization parameter which leads to essentially the same performance as the Ensemble Classifier.

3.2 Steganalysis using deep neural networks

Recent advances in image processing have shown the superiority of deep neural networks for many vision and classification tasks; steganalysis is not exception to this trend as was explained in the introduction. Since the focus of this manuscript is on steganography, this section only presents the bare minimum required to understand the steganalysis aspect of deep learning. For those readers interested to the subject, we direct to the standard reference on the subject [38].

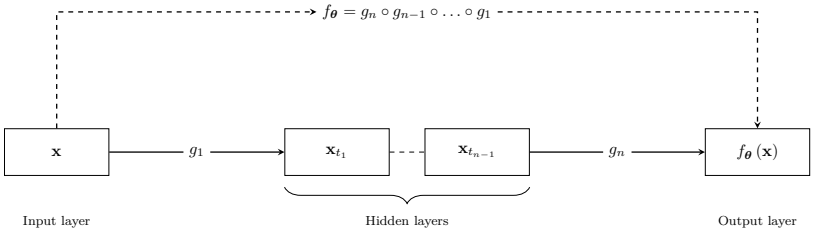
Classification for deep learning is no different than the supervised methodology presented in Section 3.1.3. However, contrary to methods based on handcrafted features, a deep learning architecture automatically learns to extract features that allow a classifier to minimize a given loss. The whole system of a deep learning network is based on three main ingredients: an architecture, a differentiable loss function and an optimization algorithm based on back-propagation.

3.2.1 *Architecture of a deep neural network*

At its core, a deep neural network is nothing but a series of parameterized functions applied on an input $\mathbf{x}$:

The architecture usually ends with a linear classifier, the role of the rest of the network being to learn a representation of the input which is easily separable for the linear classifier.

The modeling power of neural networks comes in part from the fact that each function is itself the composition of linear function h_i with a non-linear activation function σ_i:



$$g_i = \sigma_i \circ h_i \tag{3.2.1}$$

Most popular networks for image processing tasks are a special type of neural network, termed *convolutional neural network*.

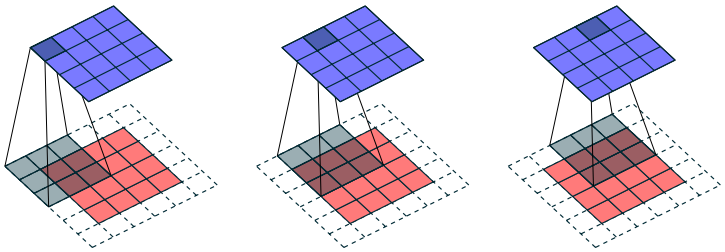
These networks are built out of two main types of operations: discrete convolutions and pooling operations.

Discrete convolution Convolution layers are made out of a learnable kernel of size $h \times w$, that is a matrix smaller than the actual input of the layer with weights which are iteratively adjusted during the optimization of the network. The kernel $\mathbf{K}$ is then convolved with the input $\mathbf{X}_i$ using a sliding window:

$$\mathbf{Z} = \mathbf{X} \star \mathbf{K}, \tag{3.2.2}$$

$$\mathbf{Z}_{i,j} = \sum_{k=i+1}^{i+h} \sum_{l=j+1}^{j+w} \mathbf{X}_{k,l} \mathbf{K}_{k-i,l-j} \tag{3.2.3}$$

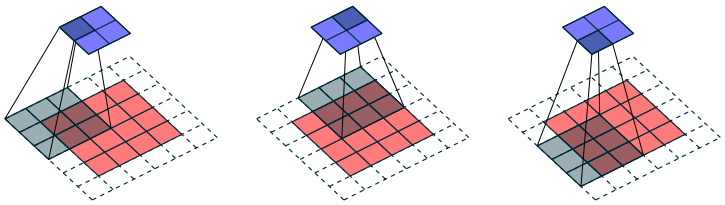
In deep neural network, these layers are usually further parametrized by a *stride* hyperparameter which controls how many elements are skipped by the kernel when it "slides" across the image during the convolution operation – see Figure 3.4-3.5. This allows to reduce the output size of the layer, effectively subsampling the input and consequently reducing the output.



(a) Step 1 (b) Step 2 (c) Step 3

Figure 3.4: Discrete convolution on input (red) of size 5×5, padded to obtain an output (blue) of size 5×5.

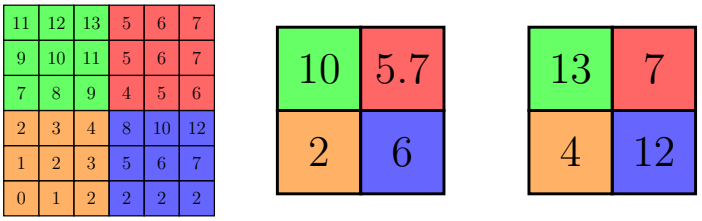
Pooling operation Pooling operations are used to reduce the dimension of a layer while keeping as much relevant information as possible. Similarly to the stride operations, pooling operations are important to reduce the amount of parameters to learn in a neural network, and hence the computational complexity of optimizing it. Two main types of pooling operations are used in practice: *average pooling* and *max*



(a) Step 1 (b) Step 2 (c) Step 3

Figure 3.5: Discrete convolution on input (red) of size 5×5, using a stride of 2 to obtain an output (blue) of size 2×2. Observe how every other element is skipped due to the stride.

pooling. Average pooling reads values on a uniform grid with window size $N \times N$ and outputs the average value of each $N \times N$ non-overlapping block – see Figure 3.6. Max pooling works in exactly the



(a) Input matrix (b) Average pooling (c) Max pooling

Figure 3.6: Results after different pooling operation. The input matrix (left) is separated into non-overlapping blocks by color. Average pooling outputs the average of each block and max pooling outputs the maximum value.

same fashion but only the maximum value of each block. These operations are usually fixed and contain no learnable parameters.

3.2.2 *Cross-entropy as a loss function*

Once a network architecture is fixed, a loss function must be chosen to evaluate the performance of the network on the training set and allow for an iterative optimization process. As we will see in the next subsection, the optimization of a neural network rests on the computation of a gradient with respect to the loss. Consequently, the chosen loss function must be differentiable.

By using the learning framework presented in Section 3.1.3, a natural loss function can be derived. First, recall that the goal of our neural network is to approximate, as well as possible, the true labeling function f. The main problem is that this function does not *apriori* have any structure which makes the problem quite difficult. An easier problem would be for our neural network to learn the data-generation model *conditioned on the current sample*. Such a problem is far simpler due to the fact that we are now restricted to the class of probability distribution functions parameterized by the parameters $\boldsymbol{\theta}$ of the network which we denote as $p_{\boldsymbol{\theta}}$. However, we don't have access to the true data generation model ; we only have access to a training set which has been generated by it. If we assume that the training set is a good representation of the true data-generation model, then we can settle with the network learning the empirical probability distribution of the data of the training set, which we denote as p_{data}.

A fairly standard approach to find a good set of values for $\boldsymbol{\theta}$ is to use the maximum-likelihood principle, that is, searching for the parameters which maximize the probability of observing the true label given the

sample. Assuming the samples are independent we can write:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n_{TRN}} p_{\boldsymbol{\theta}}\left(y_i \mid x_i\right) \tag{3.2.4}$$

$$= \arg\max_{\boldsymbol{\theta}} \frac{1}{n_{TRN}} \sum_{i=1}^{n_{TRN}} \log p_{\boldsymbol{\theta}}\left(y_i \mid x_i\right) \tag{3.2.5}$$

$$\simeq \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{p_{\text{data}}}\left[\log p_{\boldsymbol{\theta}}\left(x \mid y\right)\right], \tag{3.2.6}$$

where the last line assumes a large sample regime.

It is also easy to show that maximizing the likelihood in this setting is actually identical to minimizing the KL-divergence between $p_{\boldsymbol{\theta}}$ and p_{data}:

$$\begin{aligned} &\min_{\boldsymbol{\theta}} D_{\text{KL}}\left(p_{\text{data}}\left(x \mid y\right) \,||\, p_{\boldsymbol{\theta}}\left(x \mid y\right)\right) \\ &\quad = \min_{\boldsymbol{\theta}} \mathbb{E}_{p_{\text{data}}}\left[\log p_{\text{data}}\left(x \mid y\right) - \log p_{\boldsymbol{\theta}}\left(x \mid y\right)\right] \\ &\quad = \min_{\boldsymbol{\theta}} -\mathbb{E}_{p_{\text{data}}}\left[\log p_{\boldsymbol{\theta}}\left(x \mid y\right)\right] \\ &\quad = \max_{\boldsymbol{\theta}} \mathbb{E}_{p_{\text{data}}}\left[\log p_{\boldsymbol{\theta}}\left(x \mid y\right)\right] \end{aligned} \tag{3.2.7}$$

Consequently, if we want to follow the maximum likelihood principle, a natural loss function is the KL-divergence between the empirical distribution of the training set data and the probability distribution given as an output of the neural network. An equivalent loss function as is shown in Eq (3.2.7) is the *cross-entropy* $-\mathbb{E}_{p_{\text{data}}}\left[\log p_{\boldsymbol{\theta}}\right]$ which is differentiable and therefore usable as a loss function.

3.2.3 *Optimization*

Arguably, the most important part of the network, the optimization algorithm, allows to find the set of parameters for which the loss is minimal. Neural networks usually use a combination of stochastic gradient descent (SGD) combined with a backpropagation algorithm, allowing to process huge datasets in a relatively short amount of time.

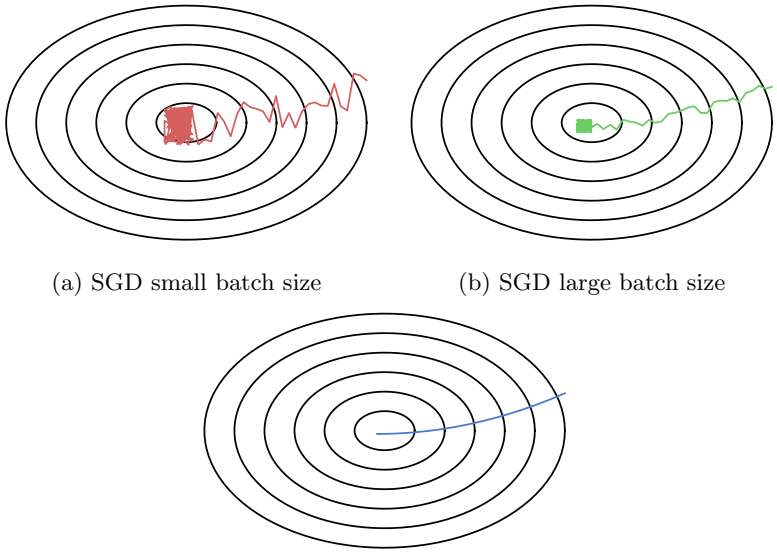
Stochastic gradient descent To find the values of the parameters θ which minimizes the cross-entropy, a neural network follows an iterative procedure where, at each step, each parameter is updated in the opposite direction of its gradient with respect to the loss:

This choice is due to the fact that this is the direction that leads to a local minimum of the loss function.

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}} l\left(\boldsymbol{\theta}_t, \mathbf{x}_t\right). \tag{3.2.8}$$

where $\mathbf{x}_t$ is the input to the network at step t and η is real hyperparameter usually called the *learning rate* for reasons we will see shortly. The specificity of stochastic gradient descent is that a random subset of the full training set $\mathbf{x}$ – called a mini-batch – is chosen as input to the network at each iteration instead of full training set. Since the gradient is only computed on this subset, we only get an estimate of the average value that would be obtained using the full training set. This

leads to different convergence behaviors of stochastic gradient descent depending on the size of the mini-batch:

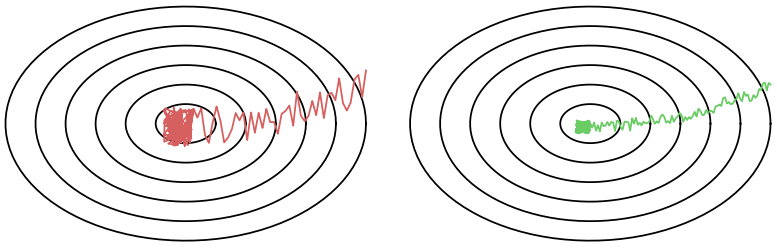


(a) SGD small batch size (b) SGD large batch size

(c) Deterministic gradient descent

Figure 3.7: This figure illustrates the behavior of stochastic gradient descent depending on the size of the mini-batch. The black curve corresponds to level set of the loss function, with the center being a local minimum. The colored curve represent the trajectory of the loss at each optimization step. As can be observed the smaller the mini-batch size, the noisier the trajectory of $\boldsymbol{\theta}$. Notice also that the smaller the mini-batch size, the larger the radius of the ball of values in which the value of $\boldsymbol{\theta}$ might end up.

At this point, the reader might wonder what is the point of using mini-batches instead of the full training set. The answer is that this considerably reduces computation time as the network only has to process a small part of training set. Furthermore, it is possible to improve SGD convergence without increasing the mini-batch size. Indeed, observe that if we divide the learning rate η by two, and perform two steps of SGD, we effectively performed one step of SGD using learning rate η except we "averaged" the loss over two different mini-batches, effectively reducing the noise in the gradient at the cost of more computation. Also note that using a noisy version of the gradient allows, to some extent, to escape local minima of the loss function.



(a) SGD large η (b) SGD small η

Figure 3.8: Illustration of the impact of the reduction of the learning rate which has a similar (but not equal !) effect than a larger batch-size.

Note that, in practice, more sophisticated variants of SGD than the one presented here are used, such as Adam [48] or Momentum SGD [36].

Backpropagation We now know how to use the gradient in order to minimize our loss function. However we still need to compute the gradient itself. Thankfully, the structure of a neural network allows the use of a simple and efficient algorithm to do so: the backpropagation algorithm [54].

The backpropagation algorithm leverages the chain rule of calculus in order to compute the gradient of the loss with respect to the parameters $\nabla_{\boldsymbol{\theta}} l\,(x)$.

In our setting, let $z_j = g_j \circ g_{j-1} \circ \ldots \circ g_1(x)$ be the output of the j-th layer for a given input x. Then the chain rule allows us to compute the i-th entry of the gradient $\frac{\partial l}{\partial \theta_i}$ as:

$$\frac{\partial l}{\partial \theta_i} = \sum_{j=1}^{n} \frac{\partial l}{\partial z_j} \frac{\partial z_j}{\partial \theta_i}. \tag{3.2.9}$$

This reduces the problem to computing $\frac{\partial z_j}{\theta_i}$ and $\frac{\partial l}{\partial z_j}$ which can be both be precomputed analytically or automatically and evaluated using the output of the network at various stages of the computation.

3.3 Conclusion

In this chapter, we presented the state of the art in steganalysis. It is based on different flavors of supervised learning. Up until 2016, steganalysis was dominated by the use of so-called rich model. They are high dimensional feature sets built out of high-order statistics on image residuals obtained by passing the image through a collection of high pass-filters. These feature sets were handcrafted to remove semantic content as much as possible and to capture neighborhood dependencies which were not taken into account by the steganography at the time. These feature sets are then used to train a – possibly linear – classifier which has to discriminate between features coming from cover and stego images. Starting from 2016, deep neural networks began to surpass the state of the art of steganalysis, be it in the spatial or JPEG domain. These networks remove the need for designing features sets by automating this step completely. This is done through a clever combination of the stochastic gradient descent and back propagation algorithms. This is at the cost of the need to design the architecture of such networks which is based on stacking different layers of linear functions and non-linear functions.

Impact of the image source in steganalysis

4

The story of this manuscript begins in the context of 2017. At that time the use of deep learning networks for steganalysis was only in its infancy and the use of rich models was still the norm. But what really defines this period for steganography and steganalysis is its high level of standardization regarding the evaluation of the performance of steganalyzers and steganographic schemes. During that time, all methods in steganography and steganalysis were evaluated using BossBase dataset built during the BOSS steganalysis competition of 2010 [5].

This dataset is composed of 10000 greyscale images taken with 7 different high-end reflex cameras and developed using the very same code from the original RAW files.

Now, such a dataset obviously does not reflect the diversity of natural images which can be found online. If one looks at mainstream social media such as *Facebook* as well as at platforms dedicated to photography such as *Flickr*, cameras can go from high end to very low quality sensors such as those found on smartphones and one can even find some exotic sensors such as those of the Fujifilm cameras. This diversity is absent from the BOSS dataset which only contain images taken from consumer-grade cameras – with the exception of the expensive Leica M9 which falls in the category of sensors with a peculiar behavior as shown in [20][Figure 2].

Regarding the standard processing pipeline of the BOSS dataset, it does not include many of the common operations usually performed on real-world images such as denoising, sharpening and it uses a demosaicking algorithm, PPG, which was already outdated by 2017 standards.

All in all, the world described by BossBase represents only a small subset of real world images, heavily biased toward the state of photography of the year 2010 and to the specific needs of a steganalysis competition. The reader, at this point, might rightly ask how this could have been a problem in practice when, even in 2017, a plethora of image datasets composed of a few million images were freely available to anyone who needed them, for example the popular ImageNet was made public as soon as 2009 [22]. As a matter of fact, the use of such datasets was not (and is still not) standard in steganography and steganalysis for reasons we will provide shortly. The resulting problem from the hegemony of a single standard, small and biased dataset was that steganographic algorithms were *designed* using their performance on Bossbase as feedback and steganalysis was consequently only evaluated on the dataset for which these algorithms were designed for. The generalization properties of this setting first strongly put into question

with works such as [71] which showed that when BossBase was developed differently from the standard processing pipeline, the ranking of different steganographic algorithms in terms of security could be completely flipped.

Why then datasets with such a low diversity were used? It mainly comes from the fact that, at least until the advent of deep learning, steganalysis was unable to cope with highly heterogenous datasets. In particular dealing with images of different size in the same dataset was costly and difficult and the routine extraction of high-dimensional features on million of diverse images was out of reach of most researchers in the discipline and the steganalysis performance on such datasets was quite poor.

It should nevertheless be acknowledged that such a standard dataset was also quite useful for benchmarking purposes.

However, since most steganographic schemes relied on steganalysis to evaluate their performance as we have explained in Chapter 2, a dataset where steganalysis was easy to perform and had reliable, known performance was needed. The causes of this lack of performance on real-world datasets were not fully understood at that time. However, it had already been linked to the phenomenon of *cover-source mismatch*, detailed in the next section, which is the starting point of the theoretical models of natural images of this thesis.

4.1 Heterogeneity of datasets and the phenomenon of the cover-source mismatch

One could be tempted to end this story with the advent of deep learning techniques and methodologies which allow to train a classifier on datasets containing more than a million sample with relative ease. This should intuitively allow us to capture the heterogeneity of the datasets and thus limit the generalization error.

Sadly, this is not the whole story, and to understand why, we must now turn to the phenomenon of cover-source mismatch in steganalysis.

This phenomenon was first documented in [37] where it was observed that training a classifier on a dataset containing images only taken with a given camera *CAM1* and testing it on a second dataset built only using another camera *CAM2* led to far worse performance than when the classifier was tested on a dataset built only with *CAM1*. This issue became even more evident during the BOSS competition where the organizers added images in the testing set which were taken with a camera not present in the training set. This led to a large drop in steganalysis performance on these very images. What is often less highlighted is that these outliers were not only taken with a unknown camera, but that they had all followed a double JPEG compression contrary to the other images which were simply JPEG compressed once. This shows that the processing pipeline might also play an important role on steganalysis performance.

The work in [50] studies this phenomenon by focusing mainly on the impact of different cameras, though it incidentally shows the greater impact of different processing pipelines on the effects of cover-source mismatch. The same authors also studied the effect of different resizing

algorithms on steganographic security, demonstrating the key role of the processing pipeline.

However, until our series of work [34, 35], there had been no systematic study of the impact of the different properties of natural images on cover-source mismatch. This lack of knowledge on the underlying causes of this phenomenon led to the impossibility of devising a theoretically sound method of constructing a training set that would at the same time limit the performance loss on unseen data while also limiting the number of samples needed. At the very least, it was important to understand on what type of images a classifier trained on a given dataset could be expected to perform well.

Our starting strategy to limit the impact of cover-source mismatch was to identify the *source* of images in given test set in order to build a bespoke training set. But this begs the question: what is the source of an image ? This is the question which we try to answer in the rest of this chapter.

4.2 What is a source ? A preliminary definition

The first step in defining the source of an image is to list and categorize all the parameters that play a role during the generation of a natural image. We claim that only three categories suffice to describe all these potential parameters:

Note that we will keep a birds-eye view of natural image generation in this chapter. The intricacies of each step of the process are analyzed in detail in the chapters of Part I of this manuscript.

- *Semantic content*: This refers to the scene represented by the image. More precisely, it is the signal transmitted and received as light by the camera's sensor.
- *Acquisition parameters*: This refers to all the parameters of the camera which are fixed in order to capture the signal of a given scene such as the camera sensor model, the ISO setting, the lens, the exposure time or the aperture.
- *Processing parameters*: This refers to all the parameters of every algorithms used to transform the RAW image captured by the camera into the final processed image.

This categorization leads us to the following definition of the source of an image:

Definition 4.2.1 (Source)**.** *A source can be defined as a set of acquisition parameters combined with a set of processing parameters that generate cover objects such that for a given semantic content, the succession of acquisitions forms a stationary signal.*

Note that this definition states that semantic content is not part of the source.

Obviously, such a definition is far too comprehensive to be of any use to the steganalyst in practice. However it provides a clear framework to the researcher for testing different parameters and measure their impact on steganalysis performance.

Our series of work [34, 35] tries to provide a systematic and empirical study of these parameters. We summarize (and update) the results

in Section 4.3. Before going on with the experimental study, we provide some definition to characterize the different types of difficulties introduced by a source for steganalysis.

Definition 4.2.2 (Intrinsic difficulty)**.** *The intrinsic difficulty can be defined practically as the prediction error (for example P_E, as defined in Eq (3.1.9)) for binary classification when both the training set and the testing set come from the same source. The more important the prediction error, the larger the difficulty.*

Definition 4.2.3 (Source inconsistency)**.** *The inconsistency between sources A and B is defined as the absolute difference between the prediction error when training and testing a classifier on a dataset source generated with source A and the prediction error when training a classifier on source B but testing it on a dataset generated with source A.*

4.3 Experimental exploration of the impact of different parameters on steganalysis

The choice of studying only the camera sensor and ISO setting for the acquisition parameters might be surprising. This choice, at first, stemmed from the fact that preliminary experiments showed not consistent patterns with regard to the impact on intrinsic difficulty or source inconsistency when fixing other acquisition parameters. The underlying cause of this fact is somewhat hidden by the choice of our categories.

Indeed, the choice of lens, exposure time and aperture, though a choice of acquisition parameter *to the photographer* actually affects the semantic content of the image as all these parameters control how much and how the light is captured by the camera. As we will see, the camera sensor and ISO are on the other hand related to the noise of the image which we claim to be the main source of cover-source mismatch in natural images.

4.3.1 *Experimental setting common to all experiments*

Several datasets are built by fixing different parameters. Each dataset is composed of 10,000 cropped JPEG cover images of dimension 264×264 and their 10,000 stego counterparts. A training set is built out of 70% of a given dataset, while the rest corresponds to the testing set.

For each experiment, we always produce a dataset composed of 10,000 images called MIX which is built by taking the same number of images in each individual datasets so that the class of images generated from a given set of parameter is balanced with each other class.

Steganalysis is always performed by using EfficientNet-b3 [76] in its original configuration with the exception that the stem stride is set to 1. The starting learning rate is set to 0.25 and divided by 2 on each loss plateau. Due to some datasets being more difficult than others, we performed curriculum learning by starting on images embedded

with a payload of 0.7 bits per DCT coefficients (bpc) followed by 0.5 bpc and finally 0.3 bpc. With the exception of an especially difficult processing pipeline, this ensured the convergence of the networks for each datasets.

Steganalysis performance is presented as a table of P_E – see Eq (3.1.9) – where the rows correspond to the training set and the columns to the testing set. The intrinsic difficulty of each dataset can be read on the diagonal of the table while source inconsistency is read by columns. To facilitate the reading of the numerous results, we subtracted the P_E of non-diagonal entry by the P_E of its corresponding diagonal entry: each non-diagonal entry thus directly refers to the source inconsistency between the training and testing set.

4.4 Acquisition parameters

4.4.1 *Camera and ISO*

To study the impact of the camera we selected five cameras of varying quality which are presented in Table 4.1. With the exception of the Canon EOS 500D, we have tried to keep the ISO relatively low, however one should understand that even if two cameras use the same ISO setting, the resulting noise will most certainly be different. As such, it is difficult to study the impact of the camera in isolation from the ISO. Therefore, the source inconsistency should here be interpreted as stemming from both the camera and ISO. All images were developed using Rawtherapee 5.8, in its default settings using the Amaze demosaicking algorithm. We chose to use the current state of the art steganographic algorithm for non side-informed JPEG steganography, J-UNIWARD [69]. Results are presented in Figure 4.1.

We explain the cause of this difficulty in Chapter 5 when constructing the model of the noise in the RAW domain.

Camera name	ISO	Year	Sensor size (mm)	Megapixels
Canon EOS 500D	1600	2009	22.3×14.9	15.1
Lumix DMC-GM1	200	2013	17.3×13.0	16
HTC One A9	93	2015	7.1 (total)	13
Apple iPad Pro	20	2015	4.80×3.60	12
Nikon D610	100	2013	35.9×24	24.3

Table 4.1: Characteristics of the different camera sensors used in the experiments of this chapter.

	MIX	Canon 500D	DMC-GM1	HTC A9	Ipad	Nikon D610
MIX	10.6	+3.6	+2.2	+2.9	+3.3	+0.9
Canon 500D	+12.7	16.8	+27.9	+18.9	+18.1	+13.0
DMC-GM1	+6.4	+15.2	6.0	+9.4	+5.1	+1.1
HTC A9	+8.4	+5.5	+10.1	10.5	+7.4	+9.0
Ipad	+9.2	+5.4	+4.5	+7.8	10.2	+3.5
Nikon D610	+19.8	+27.5	+16.1	+26.4	+23.7	1.6

Figure 4.1: Table of P_E for different cameras embedded with J-UNIWARD at payload 0.3bpp and steganalysis performed with EfficientNet-b3.

A first overall observation of these results is the large diversity of both intrinsic difficulty and source inconsistency even when the processing pipeline is fixed. It is difficult to attribute the intrinsic difficulty to either the camera or the ISO setting. For example, the iPad Pro and the HTC One A9, being handheld devices, have both low quality camera sensors and yet, despite the ISO of the HTC being higher than the ISO setting of the iPad, they both have the same intrinsic difficulty. On the other hand, the Canon EOS 500D, which has quite a high ISO setting compared to the other cameras, clearly has the highest intrinsic difficulty among these datasets. Consequently, we can conclude that both parameters play an important role here.

Regarding source inconsistency, it is almost always larger than 5% irrespective of the camera. An interesting case is that of the Nikon D610, the highest quality camera among those studied here. First it has the lowest intrinsic difficulty compared to the other cameras. This is most likely due to the quality of the sensor leading to almost noiseless images at low ISO. Secondly, it has a somewhat low source inconsistency with other sources but it leads to very high inconsistency with other datasets when it is used as the training set. This shows that even if a dataset has low source inconsistency with other sources, it might still be an extremely bad choice as a training set.

Finally, the MIX strategy seems excellent at mitigating the impact of cover-source mismatch in this case as it always leads to smallest source inconsistency when used as a training set.

4.4.2 *ISO*

The impact of the ISO is easier to study in isolation by fixing the camera. Therefore we used two in-house datasets termed M9Base1 and M9Base2 taken with a single Leica M9 camera. These dataset were made by photographing *exactly the same scenes* at different ISO which allows us to isolate the impact of the ISO from every other parameter. Once again, all images were developed using Rawtherapee 5.8, in its default settings using the Amaze demosaicking algorithm.

	MIX	ISO160	ISO320	ISO640
MIX	19.3	+2.3	+1.5	+5.0
ISO160	+1.8	10.5	+2.0	+8.9
ISO320	+2.9	+4.5	7.3	+8.0
ISO640	+0.5	+9.7	+4.5	18.7

Figure 4.2: Table of P_E for different ISO on M9Base1 embedded with J-UNIWARD at payload 0.3bpp and steganalysis performed with EfficientNet-b3.

As expected, the higher the ISO, the higher the intrinsic difficulty of the dataset. See for example in Table 4.1 where we go from an intrisinc difficulty of 10.5% at ISO160 up to 18.7% even though both the camera and the content of images are fixed.

	MIX	ISO500	ISO1000	ISO1250
MIX	17.5	+3.2	–0.9	+1.8
ISO500	+2.0	15.5	+2.1	+5.6
ISO1000	+3.8	+7.9	18.0	+2.0
ISO1250	+2.0	+8.1	–1.8	19.9

Figure 4.3: Table of P_E for different ISO on M9Base2 embedded with J-UNIWARD at payload 0.3bpp and steganalysis performed with EfficientNet-b3.

Also notice that semantic content does play a role here as images in M9Base2 taken at ISO1000 have an intrinsic difficulty similar to M9Base1 taken at ISO640.

In this setting, it looks like source inconsistency is always the lowest when using the MIX strategy, that is when training on a dataset where all the different ISO are present. However it should be noted that even in this case, source inconsistency can still be pretty high with values which can go up to 5% in the case of ISO640.

4.4.3 *Processing pipeline*

Finally, in order to study the impact of the processing pipeline, we fixed the camera and ISO of each dataset while performing different kinds of processing operations for each dataset.

Seven different processing pipelines using either Rawtherapee 5.8 (RT) or the *rawpy* library were chosen. In the case of Rawtherapee, each pipeline uses the default settings (using the Amaze demosaicking algorithm) while varying a single algorithm. In the case of *rawpy*, every setting is turned off, except for the white balance which is set in camera mode.

We now describe each of the pipeline:

- Three demosaicking algorithms: Amaze (RT), Bilinear (rawpy) and PPG (rawpy). The Amaze algorithm is one of the current state of the art among (open-source) demosaicking algorithms. The PPG algorithm is a simpler and faster algorithm which is used to generate BossBase. Finally, the bilinear algorithm (simplified as LIN) is the simplest and fastest non-trivial demosaicking algorithm possible, at the cost of large loss of quality compared to the other two.

- One sharpening algorithm from RT with two sets of parameters – USM soft and USM hard. The algorithm is a modified version of the classic Unsharp Mask algorithm [67] used to enhance the edges and contrast of an image. The first set only applies soft sharpening while the second applies very aggressive edge enhancement. The latter amplified the noise so much that our network usually did not converge due to resulting difficulty of the dataset. In theses cases, we omit to present the results.

- One denoising algorithm from RT with two sets of parameters –

DEN soft and DEN hard. The algorithm uses the Directional Pyramid Denoising based on wavelet decomposition [62].

	MIX	Amaze	LIN	PPG	USM soft	USM hard	DEN soft	DEN hard
MIX	10.0	+3.4	+1.0	+1.3	+1.5	*	+1.9	+0.6
Amaze	+14.4	10.2	+14.4	+24.2	+3.5	*	+8.8	+28.0
LIN	+31.6	+32.4	0.3	+37.3	+26.1	*	+34.4	+34.2
PPG	+19.6	+30.0	+15.8	2.7	+23.7	*	+30.5	+22.2
USM soft	+18.8	+6.7	+14.8	+34.6	19.5	*	+18.9	+33.0
USM hard	+39.8	+39.6	+49.0	+46.8	+30.3	*	+46.9	+48.8
DEN soft	+15.0	+15.5	+11.9	+30.7	+14.7	*	3.0	+17.4
DEN hard	+19.3	+33.8	+13.4	+34.5	+26.4	*	+10.0	0.6

Figure 4.4: Ipad Pro – ISO 20. Table of P_E for different processing pipelines embedded with J-UNIWARD at payload 0.3bpp and steganalysis performed with EfficientNet-b3. A column is starred (*) if Efficient-Net did not converge for the cell on the diagonal.

	MIX	Amaze	LIN	PPG	USM soft	USM hard	DEN soft	DEN hard
MIX	17.1	+6.0	+0.0	+3.8	+0.4	*	+4.5	+0.3
Amaze	+17.4	16.8	+34.1	+27.0	+7.8	*	+17.4	+13.1
LIN	+23.0	+33.2	0.1	+48.0	+13.9	*	+45.2	+34.2
PPG	+20.8	+30.9	+30.0	1.5	+13.8	*	+40.1	+38.1
USM soft	+29.5	+18.3	+47.3	+46.3	36.0	*	+45.9	+47.9
USM hard	+32.5	+33.0	+49.6	+47.6	+13.8	*	+47.4	+49.7
DEN soft	+17.9	+18.7	+13.7	+28.9	+8.7	*	2.3	+4.2
DEN hard	+17.0	+33.1	+5.6	+48.5	+13.9	*	+30.6	0.1

Figure 4.5: Canon EOS 500D – ISO 1600

	MIX	Amaze	LIN	PPG	USM soft	USM hard	DEN soft	DEN hard
MIX	9.3	+3.7	+2.4	+2.7	+3.7	+3.2	+3.1	+1.6
Amaze	+8.3	6.0	+18.4	+23.9	+2.7	+4.2	+6.2	+20.2
LIN	+24.8	+29.5	0.5	+29.7	+30.5	+23.7	+28.5	+19.1
PPG	+15.1	+24.3	+19.4	3.3	+25.2	+19.1	+16.3	+9.3
USM soft	+8.5	+2.6	+10.5	+27.9	11.0	+2.0	+8.2	+18.9
USM hard	+11.0	+3.6	+18.6	+31.3	+1.2	19.4	+9.1	+33.1
DEN soft	+7.9	+11.2	+8.5	+18.2	+13.3	+10.1	2.7	+3.1
DEN hard	+12.9	+26.9	+12.6	+13.4	+23.5	+18.7	+7.6	1.0

Figure 4.6: DMC-GM1 – ISO 200

A first overall observation is that the impact of the processing pipeline on Cover-Source Mismatch is dramatic, with source incoherency which can reach 49% even though the camera, the ISO and the scenes present in the datasets are identical. This observation was already made in [35] but it was hoped that using neural network would allow for better generalizations between sources which is clearly not the case in practice. A good news however is that using a training set which includes all the possible processing pipelines does allow for better generalization, even though we kept the number of samples identical for all datasets. This generalization results should however be studied more thoroughly as the diversity of processing pipeline "in the wild"

	MIX	Amaze	LIN	PPG	USM soft	USM hard	DEN soft	DEN hard
MIX	13.6	+3.4	+1.3	+2.4	+4.1	*	+1.8	+1.1
Amaze	+13.7	10.5	+36.5	+25.5	+9.2	*	+7.9	+21.0
LIN	+28.5	+31.2	0.1	+39.2	+25.0	*	+38.3	+31.5
PPG	+22.4	+33.3	+34.9	3.1	+25.3	*	+33.3	+28.0
USM soft	+10.9	+3.7	+11.2	+26.5	22.4	*	+9.9	+13.1
USM hard	+35.9	+39.3	+48.7	+45.8	+27.4	*	+46.6	+49.3
DEN soft	+18.8	+22.1	+30.0	+36.1	+19.7	*	3.0	+12.3
DEN hard	+17.4	+34.8	+16.0	+34.3	+25.4	*	+10.6	0.6

Figure 4.7: HTC One A9 – ISO 93

	MIX	Amaze	LIN	PPG	USM soft	USM hard	DEN soft	DEN hard
MIX	3.3	+2.1	+0.5	+0.6	+2.8	+2.3	+0.8	+0.8
Amaze	+1.7	1.6	+2.0	+4.1	+1.0	+1.4	+7.6	+1.1
LIN	+11.8	+20.0	0.2	+12.8	+20.9	+19.5	+5.7	+3.0
PPG	+2.5	+6.4	+0.9	0.5	+8.0	+8.0	+0.7	+0.9
USM soft	+0.7	+0.8	+1.5	+3.9	2.8	+1.4	+4.6	+3.4
USM hard	+1.6	+1.0	+2.5	+2.8	+0.8	5.7	+4.5	+3.5
DEN soft	+0.5	+2.5	+0.9	+1.7	+3.9	+3.7	1.4	+0.3
DEN hard	+3.6	+7.2	+2.6	+8.4	+6.2	+5.0	+1.7	0.6

Figure 4.8: NIKON D610 – ISO 100

can get quite difficult for the steganalyst to handle – this is a direction of future works not included in this manuscript.

Now, going to an analysis of each individual processing pipelines, we can observe that pipelines which amplifies details and edges – USM and the Amaze algorithm to a lesser extent – lead to higher intrinsic difficulties than pipeline which tend to smooth the image such as denoising and linear demosaicking. This is to be expected, the more textured an image is, the more difficult it is to model its content and thus to separate it from the stego signal.

In Chapter 3, we showed how steganalyzers try to separate the content from the noise in an image by using high-pass filters; edges and details are "high-frequency" content and as such will not be removed correctly using these methods.

It is also interesting to note that processing pipelines which are "closer" to each other in the sense that they perform the same operations but with different parameters tend to have lower source inconsistency. For example, in the case of the Nikon D610 camera, the two USM pipelines have a source inconsistency no higher than 1.5% but higher than at least 4% for every other pipeline except Amaze.

4.5 Conclusion : an empirical definition of the source

After collecting all these results, we are in a position where we can refine our definition of a source from Section 4.2.

First of all, we have seen that the camera and ISO have a strong impact on the phenomenon of cover-source mismatch and that even two cameras with the same ISO setting won't necessarily be coherent with each other. Secondly, and most important, is the impact of the processing pipeline. Two datasets taken with the same cameras with acquisition parameters fixed will be highly incoherent if they don't share the same processing pipeline.

Therefore, an empirical definition of a source could be the following:

Definition 4.5.1 (Source II)**.** *The source of an image is the conjunction of a camera sensor, an ISO setting and of a processing pipeline used to capture and produce the final processed image.*

We claim that as long as these three parameters are kept fixed, the phenomenon of cover-source mismatch should be mitigated or even absent.

There is still the case of the mismatch due to unseen semantic content which we studied only tangentially in this chapter. It is clear that a classifier trained only on, say, blue skies and tested on highly detailed images of forest for example might exhibit great loss of performance, something which we actually show to be true in [35, Section 8]. However, we purposefully avoided integrating semantic content in the definition of a source (and consequently all the acquisition parameters which only impact the semantic content) because measuring a meaningful distance between two different scene is a difficult if not impossible endeavor for steganalysis and steganography except maybe for extreme cases such as those alluded to earlier. Consequently, without a measure of similarity between scenes, we cannot produce reliable measure of source inconsistency since we cannot "fix" the scene parameter in the same way as we did for the camera, ISO and processing pipeline.

The next step of this manuscript is to justify theoretically this empirical definition of a source by starting from how an image is actually captured and processed.

Part I

Noise model of natural images

In the first part of this manuscript we build a statistical model of the sensor noise of natural images in the developed domain and provide methods to estimate its parameters. There are two main goals which we try to accomplish:

- First, we want to provide a theoretical justification of the empirical definition of the source of an image given in Chapter 4 where it was reduced to only three parameters of importance: the camera sensor, the ISO and the processing pipeline.
- Second, the model built in the first part of this manuscript must be suitable for the design of stenographic algorithm following the hypothesis testing framework presented in Section 2.3. In particular the model must be as general as possible. But it must also be simple enough so that we can compute or bound the power of an optimal test between a cover under this model and a simple model of stego objects.

The part is structured into three chapters going from the first principles in how light is captured by a camera sensor up to the final model after an image has been processed:

- *Chapter 5* presents the noise model in the RAW domain from first principles. We also provide a practical approximation of the model and methods to estimate its parameters for a given RAW image. This model of noise in the RAW domain is based on the classic work of Alessandro Foi on the Gaussian heteroscedastic model for signal-dependent noise [26, 27]. As such, the results of this chapter should not be considered as a contribution of this thesis but as a necessary foundation on which we rely to build the models of the next chapters.
- *Chapter 6* presents the model of the processing pipeline that will be used throughout this manuscript. It begins by presenting some of the classical operations in image processing before proposing a general linear and stationary model of the full processing pipeline. This chapter uses results from our work in [33, 32].
- *Chapter 7* presents the final model of the noise in the developed model by combining the results of the two previous chapters. It proposes a multivariate Gaussian model of the noise of macro-blocks of DCT coefficients where the covariance matrix depends only on the camera sensor, ISO setting and processing pipeline used to obtain the image. Two dependency models between macro-blocks are also proposed whereas their application to steganography is addressed in the second part of this manuscript.

5 RAW domain: Heteroscedastic model of the noise

5.1 Data-acquisition model of a natural image

The first step in deriving a model of the noise of natural images is to get an understanding of how images are acquired in the first place. In this work, we only touch upon linear imaging sensors, namely CCD and CMOS sensors, as they both make up for the vast majority of camera sensors at time of writing.

5.1.1 *Data-acquisition process*

The whole data-acquisition pipeline is summarized in Figure 5.1 alongside the different noise sources. Both sensor types function under the same principle: converting a light signal (photons) into an electrical signal (electrons). As such, they rely on a grid of photo-diodes, henceforth termed *photo-sites*, to detect photons arriving at different positions in the image. Each photo-site is able to capture only a single color, usually one among red, green or blue (RGB). This is due to a masking pattern arranged on the photo-sites that usually takes the form of the so-called Bayer grid pattern as illustrated in Figure 5.2.

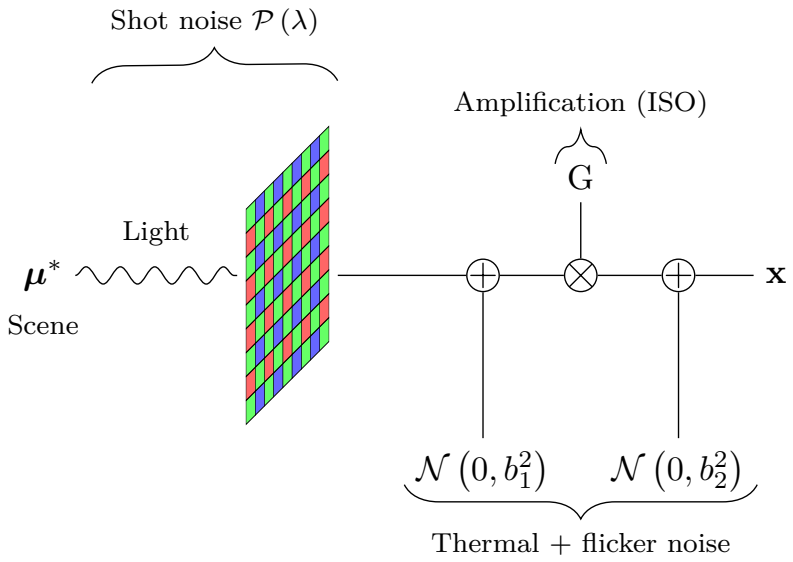


Figure 5.1: Diagram of the data-acquisition model. The light from the scene is captured by the photo-sites of the imaging sensor. The quantum nature of light introduces noise in the form of shot noise which follows a Poisson distribution. Further noise is introduced by the circuitry in the form of thermal and flicker noise. The electrical signal is then amplified by a value G controlled by the ISO setting of the camera.

Each photo-site will generate an electrical current. The average number of electrons generated by one photon is termed the *quantum*

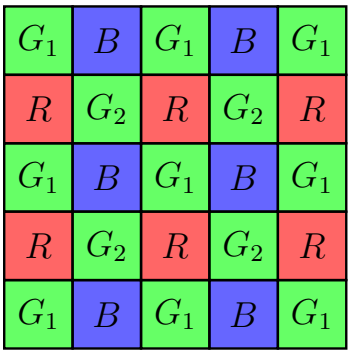


Figure 5.2: Bayer pattern used for most imaging sensors on the market. The pattern repeats over the full sensor, making each photo-site sensitive to only one of three colors.

efficiency χ and is dependent on the characteristics of the imaging sensor. Another hardware-dependent characteristic is the so-called "pedestal number": the collected charges of each photo-site is always offset by a positive value $p_0 \in \mathbb{R}^+$. This electrical current is then fed into different circuitry depending on the sensor type – see [60, 51] for references. At one point during this operation, the signal is amplified by a positive value $G > 1$ that is dependent on the ISO setting chosen by the user at the time of capturing the image.

During each step of this process, some noise is introduced to the signal. Two major sources of noise can be gathered from the description of the system: noise linked to the quantum nature of light as it arrives on the photo-sites (shot noise) and noise introduced by the different component of sensor circuitry (thermal and flicker noise).

5.1.2 *Shot noise*

The first main source of noise introduced during image acquisition is directly linked to the fact that each photo-site does not capture light continuously but only intermittently due to the discrete nature of light.

Let λ be the number of photons captured, on average, by a photo-site during an exposure time Δt for a given scene. Now divide Δt into $n > \lambda$ subintervals of size $\frac{\Delta t}{n}$. Let X_i be a random variable modeling a Bernoulli trial asking if at least one photon has been counted by the photo-site during the i-th time subinterval. Then we have:

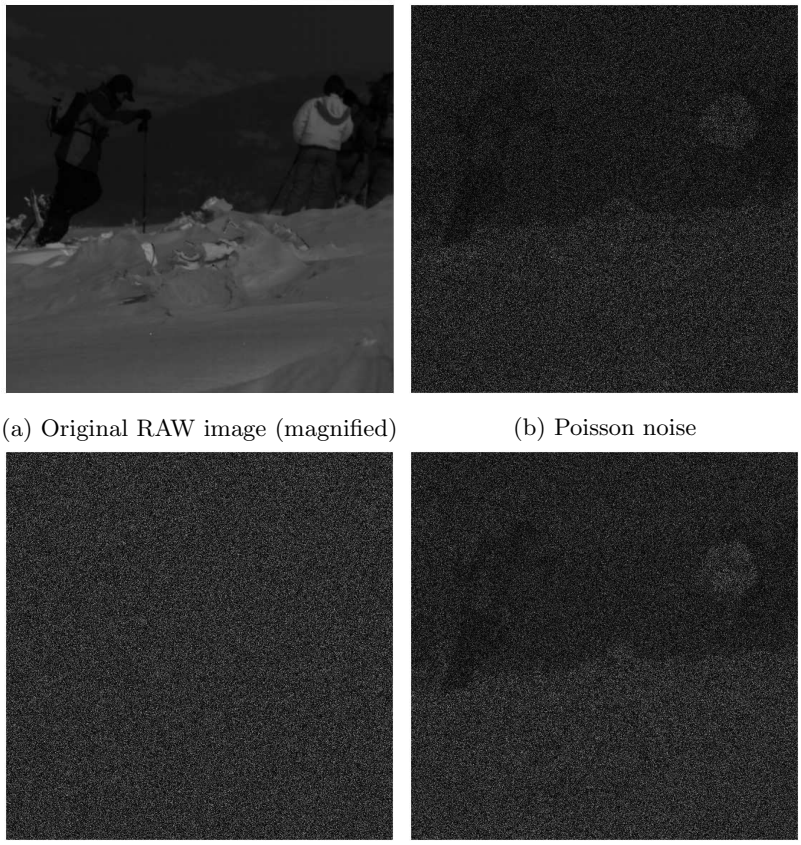
$$X_i \sim \mathcal{B}\left(1, \frac{\lambda}{n}\right), \tag{5.1.1}$$

and if we let $X_n = \sum_i^n X_i$ then we also have:

$$X_n \sim \mathcal{B}\left(n, \frac{\lambda}{n}\right). \tag{5.1.2}$$

Now, subdividing the interval into smaller and smaller subintervals, taking the limit as $n \to \infty$, we obviously have $n\frac{\lambda}{n} = \lambda$ converging to a finite value and $\lim_{n\to\infty} \frac{\lambda^2}{n} = 0$. Consequently, by LeCam's version of the Poisson limit theorem [52], we have:

$$X \triangleq \lim_{n\to\infty} X_n \rightsquigarrow \mathcal{P}(\lambda). \tag{5.1.3}$$



(a) Original RAW image (magnified) (b) Poisson noise

(c) Gaussian noise (d) Poissonian-Gaussian noise

Figure 5.3: Illustration of the different type of noise in the RAW domain. The dynamic range has been exaggerated for better visualization. Observe that Poisson noise gets stronger the more illuminated an area is. Gaussian noise on the other hand has a similar intensity over the whole the image. Poissonian-Gaussian noise is visually indistinguishable from Poisson noise as long as the image is not too dark.

That is the probability of counting k photons during a exposure time Δt is well approximated by the probability mass function of a Poisson distribution of parameter λ:

$$\mathbb{P}[X = k] = \frac{\lambda e^{\lambda}}{k!}. \tag{5.1.4}$$

Consequently, even if the hardware did not introduce any noise to the registered signal, this signal would still be different between two shots of the same scene taken with the same exposure time due to this phenomenon.

5.1.3 *Thermal noise and flicker noise*

.First of all, camera sensors are, as any electronic circuit, subject to *thermal noise*. This noise is introduced by thermal agitations of electrons in the circuitry, hence by every resistive parts of the circuit as well as by capacitors (in which case it is called ktC noise) Despite differences in the working mechanisms of CCD and CMOS sensors, the same general types of noise can be observed in both and won't require specific treatment. Note, however, that the noise sources will still differ for each sensor type. This is of no consequence for our analysis, but the interested reader can refer to [60, Chapter 3 Section 3] and [51, Chapter 3] for a thorough analysis of each circuit component and its impact on the noise.

Thermal noise is also known as Johnson-Nyquist noise, referring respectively to the discoverer and modeler of this type of noise.

Thermal noise is usually modelled as a Gaussian random variable owing to its direct theoretical link to the Brownian motion of particles in a fluid. However, it should be noted that Nyquist's paper [63] only proved the whiteness of the noise but not its Gaussianity.

Flicker noise, also called $1/f$ noise, is a type of noise that dominates thermal noise at low frequency during the amplification phase. Its exact mechanism has yet to be elucidated though it has been shown to be well modeled by Gaussian noise [81].

Since the sources for each type of these noises are numerous in a given circuit and assuming each of these sources to be independent from other sources, it is usual to invoke the central limit theorem [23, Theorem 3.4.5] to approximate the overall noise by a Gaussian random variable with zero mean and a certain variance. As the reader can observe in Figure 5.1, we distinguish between the noise added before and after amplification.

5.2 Poissonian-Gaussian noise model

As we have seen so far, all the noise sources for each photo-site can be modeled by either a Gaussian or Poisson random variable. Consequently, let x_i be the signal recorded at the i-th photo-site and let μ_i^* be the signal that would be received *on average* by the photo-site for a given exposure time Δt.

Then following the data-acquisition model presented in Section 5.1.1 and summarized in Figure 5.1 we first define $\eta_i^{g_1}$ and $\eta_i^{g_2}$ as the contribution of the thermal and flicker noise before and after amplification, respectively. On the other hand, we note η_i^p contribution due to Poisson (shot) noise:

$$\eta_i^{g_1} \sim \mathcal{N}\left(0, b_1^2\right), \tag{5.2.1}$$
$$\eta_i^{g_2} \sim \mathcal{N}\left(0, b_2^2\right), \tag{5.2.2}$$
$$\eta_i^{p} \sim \mathcal{P}\left(\mu_i^*\right). \tag{5.2.3}$$

Now following the data-acquisition process we can compute the output signal at the i-th photo-site x_i:

$$x_i = G\left(\chi\eta_i^p + p_0 + \eta^{g_1}\right) + \eta_i^{g_2}. \tag{5.2.4}$$

First notice that we multiply η_i^p by the quantum efficiency χ as even if μ_i^* photons are received on average by the photo-site, only a fraction of electrons will be produced as a result. Secondly, notice also that we add the offset p_0 to $\chi\eta_i^p$ as was described in Section 5.1.1.

Now, assuming a scene with sufficient lighting and a large enough quantum efficiency we can use the normal approximation of the Poisson distribution using the central limit theorem [23, Theorem 3.4.1] and write:

$$x_i \sim \mathcal{N}\left(G\chi\mu_i^* + Gp_0, G^2\chi^2\mu_i^*\right) + \mathcal{N}\left(0, G^2b_1^2 + b_2^2\right) \tag{5.2.5}$$

We can easily compute the first two moments of x_i:

$$\mathbb{E}\left[x_i\right] = G\chi\mu_i^* + Gp_0 \triangleq \mu_i \tag{5.2.6}$$
$$\mathrm{Var}\left[x_i\right] = G^2\chi^2\mu_i^* + G^2b_1^2 + b_2^2. \tag{5.2.7}$$

We consequently end with the classic Gaussian heteroscedastic model of the sensor noise:

$$x_i \sim \mathcal{N}\left(\mu_i, c_1\mu_i + c_2\right), \tag{5.2.8}$$

with:

$$c_1 = G\chi \tag{5.2.9}$$
$$c_2 = G^2 b_1^2 + b_2^2 - G^2\chi p_0, \tag{5.2.10}$$

Now, it is quite difficult to compute G, χ and p_0 when only having access to the RAW image. The goal of the next section is to solve this problem by giving a method to compute c_1 and c_2 without having access to the camera's parameters.

5.3 Estimation of the heteroscedastic model paramaters

5.3.1 *Introduction*

The estimation of the mean-variance curve for signal-dependent noise is a well-studied problem in the literature starting with the landmark paper of Foi [26] in 2008. Excellent reviews are available such as [53] or [3], to which we direct the reader for more detailed information as we will only give a brief summary of this literature here.

Two main strategies are usually applied to estimate the parameters c_1 and c_2 as defined in Eq (5.2.9):

Patch-based methods These methods are based on the assumption of self-similarity of images. The core idea is that there is a high probability that there exist several areas in a given image which share the same, or at least very similar, content. To leverage this property, the image is first subdivided into (possibly overlapping) patches of equal sizes. A distance function which measure similarity between patches is then used to group patches which are thought to have the same or close average signal. Using the fact that photo-sites with the same μ_i also have the same variance, the sample mean and sample variance of each group of patch are then computed to obtain an estimation of the mean-variance curve.

Segmentation-based methods These methods are based on the use of homogeneous areas in an image for easy estimation of the mean and variance. The main idea is to segment the image into homogeneous zones and remove areas with high variance which are difficult to model such as edges and highly textured areas. Once these areas are removed, values in an given interval $[x_i - \Delta, x_i + \Delta[$ are grouped together. The sample mean and sample variance of the values in each interval are then computed to obtain point-estimates of the mean-variance curve.

Following the construction of the mean-variance curve, each of these methods rely on a parametric curve fitting to find c_1 and c_2

For the purpose of this thesis, we have chosen to work with the original method presented in [26] as its performance is still extremely relevant as of today for the estimation of the mean-variance curve. However, we used the superior method present in [79] for the curve fitting step.

In the rest of this section, we give the algorithm for the estimation of c_1 and c_2 though we refrain from giving the full statistical analysis present in the original paper as it will not be useful for the rest of this work. We refer the interested reader to [26, Section III] for the detailed analysis. This algorithm belongs to the family of segmentation-based methods and is consequently composed of two steps: a local estimation of the mean-variance pairs followed by a parametric fitting of the mean-variance curve using Eq (5.2.8).

5.3.2 *Local estimation of the mean-variance pairs*

The estimation of the mean-variance pairs is itself divided into three steps. First, a wavelet analysis of the RAW image is performed in order to facilitate the estimation by separating the image into a smooth approximation component and a component containing most of the fine details of the image. Secondly, the image is segmented and grouped into level sets where each element belonging to a level set can be safely assumed to have an intensity close to other elements present in the same level set. Finally, a robust estimation of the mean-variance pairs is performed using these levels sets.

Wavelet analysis The wavelet analysis is performed using 1D Daubechies wavelet:

$$\boldsymbol{\psi}_1 = [0.035, 0.085, -0.135, -0.460, 0.807, -0.333] \tag{5.3.1}$$

$$\boldsymbol{\phi}_1 = [0.025, -0.060, -0.095, 0.325, 0.571, 0.235] \tag{5.3.2}$$

The RAW image is first separated into sub-images $\mathbf{X}_R$, $\mathbf{X}_B$, $\mathbf{X}_{G_1}$, $\mathbf{X}_{G_2}$ corresponding to each channel of the Bayer pattern as shown in Figure 5.2. These four sub-images are then concatenated into a single image $\mathbf{X}$.

The image $\mathbf{X}$ is then separated into two components using the Daubechies wavelets :

$$\mathbf{X}^{det} = \downarrow_2 \left(\mathbf{X} \star \boldsymbol{\psi}_1 \star \boldsymbol{\psi}_1^T \right) \tag{5.3.3}$$

$$\mathbf{X}^{app} = \downarrow_2 \left(\mathbf{X} \star \boldsymbol{\phi}_1 \star \boldsymbol{\phi}_1^T \right) \tag{5.3.4}$$

where $\downarrow_2$ is the decimation operator which removes every other photo-sites of the image. The main interest of this wavelet decomposition is that in uniform regions, the following approximation holds [26, Section 3]:

$$\mathrm{Var}\left[\mathbf{X}^{det}\right] \approx \mathrm{Var}\left[\mathbb{E}\left[\mathbf{X}^{app}\right]\right]. \tag{5.3.5}$$

Furthermore, it can safely be assumed that in uniform regions of the image, $\mathbf{X}^{det}$ mainly contains the noise of the image since, by definition, uniform regions don't contain edges or textures areas. From this observation, the main consequence of Eq (5.3.5) is that we can easily obtain the variance of noise if we can compute the expectation of $\mathbf{X}^{app}$ which is also simple in uniform regions for the same reasons we just invoked.

To estimate $\mathbb{E}[\mathbf{X}^{app}]$ we apply an 7×7 uniform kernel:

$$\mathbf{X}^{smo} = \mathbf{X}^{app} \star \mathbb{1}^{7\times 7}. \tag{5.3.6}$$

Now the approximation $\mathbf{X}^{smo} \approx \mathbb{E}[\mathbf{X}^{app}]$ only holds in uniform regions due to our choice of kernel, hence we have to remove all the regions of the image that are not uniform, this is exactly the goal of the segmentation step.

Segmentation The segmentation step is based on the classic methodology where edges are computed using an estimate of the local gradient of the images. If the local gradient is greater than the local standard deviation, it is then deemed to be an edge. The local standard deviation $\mathbf{S}$ of $\mathbf{X}^{det}$ is first estimated using:

$$\mathbf{S} = \sqrt{\frac{\pi}{2}} |\mathbf{X}^{det}| \star \mathbb{1}^{7\times 7}. \tag{5.3.7}$$

We then compute $\mathbf{X}^{edge}$ which corresponds to the estimation of the edges in the image as:

$$\mathbf{X}^{edge} = |\nabla\left(\nabla^2 \text{medfilt}\left(\mathbf{X}^{app}\right)\right)| + |\nabla^2 \text{medfilt}\left(\mathbf{X}^{app}\right)|, \tag{5.3.8}$$

where medfilt is 3×3 median filter while ∇ and ∇^2 are 9×9 gradient and Laplacian filters respectively. This form of estimation is a heuristic way to obtain "thick" edges allowing to keep only the most uniform regions.

Finally, the smooth photo-sites are obtained by thresholding Eq (5.3.8) against Eq (5.3.7):

$$\mathbf{x}^{smo} = \left\{ X^{edge}_{i,j} | X^{edge}_{i,j} < \tau S_{i,j} \right\}. \tag{5.3.9}$$

where $\tau \in \mathbb{R}^+$ is a fixed threshold chosen by the user.

Finally, we construct n level sets $\mathbf{l}$ containing the indices of the photo-sites in a given interval of intensity values:

$$\mathbf{l}_i = \left\{ k | x^{smo}_k \in \left[i\Delta - \frac{\Delta}{2}, i\Delta + \frac{\Delta}{2} \right[\right\}, \tag{5.3.10}$$

where Δ is the size of each level sets, also specified by the user.

Maximum-likelihood estimation of mean-variance pairs Now that we have constructed the level sets, the final step is simply to estimate the mean and variance of each level sets. Indeed, assuming that all observations from one level-set have the same mean value corrupted

by noise following the model in Eq (5.2.8), then each level set gives us a point estimate of the mean-variance curve.

For each level set, we use the maximum likelihood estimation of the mean and variance of Gaussian random variable:

$$\hat{\mu}_i = \frac{1}{n_i} \sum_{k=1}^{n_i} x_{l_{i,k}}^{smo}, \tag{5.3.11}$$

$$\hat{\sigma}_i^2 = \frac{1}{n_i - 1} \sum_{k=1}^{n_i} \left(x_{l_{i,k}}^{det} - \bar{x}_{\mathbf{l}_i}^{det} \right)^2, \tag{5.3.12}$$

with $\bar{x}_{\mathbf{l}_i}^{det} = \frac{1}{n_i} \sum_{k=1}^{n_i} x_{l_{i,k}}^{det}$ and where n_i is the number of elements in the i-th level set.

Note that $\hat{\mu}_i$ is unbiased and follows a Gaussian distribution:

$$\hat{\mu}_i \sim \mathcal{N}\left(\mu_i, \frac{\sigma_i^2}{n_i} \right), \tag{5.3.13}$$

while $\hat{\sigma}_i^2$, on the other hand, follows a gamma distribution, of which we will use the asymptotic Gaussian approximation assuming that n_i is large enough. This leads to:

$$\hat{\sigma}_i^2 \sim \mathcal{N}\left(\sigma_i^2, \frac{2\sigma_i^4}{n_i} \right). \tag{5.3.14}$$

5.3.3 *Parametric curve fitting*

Having access to an estimation of the mean-variance curve, we can now fit a parametric curve such that $\sigma_i^2 = c_1 \mu_i + c_2$ to obtain an estimate of the c_1 and c_2 parameters of the heteroscedastic model in Eq (5.2.8). To do so we will use the methodology presented in [79, Section 3] based on a weighted-least-square estimation.

The first part of the curve fitting step is to perform an ordinary least-square (OLS) fit of the data to the heteroscedastic model. Under the OLS model, it is assumed that:

$$\hat{\sigma}_i^2 = c_1 \hat{\mu}_i + c_2 + s_i \epsilon_i \tag{5.3.15}$$

where ϵ_i is a standard Gaussian noise and s_i depends on μ_i and controls the variance of the residuals.

A first estimation of c_1 and c_2, which we name $\hat{c}_1'$ and $\hat{c}_2'$ respectively, can be obtained by using the usual solution of the least-square problem:

$$\begin{pmatrix} \hat{c}_1' \\ \hat{c}_2' \end{pmatrix} = \left(\mathbf{M}^T \mathbf{M} \right)^{-1} \mathbf{M}^T \mathbf{\Sigma}, \tag{5.3.16}$$

with:

$$\mathbf{M} = \begin{pmatrix} \hat{\mu}_1 & 1 \\ \hat{\mu}_2 & 1 \\ \vdots & \vdots \\ \hat{\mu}_n & 1 \end{pmatrix} \; ; \; \mathbf{\Sigma} = \operatorname{diag}\left(\hat{\boldsymbol{\sigma}}^2 \right). \tag{5.3.17}$$

The variance of the residuals s_i^2 can readily be computed using Eq (5.3.13) and Eq (5.3.14) :

$$\begin{aligned} s_i^2 &= \mathrm{Var}\left[\hat{\sigma}_i^2\right] - \mathrm{Var}\left[c_1\hat{\mu}_i + c_2\right] \\ &= \frac{2}{n_i}\sigma_i^4 - \frac{c_1^2}{n_i}\sigma_i^2, \end{aligned} \tag{5.3.18}$$

where n_i is the number of sample in the i-th level set.

The strategy followed by the weighted-least square (WLS) estimator is to use the variance of the residuals as weights of the samples for another iteration of least-square estimation. The idea is that levels sets with larger residuals are less trustworthy than those with low residuals. Consequently, let $w_i = \frac{1}{s_i^2}$ and build the weight matrix $\mathbf{W} = \mathrm{diag}\,(\mathbf{W})$. The WLS estimates are then obtained by:

$$\begin{pmatrix} \hat{c}_1' \\ \hat{c}_2' \end{pmatrix} = \left(\mathbf{M}^T\mathbf{W}\mathbf{M}\right)^{-1}\mathbf{M}^T\mathbf{W}\boldsymbol{\Sigma}. \tag{5.3.19}$$

We provide some examples of mean-variance curve estimated with this method in Figure 5.4-5.6. Each curve was built by merging the three channels of 25 RAW images taken with the same camera and the same ISO into a single large image. Some sensors/ISO have a cleaner curve than others; this is mainly a question of how textured the images were to allow sufficient samples to be collected.

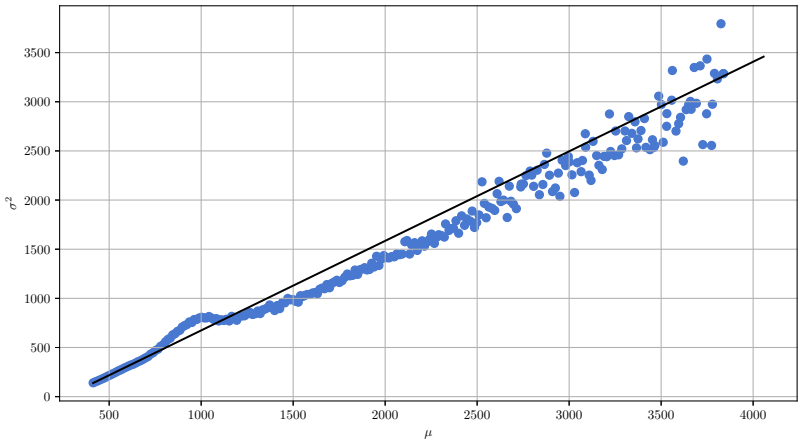


Figure 5.4: Mean-variance curve for the Canon EOS 400D camera at ISO 800. The blue dots correspond to the mean/variances sample while the black line is the estimated curve. Notice the small bump on the curve which is due to textured parts of the image which have not been filtered out.

5.4 Conclusion

In this chapter, based on the work of A. Foi [26, 27], we derived from first principles a model of the noise in the RAW domain. We explained how noise is introduced at each step of the RAW image generation: first through the quantum nature of light and secondly by the circuitry due to thermal agitation and amplification. This leads to two major types of noise: Poisson and Gaussian noise. After modeling the contribution of each of these types, we were able to provide an excellent approximation of the noise present on each photo-site as independent,

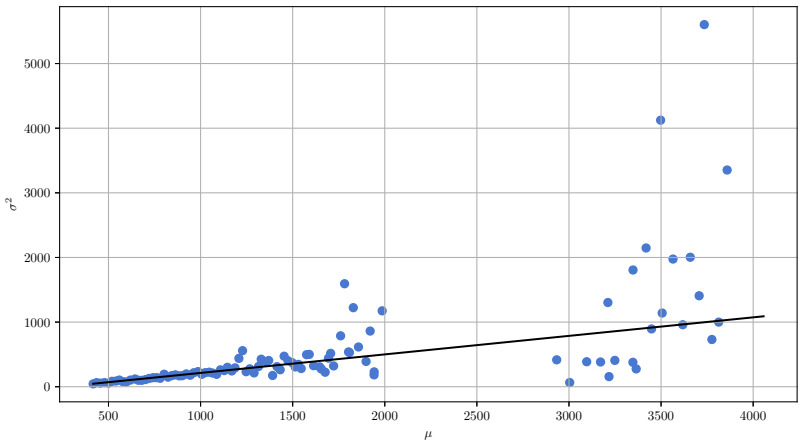


Figure 5.5: Mean-variance curve for the Canon EOS DIGITAL REBEL XSi camera at ISO 200. The blue dots correspond the mean/variances sample while the black line is the estimated curve.

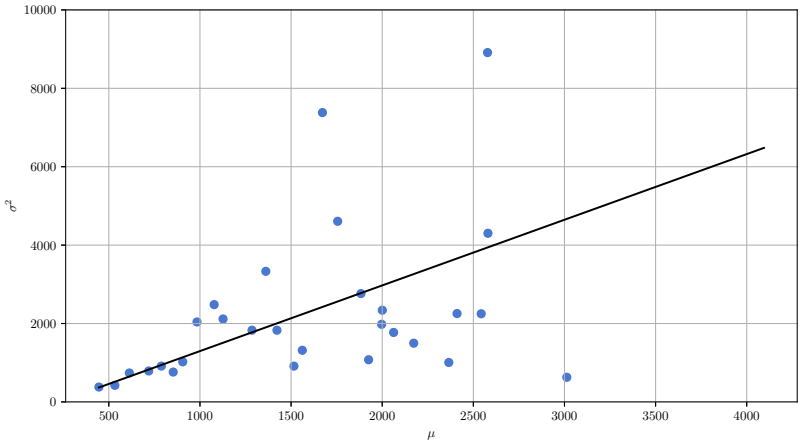


Figure 5.6: Mean-variance curve for the NIKON D70 camera at ISO 400. The blue dots correspond the mean/variances sample while the black line is the estimated curve. Very few usable samples were available for this curve due to the highly textured nature of the images.

heteroscedastic Gaussian noise – Eq 5.2.8. That is, the variance of the noise on each photo-sites is linear in its mean value. An important observation is that the parameters which control the variance in this model only depend on the camera and ISO parameters which were used to capture the image, thus validating in part or empirical definition of the source in Chapter 4.

Finally, we provided for completeness a method for estimating the parameters of the heteroscedastic model, first proposed by A. Foi in [26]. The method is based on segmenting the image in different classes so that areas belonging to the same class have the same mean value. This is done through a combination of wavelet analysis and filtering. The wavelet analysis allows a simple estimation of local statistics of the photo-sites. The filtering operations allow to find and discard edges and areas of high variance that are difficult to model. Finally, using the properties of the heteroscedastic model, a weighted least-square approach is used to fit a curve through the local estimates of the image.

In the next chapter, we continue our path through the image generation process by proposing a model of the processing pipeline.

6 Model of the processing pipeline

The RAW image model developed in Chapter 5 determined which part the sensor and the ISO parameter played in the structure of the noise of natural images. The only parameter left to incorporate into the model is the processing pipeline. Due to the great diversity of the possible operations available in image processing, we will only touch upon the most important ones on this chapter in order to build a model of the processing pipeline that is both sufficiently expressive to describe these operations while also being amenable to mathematical analysis.

6.1 Fundamental operations in image processing

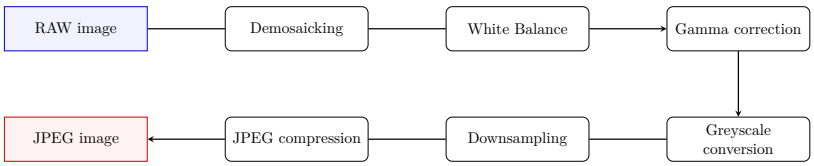


Figure 6.1: Order of the different image processing operations presented in this chapter. For an example of more comprehensive image processing pipeline – like the one used in the popular open-source software Rawtherapee – see https://rawpedia.rawtherapee.com/Toolchain_Pipeline.

In this section, we give an overview of the most fundamental operations of image processing that will be used throughout this manuscript. In particular, we study each operation through the lens of two categories: linearity and stationarity.

Linear operations are functions f of the form:

$$f(\sum_k \mathbf{x}_k) = \sum_k f(\mathbf{x}_k), \tag{6.1.1}$$

$$\forall \alpha \in \mathbb{R}, f(\alpha \mathbf{x}) = \alpha f(\mathbf{x}). \tag{6.1.2}$$

Stationary operations are operations that are applied identically for every input of the operation. For example, if the operation takes blocks of photo-sites as input, then every block should be processed in the same way. The demosaicking algorithm PPG, which will be described shortly, is an example of a non-stationary algorithm as it performs different operations on blocks depending on the gradient of the center photo-sites with its neighbors.

6.1.1 *Demosaicking*

The first fundamental step in the processing of a RAW image is demosaicking. This step transforms the RAW image, with each photo-sites

coding for only one color, into a color image composed of three color channels, usually red, green and blue (RGB).

The general process consists in leveraging the neighborhood of each photo-site in order to interpolate the missing colors at each position of the image.

Demosaicking operations can contain both linear and non-linear operations. The main algorithm often involves linear operations in the form of a series of convolutions with some family of kernels. But some non-linearities are also often present, as we will see with the PPG demosaicking algorithm described later in this section.

Similarly, they also contain both stationary and non-stationary operations. Non-stationary operations are often built-in to adapt to certain zones that are known to produce artifacts if not interpolated in a certain manner. Hence different operations are performed depending on, for example, the gradient between two neighboring photo-sites.

To illustrate these different types of operations, we now detail two classic demosaicking algorithms: (1) bilinear demosaicking, which will serve as the basis of our model and (2) the so-called Pixel Pattern Grouping (PPG) demosaicking algorithm, which contains linear/non-linear and stationary/non-stationary operations. It is also the demosaicking algorithm that has been the most extensively used in steganography benchmarking as it is the only algorithm used to produce the standard image dataset BOSSBase [5].

For the description of the demosaicking algorithm we need some notation pertaining to the red, green and blue sub-images of the RAW-image. We denote the set of spatial indices corresponding to red photo-sites as $\mathbf{l}_R$ and similarly for other colors and compute them by following the Bayer pattern in Figure 5.2:

$$\mathbf{l}_R = \{(i,j) | (i,j) \bmod 2 = (0,1)\} \tag{6.1.3}$$
$$\mathbf{l}_B = \{(i,j) | (i,j) \bmod 2 = (1,0)\} \tag{6.1.4}$$
$$\mathbf{l}_{G_1} = \{(i,j) | (i,j) \bmod 2 = (0,0)\} \tag{6.1.5}$$
$$\mathbf{l}_{G_2} = \{(i,j) | (i,j) \bmod 2 = (1,1)\} \tag{6.1.6}$$
$$\mathbf{l}_G = \mathbf{l}_{G_1} \cup \mathbf{l}_{G_2} \tag{6.1.7}$$

Bilinear demosaicking This algorithm is the most basic algorithm for interpolation[1]. It involves estimating, for each RGB channel, the pixel value of a given photo-site by computing a weighted average of the nearest-neighbours of the missing color.Let the RAW image in its matrix form $\mathbf{X}$. Similarly, let each channel of the demosaicked image be $\mathbf{Y}^R$, $\mathbf{Y}^G$, $\mathbf{Y}^B$. Each of these channels is obtained using the following procedure:

$$\mathbf{Y}^G_{i,j} = \begin{cases} X_{i,j} & \text{if } (i,j) \in \mathbf{l}_G \\ \frac{1}{4}\left(X_{i+1,j} + X_{i,j+1} + X_{i-1,j} + X_{i-1,j}\right) & \text{otherwise} \end{cases} \tag{6.1.8}$$

[1] Without considering the naive algorithm consisting in not interpolating but only taking the closest color value as the final value.

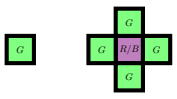


Figure 6.2: Green interpolation

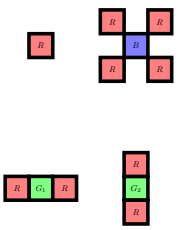


Figure 6.3: Red interpolation

$$\mathbf{Y}^R_{i,j} = \begin{cases} X_{i,j} & \text{if } (i,j) \in \mathbf{l}_R \\ \frac{1}{2}\left(X_{i,j+1} + X_{i,j-1}\right) & \text{if } (i,j) \in \mathbf{l}_{G_2} \\ \frac{1}{2}\left(X_{i+1,j} + X_{i-1,j}\right) & \text{if } (i,j) \in \mathbf{l}_{G_1} \\ \frac{1}{4}\left(X_{i+1,j+1} + X_{i-1,j+1} + X_{i-1,j+1} + X_{i-1,j-1}\right) & \text{otherwise} \end{cases} \tag{6.1.9}$$

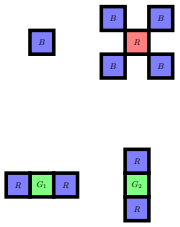


Figure 6.4: Blue interpolation

$$\mathbf{Y}^B_{i,j} = \begin{cases} X_{i,j} & \text{if } (i,j) \in \mathbf{l}_B \\ \frac{1}{2}\left(X_{i,j+1} + X_{i,j-1}\right) & \text{if } (i,j) \in \mathbf{l}_{G_1} \\ \frac{1}{2}\left(X_{i+1,j} + X_{i-1,j}\right) & \text{if } (i,j) \in \mathbf{l}_{G_2} \\ \frac{1}{4}\left(X_{i+1,j+1} + X_{i-1,j+1} + X_{i-1,j+1} + X_{i-1,j-1}\right) & \text{otherwise} \end{cases} \tag{6.1.10}$$

For each channel, all these linear operations can be rewritten as a single matrix. This is characteristic of the linearity and stationarity properties which will heavily be used for the modeling of the processing pipeline at the end of the chapter.

PPG demosaicking This algorithm is composed of two distinct steps. The first step is fully linear though non-stationary and involves estimating the full green channel. The second step is both non-linear and non-stationary and uses the green channel estimates to provide better estimates of the blue and red channels.

To the best of our knowledge, PPG full description can only be found in two places: by directly reading the *dcraw* source code or by reading an old page of the creator of the algorithm Chuan-Kai Li which can only be found on archive.org. Seeing the importance of this algorithm for the benchmarking of steganography and stegnanalysis, it seemed important for us to provide here a full description of the algorithm.

The first steps begin by recording the value of the green photo-sites in the green channel: they don't need to be interpolated. For finding the green value at blue and red photo-sites, we first compute the vertical and horizontal gradients:

$$\Delta N_{i,j} = 2|X_{i,j} - X_{i,j+2}| + |X_{i,j-1} - X_{i,j+1}| \tag{6.1.11}$$
$$\Delta S_{i,j} = 2|X_{i,j} - X_{i,j-2}| + |X_{i,j-1} - X_{i,j+1}| \tag{6.1.12}$$
$$\Delta E_{i,j} = 2|X_{i,j} - X_{i+2,j}| + |X_{i-1,j} - X_{i+1,j}| \tag{6.1.13}$$
$$\Delta W_{i,j} = 2|X_{i,j} - X_{i-2,j+2}| + |X_{i-1,j} - X_{i+1,j}| \tag{6.1.14}$$

Now let $\delta_{i,j}$ be the smallest of these four gradients, then the pixel value of the green channel $Y^G_{i,j}$ is obtained as:

$$Y^G_{i,j} = \begin{cases} \frac{1}{4}(3X_{i,j+1} + X_{i,j-1} + X_{i,j} - X_{i,j+2}) & \text{if } \delta_{i,j} = \Delta N_{i,j} \\ \frac{1}{4}(3X_{i,j-1} + X_{i,j+1} + X_{i,j} - X_{i,j-2}) & \text{if } \delta_{i,j} = \Delta S_{i,j} \\ \frac{1}{4}(3X_{i+1,j} + X_{i-1,j} + X_{i,j} - X_{i+2,j}) & \text{if } \delta_{i,j} = \Delta E_{i,j} \\ \frac{1}{4}(3X_{i-1,j} + X_{i+1,j} + X_{i,j} - X_{i-2,j+2}) & \text{if } \delta_{i,j} = \Delta W_{i,j} \end{cases} \tag{6.1.15}$$

The second step uses the following ad-hoc function

$$f(l_1, l_2, l_3, v_1, v_2) = \begin{cases} v_1 + (v_2 - v_1)\frac{(l_2 - l_1)}{l_3 - l_1} & \text{if } l_1 < l_2 < l_3 \text{ or } l_1 > l_2 > l_3 \\ \frac{v_1 + v_3}{2} + \frac{2l_2 - l_1 - l_3}{4} & \text{otherwise} \end{cases} \tag{6.1.16}$$

To estimate $Y^R_{i,j}$ or $Y^B_{i,j}$ for $(i,j) \in \mathbf{l}_{G_1}$, the following equations are used:

$$Y^R_{i,j} = f\left(Y^G_{i-1,j}, Y^G_{i,j}, Y^G_{i+1,j}, X^R_{i-1,j}, X^R_{i+1,j}\right) \tag{6.1.17}$$
$$Y^B_{i,j} = f\left(Y^G_{i,j-1}, Y^G_{i,j}, Y^G_{i,j+1}, X^B_{i-1,j}, X^B_{i+1,j}\right) \tag{6.1.18}$$

Similarly for if $(i,j) \in \mathbf{l}_{G_2}$:

$$Y^R_{i,j} = f\left(Y^G_{i,j-1}, Y^G_{i,j}, Y^G_{i,j+1}, X^R_{i-1,j}, X^R_{i+1,j}\right) \tag{6.1.19}$$
$$Y^B_{i,j} = f\left(Y^G_{i-1,j}, Y^G_{i,j}, Y^G_{i+1,j}, X^B_{i-1,j}, X^B_{i+1,j}\right) \tag{6.1.20}$$

To estimate $Y^R_{i,j}$ for $(i,j) \in \mathbf{l}_B$ we need to compute the diagonal gradients:

$$\begin{aligned}\Delta\mathrm{NE}_{i,j} = &|X_{i-1,j+1} - X_{i+1,j-1}| + |X_{i+2,j-2} - X_{i,j}| \\ &+ |X_{i-2,j+2} - X_{i,j}| + |Y^G_{i-1,j+1} - Y^G_{i,j}| \\ &+ |Y^G_{i+1,j-1} - Y^G_{i,j}|,\end{aligned} \tag{6.1.21}$$

$$\begin{aligned}\Delta\mathrm{NW}_{i,j} = &|X_{i+1,j+1} - X_{i-1,j-1}| + |X_{i-2,j-2} - X_{i,j}| \\ &+ |X_{i+2,j+2} - X_{i,j}| + |Y^G_{i-1,j-1} - Y^G_{i,j}| \\ &+ |Y^G_{i+1,j+1} - Y^G_{i,j}|.\end{aligned} \tag{6.1.22}$$

Using these gradients we obtain the estimates using f, for all $(i,j) \in \mathbf{l}_B$:

$$Y^R_{i,j} = \begin{cases} f(Y^G_{i-1,j+1}, Y^G_{i,j}, Y^G_{i+1,j-1}, X_{i-1,j+1}, X_{i+1,j-1}) \text{ if } \Delta\mathrm{NE}_{i,j} < \Delta\mathrm{NW}_{i,j} \\ f(Y^G_{i-1,j-1}, Y^G_{i,j}, Y^G_{i+1,j+1}, X_{i-1,j-1}, X_{i+1,j+1}) \text{ else} \end{cases} \tag{6.1.23}$$

The procedure is strictly identical for $Y^B_{i,j}$ at $(i,j) \in \mathbf{l}_R$.

6.1.2 *Gamma transform*

(a) With gamma transform (b) No gamma transform

Figure 6.5: Visual difference when using and not using the gamma transform when developing an image.

The gamma transform is an operation used to correct the global illumination of an image. Its standard form [64, p.3, Item 1.2] is given by the following function:

$$\Gamma(x) = \begin{cases} \gamma_0 x, & \text{if } x < 0.018 \\ 1.099 x^{\frac{1}{\gamma_1}} - 0.099, & \text{if } x \geq 0.018 \end{cases} \tag{6.1.24}$$

where γ_0 and γ_1 are user-specified parameters and $x \in [0, 1]$ is the normalized pixel value.

The gamma transform is another example of a function which is non-stationary and contains both a linear and non-linear part.

6.1.3 *White Balance*

(a) With white balance (b) No white balance

Figure 6.6: Visual difference when using and not using white balance when developing an image.

White balance is a corrective operation applied on each color channel. Its main aim is usually to make it so that a given neutral tone, such as white, will be perceived correctly by the eye when the image is displayed.

It is a very simple operation that only involves multiplying each color channel by a scalar. For example, let the red channel be denoted $\mathbf{Y}^R$, then the corrected channel $\mathbf{Y}^R_{WB}$ is simply:

$$\mathbf{Y}^R_{WB} = i^R \mathbf{Y}^R \tag{6.1.25}$$

where i^R is the corrective factor of the red channel. It is an example of a purely linear and stationary operation.

6.1.4 *Downsampling*

Sub-sampling is a destructive operation aiming at reducing the size of an image. In the context of image processing, it usually involves two steps. The first step, called the *anti-aliasing* aims to smooth the image content before the second step, named the *decimation step* which suppresses a given number of pixels until a given size is reached. The anti-aliasing step is performed to prevent the apparition of so-called Moiré effects, which are visual artifacts introduced by the removal of pixels in areas with strong intensity variations. The first operation is performed using a low-pass filter, which is identical on all the image. The decimation step is usually performed on a regular-grid making the full operation both linear and stationary.

6.1.5 *JPEG compression*

The processing pipeline usually ends by some form of compression to facilitate image storage and sharing by reducing its size on disk. The most popular standard for image compression is JPEG 1[2] – abbreviated as JPEG. This standard's specification contains both the encoding and decoding step which are summarized in Figure 6.7. Since this

[2] ISO/IEC 10918

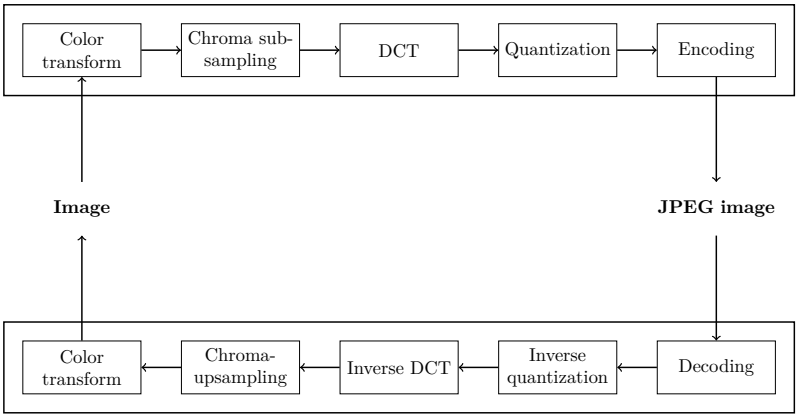
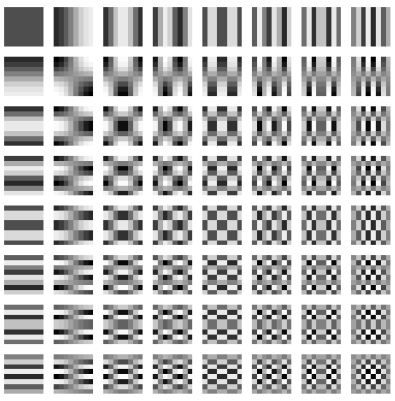
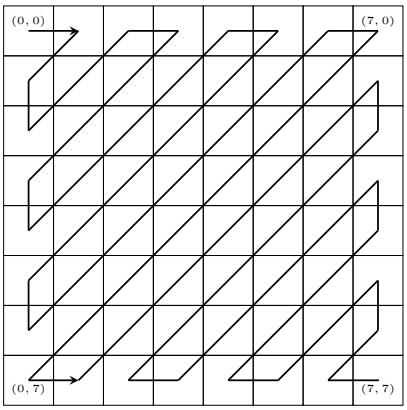


Figure 6.7: Series of operation for encoding and decoding a given image using the JPEG standard.

work only treats greyscale images, we do not present the first two operations, namely the colorimetric transform and chromatic sub-sampling.



(a) Representation of each DCT modes in the DCT transform. Each of checkerboard pattern corresponds to one of the cosine function used to represent the signal. The modes are here presented in the same spatial order as they would in a given DCT block.



(b) This diagram illustrate how DCT blocks are usually read during the encoding phase, that is, in a zig-zag order. This order allows sorting from the lowest-frequency mode to the highest frequency mode in an increasing frequency order.

Figure 6.8: Structure of the DCT basis of the DCT transform as cosine functions oscillating at different frequencies with lowest frequencies at the top-left corner and the highest at the bottom-right corner.

DCT transform The discrete cosine transform, often abbreviated improperly as DCT transform, is an operation applied on a uniform grid of 8×8 blocks of an image.

It is an orthonormal transform, analogous to the discrete Fourier transform but with additional properties relevant to image processing – see the footnote and the foundational publication [1]. It is based on the decomposition of an image into a linear combination of cosine functions oscillations at different frequencies. Each of these functions, called *DCT modes* are represented in Figure 6.8.

Historically, the DCT transform has been built as a fast approximation to the celebrated Karhunen-Loeve transform [2] which allows an optimal representation – in the least-square sense – of a stochastic process. In an image processing context, the main idea was to approximate a first-order Markov chain used to model the content of an image and use this representation to obtain a signal structure more amenable to compression.

For a given block $\mathbf{Y}$, the DCT transform is computed as:

$$Z_{k,l} = \sum_{i,j=0}^{7} \frac{w(k)w(l)}{4} \cos\left(\frac{\pi}{16}k(2i+1)\right) \cos\left(\frac{\pi}{16}k(2j+1)\right) Y_{i,j}, \tag{6.1.26}$$

with:

$$w : x \rightarrowtail \begin{cases} \frac{1}{\sqrt{2}} & \text{if } x = 0, \\ 1 & \text{else.} \end{cases} \tag{6.1.27}$$

where $\mathbf{Z}$ is called a DCT block and its elements $Z_{k,l}$ DCT coefficients. The first DCT coefficient (the upmost, leftmost mode in Figure 6.8) is referred to as a DC coefficient referencing, by analogy to direct current, the fact that it is simply the average of all pixels in the block. Consequently, the rest of the coefficients are termed AC coefficients – by analogy to alternative current.

The inverse transform is strictly analogous:

$$Y_{i,j} = \sum_{k,l=0}^{7} \frac{w(k)w(l)}{4} \cos\left(\frac{\pi}{16}k(2i+1)\right) \cos\left(\frac{\pi}{16}k(2j+1)\right) Z_{k,l} \tag{6.1.28}$$

The DCT transform allows representing each block of pixels as a linear combination of DCT modes. The main property of interest of this representation is that most of the signal energy in natural images is concentrated at low frequencies, with details and noise usually concentrated in the high frequencies. This means that after the DCT transform, blocks of DCT coefficients will tend to have their energy concentrated in coefficients close to the DC coefficients, whereas the value of the DCT coefficients will be closer to zero at the bottom right corner of the block. This property is the basis of the quantization operation described in the next paragraph.

(a) QF100 (b) QF5

Figure 6.9: Impact of quantization for different quality factors. Notice the loss of details for QF5, one of the lowest quality factor and rarely used, when compared to QF100, the highest and most widely used quality factor.

Quantization The quantization step is the only step that induces a loss of information and, consequently, a loss of visual quality during JPEG compression. This loss of information allows reducing drastically the file size; the end goal being to optimize the trade-off between visual quality and file size.

To do so, the quantization step will concentrate most of the information loss in the high-frequency modes. These frequencies correspond to image features which are overall less perceptible than features corresponding to low frequencies. Therefore, suppressing them allows a high gain in storage with minimal visual impact.

In practice, the quantization operation is performed in three steps. At the first step, the user chooses a value between 0 and 100, usually referred to as the quality factor (QF). The quality factor determines the trade-off between visual quality and final image size with higher values corresponding to better visual quality. Each quality factor is associated to a *quantization matrix* $\mathbf{Q} \in \mathbb{R}^{8\times 8}$ defined for each quality factor by the JPEG standard.

Once it has been selected, each DCT coefficient of a given block is divided by the corresponding element in the quantization matrix. Lastly, a rounding operation is performed on every DCT coefficient. Formally, each coefficient $\tilde{Z}_{k,l}$ of a quantized DCT block $\tilde{\mathbf{Z}}$ is obtained by:

$$\tilde{Z}_{k,l} = \text{round}\left(\frac{Z_{k,l}}{Q_{k,l}}\right), \tag{6.1.29}$$

Encoding The coding step is a sequence of lossless operations performed after quantization leveraging the structure of the quantized signal to decrease the size of the JPEG file on disk even more.

Indeed, one of the main results of the quantization step is that high-frequency modes in a DCT block have a high probability of being zeroed out. Consequently, the first step is to leverage the high number of zeros in the DCT blocks after quantization. For each DCT block, a sequence of coefficients is built by reading each block in a zigzag manner as illustrated in Figure 6.8. High frequencies will hence be at the end of the sequence and will have a high-probability to contain long strings of zeros at the end. A run-length encoding (RLE) scheme is used to reduce the size of this sequence by removing, among over things, long sequences of zeros. As a final step, a Huffman-type entropic coding scheme is applied to the whole sequence of DCT coefficients to obtain the final JPEG file.

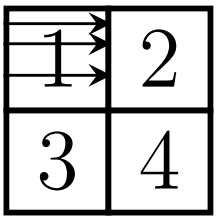
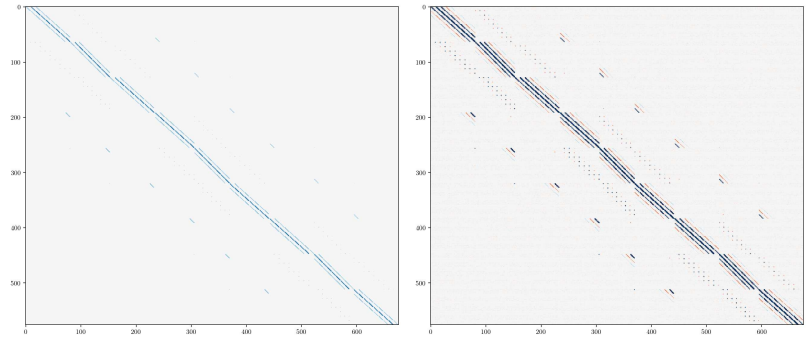


Figure 6.10: Illustration of how blocks of DCT coefficients are read to produce figures such as Figure 6.11. Each numbered square represents a 8×8 block of DCT coefficients. Elements inside a block are read from left to right, up to down. All elements of a block are read before going to the next. Similarly, blocks are also read from left to right, up to down as the numbers on the illustration indicate.

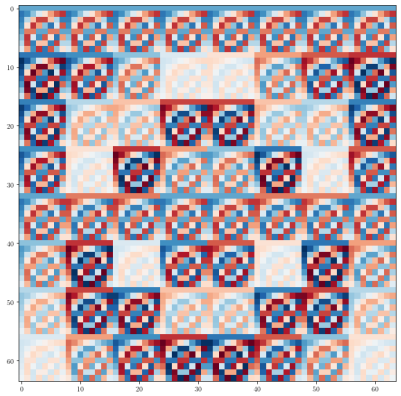
6.2 Linear approximation of the processing pipeline

With the exception of PPG demosaicking and of the Gamma transform, all the operations of the processing pipeline we studied are both linear and stationary. Though we only touched upon a small subset of every possible operations; this observation would tend to show that an approximation of the processing pipeline as a single linear and stationary operator would allow for both generality and computational

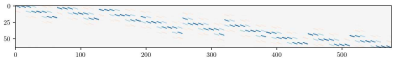


(a) Linear demosaicking

(b) PPG demosaicking



(c) DCT transform



(d) Lanczos downsampling

Figure 6.11: Matrix representation of different pipeline and different choice of neighborhood size. These matrices were estimated using the method presented in Section 7.4.2. Maximum and minimum values have been selected for a better visualization of the matrix structure. Blue entries are positive while red entries are negative.

tractability.

Formally, we model the processing pipeline as a full-rank matrix $\mathbf{H} \in M \times N$. The choice of M of N is a trade-off between the precision of the model and its computational complexity. The larger M and N are, the more dependencies between pixel and DCT coefficients can be taken into account. At the same time, a large $\mathbf{H}$ incurs a high cost for the matrix multiplication that will have to be performed to use this model. However, it has been shown that for some processing pipelines, optimal values of M and N exist. As an example, Taburet et al. have shown in [75] that $M = 26^2$ and $N = 24^2$ is sufficient to capture all the dependencies introduced by a bilinear demosaicking followed by a DCT transform.

For the rest of this manuscript, we will only work with grayscale JPEG images. Consequently, we always assume that N is a perfect square[3] and is a multiple of 8 in order to take into account the fact that the DCT transform acts on 8×8 blocks of pixels.

[3] That is, there exist $N' \in \mathbb{N}$ such that $N = (N')^2$

6.2.1 *Estimation of the matrix representing the processing pipeline*

In this manuscript, we always assume that the steganographer has at least a black-box access to the processing pipeline which was used to generate the cover image they will use to hide information in. That is, the steganographer has no direct knowledge of the inner workings of the processing pipeline but can use it to develop images.

From this black-box access, we want to estimate the matrix representation $\mathbf{H}$ of this processing pipeline.

To do so we first generate a constant image $\bar{\mathbf{x}}$ corrupted by independent and identically distributed centred Gaussian noise with variance σ^2:

$$\bar{x}_i \sim \mathcal{N}\left(c, \sigma^2\right). \tag{6.2.1}$$

The constant value c can, in theory, be chosen arbitrarily. But most image processing software will clip values that fall below zero. Consequently, we always choose a constant value that is close to the mid-value of the range of a given RAW image, usually 2000 or 8000 since the standard ranges are either 2^{12} or 2^{14}.

The image $\bar{\mathbf{x}}$ is then developed, using the black box access to the processing pipeline, into the developed image $\bar{\mathbf{y}}$. Following our stationarity and linearity assumptions, we can express $\bar{\mathbf{y}}$ as a function of $\bar{\mathbf{x}}$ and $\mathbf{H}$:

$$\bar{\mathbf{y}}_k = \mathbf{H}\,\bar{\mathbf{x}}_k, \tag{6.2.2}$$

that is, every macro-block $\bar{\mathbf{y}}_k$ of the developed image is obtained by multiplying the corresponding macro-block in the RAW domain $\bar{\mathbf{x}}_k$ by $\mathbf{H}$.

From Eq (6.2.2), we can reduce the problem of estimating $\mathbf{H}$ to a simple linear regression problem. For example we can solve Eq (6.2.2) for $\mathbf{H}$ using the least-square method. Let m_x be the number of macro-blocks of size M^2 in $\bar{\mathbf{x}}$ and m_y the number of corresponding macro-blocks of size N^2 in $\bar{\mathbf{y}}$ (in vector form). Then let $\bar{\mathbf{X}}$ be the $M^2 \times m_x$ matrix built by stacking each non-overlapping macro-block in vector form of $\bar{\mathbf{x}}$ and similarly for $\bar{\mathbf{Y}}$. Then the least-square solution of $\mathbf{H}$ is given by:

$$\mathbf{H} = \bar{\mathbf{Y}}\bar{\mathbf{X}}^T\left(\bar{\mathbf{X}}\bar{\mathbf{X}}^T\right)^{-1}. \tag{6.2.3}$$

6.3 Conclusion

In this chapter, we first presented some of the most common operations in image processing, from demosaicking to JPEG compression. By observing the common properties of these operations, we proposed a general model of the processing pipeline built around two major assumptions: linearity and stationarity. Concretely, this leads to modeling the processing pipeline as a matrix $\mathbf{H}$ (linearity) that takes (macro)-blocks of photo-sites as input and outputs (macro)-blocks of DCT coefficients. The stationarity assumption means that this matrix is not dependent on the input.

We ended the chapter by presenting a simple estimation method of the $\mathbf{H}$ matrix when having access only to the processing pipeline as a black box. It is based on generating a constant image to which noise is added. This constant image is then developed using the processing pipeline. A simple least square estimator is then applied on the undeveloped and developed block of the image to obtain $\mathbf{H}$.

In the following chapter, we combine the results of this chapter and of Chapter 5 to finally obtain a realistic model of the noise in the developed model for natural images.

7 Model of the noise in the developed domain

In this chapter, we finally derive the general model of the noise in the developed domain by combining the models from Chapter 5 and Chapter 6. Then, we continue by giving two different methods to estimate the parameters of the model depending on the information available to the steganographer. We finish by experimentally validating our model through a set of experiments using different pipelines.

7.1 Derivation of the multivariate Gaussian model

Let $\mathbf{x}_k$ be a macro-block containing M photo-sites. From the model in Chapter 5 we have:

$$\mathbf{x}_k \sim \mathcal{N}\left(\boldsymbol{\mu}_k, \mathrm{diag}\left(\boldsymbol{\sigma}_k\right)\right). \tag{7.1.1}$$

Assuming the stationarity of the processing pipeline, we can write the developed macro-block as:

$$\mathbf{y}_k = \mathbf{H}\,\mathbf{x}_k\,. \tag{7.1.2}$$

Furthermore, assuming the linearity of the processing pipeline and using the properties of the Gaussian distribution, it is immediate that $\mathbf{Y}_k$ follows a multivariate Gaussian distribution:

$$\mathbf{y}_k \sim \mathcal{N}\left(\mathbf{H}\boldsymbol{\mu}_k, \Sigma_k\right), \tag{7.1.3}$$

with the covariance Σ_k computed by:

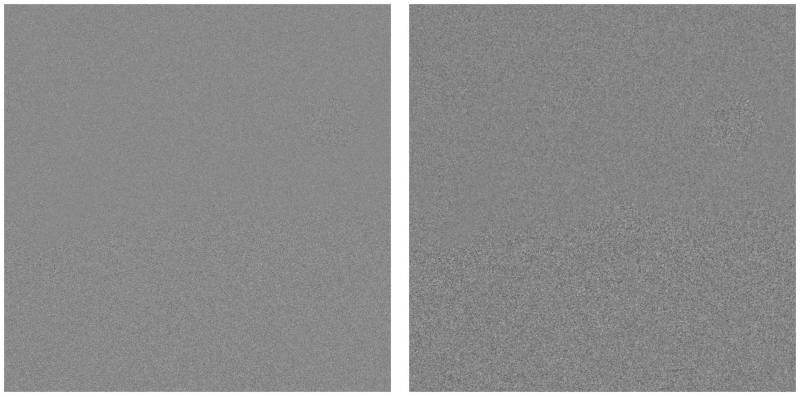
$$\Sigma_k = \mathbf{H}\mathrm{diag}\left(\boldsymbol{\sigma}_k\right)\mathbf{H}^T. \tag{7.1.4}$$

To sum up, under the hypotheses of our model, the noise of macro-blocks in the developed domain follows a multivariate Gaussian model. This model is a function of only three parameters: the heteroscedastic parameters c_1 and c_2, which depend on the imaging sensor and on the ISO setting, and the processing pipeline, here modeled as a matrix $\mathbf{H}$.

7.2 Dependency model between macro-blocks

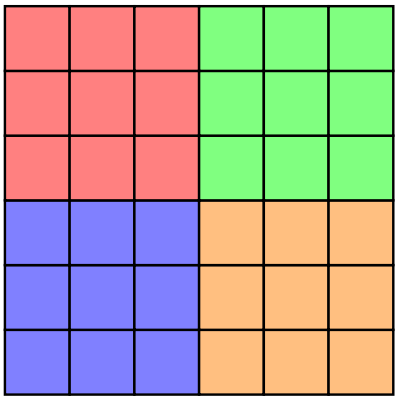
The last element missing in this model is the dependency model between macro-blocks $\mathbf{y}_k$. Indeed, the model only explicitly provides dependencies inside a given macro-block of a specified size but does not inform us on dependencies that might have been introduced between two neighboring macro-blocks, for example.

In this manuscript, we will study two models of dependencies between macro-blocks:



(a) Uncorrelated noise (b) Correlated noise

Figure 7.1: Comparison between correlated noise (right) and uncorrelated noise (left) which share the same diagonal in their covariance matrix. The correlation has been produced using linear demosaicking.



(a) Independent macro-block model: every non-overlapping macro-blocks of a given size is considered to be independent with the others. In this figure,the chosen size is 24×24, hence every block is only dependent with blocks with which it shares its color.

(b) Lattice model: every block is considered dependent only with its direct neighbouring blocks. This induces four sets of blocks, called lattices, $\mathbf{\Lambda}_1, \mathbf{\Lambda}_2, \mathbf{\Lambda}_3, \mathbf{\Lambda}_4$ where each block in a given set is independent of every other block in the same set.

Figure 7.2: The two dependency models studied in this work.

1. *Independent macro-blocks model*: all macro-blocks of size $\sqrt{N} \times \sqrt{N}$ on a uniform grid are considered independent of each other.

2. *Lattice model*: all blocks of size 8×8 on a uniform grid are considered dependent only with all their neighboring blocks, including the diagonal ones.

   These two models are illustrated in Figure 7.2.

7.2.1 *Independent macro-block*

This model is based on a uniform grid of macro-blocks of size $\sqrt{N} \times \sqrt{N}$, that is, only non-overlapping macro-blocks are considered. Each of the macro-blocks constructed in this way is assumed to be independent with other non-overlapping macro-blocks. The dependencies are then only modeled for elements inside a given macro-block.

The model is only exact at the limit where the chosen dimensions of the – then unique – macro-block matches the dimensions of the image. Otherwise this model is not able to capture any dependencies between macro-block. Nevertheless, these dependencies will exist most of the time. Indeed, if there is but one operation in the processing pipeline involving a convolution with a kernel of size greater than one – which might be considered the case for all pipelines of interest, in particular for demosaicking – dependencies will be introduced between neighboring macro-blocks – see Figure 7.3 for a visual explanation of this fact.

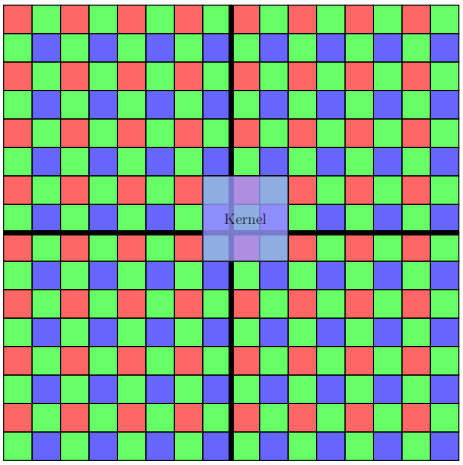


Figure 7.3: Illustration of the dependencies introduced between 8×8 blocks after demosaicking. Blocks are separated by thick black lines, whereas the kernel of the demosaicking algorithm is shown as a blue square. In practice, the kernel is moved along each photo-site. Some pixels will necessarily be a weighted sum of photo-sites belonging to different blocks, hence the fact that the independent macro-block model is necessarily inexact for most pipeline of interest.

Using the exact version of this model is evidently extremely costly due to the resulting size of the $\mathbf{H}$ matrix. For practical applications, we will only use the sizes $M = 8^2$ or $M = 24^2$ corresponding respectively to a modeling of intra block dependencies and intra+inter block in the DCT domain.

On the other hand, thanks to its simplicity and its generality, this model will be useful as a baseline for the analysis of the more complicated lattice model in Chapter 10. In particular, we will see that

the steganographic algorithms based on these two models are almost identical in term of implementation.

7.2.2 *Lattice model*

This model assumes 8×8 blocks to be dependent with their neighboring blocks. As a consequence, there exist a natural partition of block into four sets of vectors Λ_1, Λ_2, Λ_3 and Λ_4 which we will denote as lattices. Each lattice is built by collecting every other block starting at different spatial indices for each lattice – see Figure 7.2. From our starting assumption, each block in a given lattice is independent of every other block in the same lattice. This structure, combined with the Gaussianity of the noise, has extremely interesting properties for steganography which we review in Section 2.3.2.

Furthermore, this model allows to model dependencies in the DCT domain for all pipelines involving only operations that use a neighborhood of fewer than eight elements in each direction – that is, operations that do not span more than two DCT blocks. This is a very common case in practice as convolutions with kernels of size greater than 16 is quite expensive in terms of computation.

7.3 Covariance matrix estimation with access to the RAW image

In this section and the next one, we provide two methods to estimate the covariance matrices Σ_k of an image macro-blocks in the developed domain. The first method, presented in this section, is exact as long the model assumptions are valid and assumes that one has access to the RAW image $\mathbf{x}$ as well as to a black-box access to the processing pipeline.

The first step is to estimate the $\mathbf{H}$ matrix representing the processing pipeline. For this we refer to Section 6.2.1. Secondly, we estimate the parameters c_1 and c_2 of the heteroscedastic model following the method described in Section 5.3.

Once all these parameters have been estimated, we can compute the variance of each photo-site x_i by using:

$$\sigma_i^2 = c_1 \mu_i + c_2. \tag{7.3.1}$$

The mean value of the photo-site μ_i can be estimated using any denoising algorithm in the RAW domain such as [27]. However we have found that, in practice, using the actual value of the photo site as an estimate of the mean value does not lead to any loss for steganography.

We now rewrite Eq (7.3.1) as macro-blocks of variance:

$$\boldsymbol{\sigma}_k^2 = c_1 \, \boldsymbol{\mu}_k + c_2. \tag{7.3.2}$$

The covariance Σ_k of each macro-block is then simply obtained by using Eq (7.1.4).

7.4 Estimation and approximation of the covariance matrix without the RAW file

This section presents a second method to estimate the covariance matrix without having access to the RAW file. However, we still assume access to the processing pipeline, at least as a black box, since the estimation method mostly relies on having a linear approximation of this pipeline. We will also assume that the steganographer knows the c_1 and c_2 parameters of the heteroscedastic model of the cover. This knowledge does not necessarily require access to the RAW image, it is sufficient to know the camera and ISO which were used to capture the cover; this information is usually available in the EXIF metadata of an image. This second method relies on certain approximations. We show in Section 11.2 that it does not lead to a loss in performance when used for steganography.

The estimation method presented in this section is based on estimating a generic correlation matrix that depends only on the processing pipeline. The idea is then (1) to compute the variance of every DCT coefficient by using an approximate heteroscedastic model of the DC coefficients before (2) computing an estimation of the covariance of each block by scaling the correlation matrix using these estimated variances.

7.4.1 *Heteroscedastic model of the DC coefficients*

Our goal here is to show that, under some additional assumptions on the processing pipeline, the variance of the DC coefficients of each block is linear with respect to the expectation of this DC coefficients. Note that, for this section only, we distinguish between $\mathbf{H}^{DCT}$, the 64×64 matrix representing the DCT transform and $\mathbf{H}^s$ the $64 \times M$ matrix representing all operations performed in the RAW and spatial domain. We therefore have: $\mathbf{H} = \mathbf{H}^{DCT}\mathbf{H}^s$.

We assume, as usual, that the processing pipeline is both linear and stationary. Furthermore, we assume the RAW image is almost constant by block, that is, for all k we have $\boldsymbol{\mu}_k = \hat{\mu} + \mathbf{e}_k$ with $|e_{k,i}| \ll \hat{\mu}, 1 \leq i \leq M$.

First of all, let us rewrite the models of the block of photo-sites and of DCT coefficients:

$$\mathbf{x}_k \sim \mathcal{N}\left(\boldsymbol{\mu}_k, c_1 \mathrm{diag}\left(\boldsymbol{\mu}_k\right) + c_2\right), \tag{7.4.1}$$

$$\mathbf{y}_k \sim \mathcal{N}\left(\mathbf{H}^{DCT}(\mathbf{H}^s\,\boldsymbol{\mu}_k - 128), \mathbf{H}\mathrm{diag}\left(c_1\,\boldsymbol{\mu}_k + c_2\right)\mathbf{H}^T\right). \tag{7.4.2}$$

Note here that, contrary to Eq 7.1.2, we take into account the fact that we subtract 128 to each pixel before the DCT transform as this has an impact on the estimation method of this section (whereas it does not when using the method which uses RAW file). We can express the first two moments of the DC coefficient $y_{k,1}$ of each block:

$$\mathbb{E}\left[y_{k,1}\right] = \sum_{l=1}^{M} H_{1,l}\left(\hat{\mu} + e_{k,l}\right) - 1024, \tag{7.4.3}$$

$$\mathrm{Var}\left[y_{k,1}\right] = c_1 \sum_{l=1}^{M} H_{1,l}^2 \hat{\mu} + H_{1,l}^2 c_2 + c_1 \sum_{l=1}^{M} H_{1,l}^2 e_{k,l}. \tag{7.4.4}$$

Let us note the following quantities:

$$\bar{H}_{i,j} \triangleq \frac{\sum_{l=1}^{M} H_{i,l}^2}{\sum_{l=1}^{M} H_{j,l}}, \bar{H}_{i,j}^{(2)} \triangleq \frac{\sum_{l=1}^{M} H_{i,l}^2}{\sum_{l=1}^{M} H_{j,l}^2}. \tag{7.4.5}$$

The variance of $y_{k,1}$ can be expressed using its expectation:

$$\begin{aligned}\mathrm{Var}\left[y_{k,1}\right] &= c_1 \bar{H}_{1,1} \mathbb{E}\left[y_{k,1}\right] + \sum_{l=1}^{M} H_{1,l}^2 c_2 + 1024 \cdot c_1 \bar{H}_{1,1} \\ &\quad + c_1 \left(\sum_{l=1}^{M} H_{1,l}^2 e_{k,l} - \bar{H}_{1,1} \sum_{l=1}^{M} H_{1,l} e_{k,l} \right) \\ &\triangleq c_1^{DC} \mathbb{E}\left[y_{k,1}\right] + c_2^{DC} + \text{error},\end{aligned} \tag{7.4.6}$$

with c_1^{DC} and c_2^{DC} defined as:

$$c_1^{DC} = c_1 \bar{H}_{1,1}, \tag{7.4.7}$$

$$c_2^{DC} = \sum_{l=1}^{M} H_{1,l}^2 c_2 + 1024 \cdot c_1 \bar{H}_{1,1}. \tag{7.4.8}$$

This shows that the variance of the DC coefficients is linear with their expectation up to an error which is small as long as the ratio $\bar{H}_{1,1}$ is small. As a particular case, note that if all the $H_{1,l}$ are constant, for example if the only operation of the processing pipeline performed is the DCT transform, then the error is simply 0. In this manuscript, we assume that the error is negligible in practice.

Note however that, the $H_{i,l}$ usually alternate sign for $i > 1$ because of the structure of the DCT transform. This leads to the ratio $\bar{H}_{i,l}$ possibly exploding and the error can not be considered small anymore for AC coefficients in general. However we can still express the variance of the AC coefficients as a function of the variance of the DC coefficient:

$$\begin{aligned}\mathrm{Var}\left[y_{k,i}\right] &= c_1 \hat{\mu} \sum_{l=1}^{M} H_{i,l}^2 + c_2 \sum_{l=1}^{M} H_{i,l}^2 + c_1 \sum_{l=1}^{M} H_{i,l}^2 e_{k,l} \\ &\simeq \bar{H}_{i,1}^{(2)} \mathrm{Var}\left[y_{k,1}\right],\end{aligned} \tag{7.4.9}$$

using the fact that $|e_{k,l}|$ is small compared to $\hat{\mu}$.

7.4.2 *Processing pipeline and correlation matrix estimations*

The first step is to estimate the processing pipeline matrix $\mathbf{H}$ since we only assumed a black-box access to the processing pipeline.

To do so, we first generate an image $\bar{\mathbf{x}}$ so that:

$$\bar{x}_i \sim \mathcal{N}\left(0, \sigma^2\right), \tag{7.4.10}$$

where the value of σ^2 can be freely chosen and does not impact the estimation.

Using the black-box access to the pipelines, we develop this image to obtain the developed image $\bar{\mathbf{y}}$.

Using Eq (7.1.2), we know that for a given macro-block size M, the processing pipeline outputs a macro-block of size N and that the k-th macro-block of the image follows:

$$\bar{\mathbf{y}}_k = \mathbf{H}\bar{\mathbf{x}}_k, \tag{7.4.11}$$

which can be solved using any type of linear regression method. For example, we can solve for $\mathbf{H}$ using a least-square estimation:

$$\mathbf{H} = \bar{\mathbf{y}}\bar{\mathbf{x}}^T \left(\bar{\mathbf{x}}\bar{\mathbf{x}}^T\right)^{-1}, \tag{7.4.12}$$

where $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$ here correspond respectively to the $M \times m$ and $N \times n$ matrices where each column corresponds to a vectorized macro-block.

To approximate the covariance matrices, we will also use a correlation matrix $\boldsymbol{\rho}$ based on the processing pipeline. First we compute the sample covariance matrix of $\bar{\mathbf{y}}$:

$$\Sigma_{\bar{\mathbf{y}}} = \frac{1}{m-1} \sum \bar{\mathbf{y}}_k \bar{\mathbf{y}}_k^T, \tag{7.4.13}$$

and obtain the correlation matrix:

$$\boldsymbol{\rho} = \operatorname{diag}\left(\Sigma_{\bar{\mathbf{y}}}\right)^{-\frac{1}{2}} \Sigma_{\bar{\mathbf{y}}} \operatorname{diag}\left(\Sigma_{\bar{\mathbf{y}}}\right)^{-\frac{1}{2}}. \tag{7.4.14}$$

7.4.3 *Approximation of the covariance matrix*

With the use of the heteroscedastic parameters c_1^{DC} and c_2^{DC} and of the AC-DC model, it is now possible to compute an approximation of the covariance matrix using the correlation matrix $\boldsymbol{\rho}$ computed in Section 7.4.2.

To do so, we first compute an approximation of the true variance map of the DCT coefficients and use it to scale the correlation matrix:

1. Compute the variance $\varrho_{k,1}^2$ of the DC coefficient of the k-th block $\mathbf{y}_k$ as:
$$\varrho_{k,1}^2 = c_1^{DC} y_{k,1} + c_2^{DC}. \tag{7.4.15}$$

2. Compute the variances of the i-th coefficients of the k-th block simply as:
$$\varrho_{k,i}^2 = \bar{H}_{i,1}^{(2)} \varrho_{k,1}^2. \tag{7.4.16}$$
.

3. Compute the covariance matrix of the k-th block by scaling the correlation matrix $\boldsymbol{\rho}$ with the variances of the DCT coefficients of the block using the standard formula:

$$\hat{\Sigma}_k = \text{diag}(\boldsymbol{\varrho}_k)\ \boldsymbol{\rho}\ \text{diag}(\boldsymbol{\varrho}_k), \tag{7.4.17}$$

where $\boldsymbol{\varrho}_k$ is the vector of standard deviation of the k-th block of DCT coefficients and $\hat{\Sigma}_k$ the resulting estimation of the covariance matrix.

7.5 Correction due to clipping

Up until now, we have ignored an important aspect of the behavior of the noise in the RAW domain: the clipping of values outside of the range of the sensors. Camera sensors have a range of value which they can register, usually going from 0 to 2^{14} or 2^{12}. Any value below the minimum value or above the maximum value will be clipped. This leads to areas where the variance of the noise is very low due to photo-sites having values close or equal to the minimum or maximum.

Consequently, the model of the variance presented in Chapter 5 is slightly incomplete as it does not take this aspect into account. A model closer to reality has the form given in Figure 7.4.

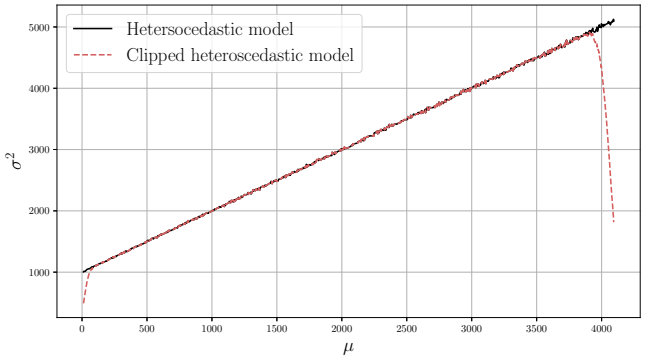


Figure 7.4: Variance σ^2 of 10000 samples of $\mathcal{N}(\mu, c_1\mu + c_2)$ as function of μ with $c_1 = 1$ and $c_2 = 1000$. The curve in red is obtained when the samples are clipped at 0 and 4096 while the curve in black is obtained when the samples are not clipped.

.

Statistical models taking into account this phenomenon have been devised such as in [27], [4] and [78] for a specific application to steganalysis. However, these model are a lot more complicated to handle mathematically for our purpose.

In order to solve this problem while keeping the simplicity of our model, we have experimentally found that forbidding some embedding locations is counter-productive as it destroys the covariance structure of the block where the changes are forbidden.

A better solution is to modify directly the covariance matrix in the RAW domain during the estimation phase. In particular, we empirically found that a relevant heuristic is to set the variance to 1 when the mean value of the photo-site μ_i is greater than $0.95S$ where S is the saturating value of the camera sensor.

Similarly, it is important to set a threshold for variances near zero for numerical stability as we use the Cholesky decomposition of the covariance matrix later in this manuscript. As long as the value is small we haven't found the exact value of the threshold to matter; as

a consequence we fixed it to 10^{-5}, that is we fix the variance σ_i^2 to 10^{-5} if $\sigma_i^2 < 10^{-5}$.

When using the method which does not use the RAW file, we have found that a sufficient method to obtain satisfying results is to set $\varrho_{k,i} = 10^{-7}$ for all i if the k-th block contains saturated pixels in the spatial domain.

These simple heuristics are empirically validated in Chapter 11 of this manuscript.

7.6 Experimental validation of the model

We finish this chapter by presenting a series of experiments to validate the model of the noise for developed images derived in this chapter. Two main features of the model need to be validated: the multivariate Gaussianity of the noise and the form of the covariance which should only depend on the sensor type, ISO and processing pipeline matrix $\mathbf{H}$.

Since the model is meant to be exact for a linear and stationary pipeline but only approximate for other type of pipelines we will perform our experiments on three different pipelines which exhibit different settings from the ideal case. The first pipeline, which we name the *Linear pipeline* is composed only of purely linear and stationary operations, namely bilinear demosaicking and DCT transform. The other pipeline, named the *Bosslike pipeline* since it is almost identical to the pipeline used to build the standard image dataset BOSSBase. It will introduce both non-linear and non-stationary operations due to the presence of the PPG demosaicking algorithm. However the Bosslike pipeline is still composed of several linear and stationary operations such as white-balance, sub-sampling and the DCT transform which are very well captured by our model. Consequently we also provide experiments using the *ALASKA* pipeline[1] which is composed of a variety of non-linear and non-stationary operations such as the aMaZe demosaicking algorithm, denoising and sharpening using the *Rawtherapee 5.8* software.

[1] The pipeline is named after the ALASKA competitions [12] which proposed a dataset composed of images developed randomly with a large variety of processing pipelines.

The Linear and Bosslike pipelines, which will be used throughout this manuscript, are summarized in Table 7.1.

Pipeline name	Demosaicking	White Balance	RGB to grey	Downsampling method
Linear Pipeline	Bilinear	No	Yes	Crop, 264×264
BOSS Pipeline	PPG	Yes, Camera	Yes	Resize from 792×792 crop to 264×264, Lanczos kernel

Table 7.1: Names and operations of the processing pipelines used in this manuscript. The operations are performed in the order they are presented in the table

7.6.1 *Experimental setup*

In these experiments we want to compare the covariance estimated with Monte-Carlo simulations Σ^{dev} which we will consider the ground truth and the covariance $\Sigma^{\mathbf{H}}$ estimated with the method presented in Section 7.3. For each of the three pipelines we generate n RAW images such that the photo-sites of all images follow a Gaussian random variable with mean 3958 and variance 29145. Each of these images are then developed using the chosen processing pipeline. The RAW images

are then split into m_1 macro-blocks of size $M \times M$ and the developed image into m_2 macro-blocks of size $N \times N$. The nm_1 RAW macro-blocks are collected into a $nm_2 \times M$ matrix denoted as $\mathbf{X}$ and similarly for the developed macro-blocks which are collected into a nm_2 matrix denoted as $\mathbf{Y}$.

Then, we compute $\mathbf{H}$ using a linear regression between $\mathbf{X}$ and $\mathbf{Y}$ in the same way as in Section 6.2.1. The covariance of RAW macro-blocks Σ^{raw} and the covariance of the developed macro-blocks Σ^{dev} are computed as:

$$\Sigma^{\text{raw}} = \frac{1}{nm_1 - 1} \sum_{k=1}^{nm_1} \mathbf{X}_k \, \mathbf{X}_k^T \tag{7.6.1}$$

$$\Sigma^{\text{dev}} = \frac{1}{nm_2 - 1} \sum_{k=1}^{nm_2} \mathbf{Y}_k \, \mathbf{Y}_k^T \tag{7.6.2}$$

We want to compare Σ^{dev} to the following estimate, $\Sigma^{\mathbf{H}}$:

$$\Sigma^{\mathbf{H}} = \mathbf{H} \, \Sigma^{\text{raw}} \, \mathbf{H}^T \tag{7.6.3}$$

To do so we will use three measure of error between $\Sigma^{\mathbf{H}}$ and Σ^{dev}. The first two measures of error are the distances induced by the Frobenius and max norm of matrices:

$$|| \, \mathbf{A} - \mathbf{B} \, ||_F \triangleq \sqrt{\sum_{i,j} (A_{i,j} - B_{i,j})^2} \tag{7.6.4}$$

$$|| \, \mathbf{A} - \mathbf{B} \, ||_\infty \triangleq \max_{i,j} |(A_{i,j} - B_{i,j})| \tag{7.6.5}$$

The third measure of error is of more interest to our application to steganography. It is based on the Kullback-Leibler divergence between two Multivariate Gaussian random variables with equal mean but different covariances:

$$|| \, \mathbf{A} - \mathbf{B} \, ||_{KL} \triangleq \frac{1}{2} \left(\text{trace} \left(\mathbf{B}^{-1} \, \mathbf{A} \right) - K + \log \left(\frac{| \, \mathbf{A} \, |}{| \, \mathbf{B} \, |} \right) \right). \tag{7.6.6}$$

where $\mathbf{A}$ and $\mathbf{B}$ are two positive-definite matrices of equal size $K \times K$.

The main idea behind using this divergence is to measure how much power one can expect an optimal detector to gain if it has to discriminate between a multivariate Gaussian using the true covariance Σ^{dev} (here estimated by Monte-Carlo simulation) and the multivariate Gaussian using our estimate of the covariance $\Sigma^{\mathbf{H}}$. The KL-divergence bounds the power of the optimal detector – see Theorem 2.3.10 – and as such, if it is close to zero, we can conclude that using our approximation would not hurt our steganographic scheme since the steganalyst would not gain much discriminatory power if the steganographer were to use our estimation method.

The results are presented in Table 7.2. The covariance matrix Σ^{dev} obtained for each pipeline are also presented in Figure 7.5-7.7. As to be

	Error		
Pipeline	$\|\|\Sigma^{\text{dev}} - \Sigma^{\mathbf{H}}\|\|_F$	$\|\|\Sigma^{\text{dev}} - \Sigma^{\mathbf{H}}\|\|_\infty$	$\|\|\Sigma^{\text{dev}} - \Sigma^{\mathbf{H}}\|\|_{KL}$
Linear	$1.7 \cdot 10^{-5}$	$1.5 \cdot 10^{-6}$	$1.2 \cdot 10^{-5}$
Bosslike	$4.0 \cdot 10^{-1}$	$1.9 \cdot 10^{-1}$	$1.8 \cdot 10^{-2}$
ALASKA	$1.1 \cdot 10^{+2}$	$8.2 \cdot 10^{+1}$	$6.1 \cdot 10^{+3}$

Table 7.2: Different measures of error of estimation between the covariance estimated with Monte Carlo simulations and the covariance estimated with the method presented in this chapter.

expected, the errors for the linear pipeline are negligible, in particular, even an optimal detector would not gain any significant power if a steganographer were to use $\Sigma^{\mathbf{H}}$ instead of Σ^{dev}.

The errors for the Bosslike pipeline are still quite low though we can expect an optimal detector in this case to gain at most 1.23% in terms of probability of detection which is not negligible. The larger error is due to at least two factors. The first is that, as shown in Section 6.1.1, the PPG demosaicking algorithm is neither linear nor stationary. Consequently, $\mathbf{H}$ is not a perfect representation of the processing pipeline. The second, more subtle factor, is that we used 74×74 RAW macro-blocks to estimate $\mathbf{H}$, that is, we used macro-blocks of size $8 \cdot 9 \times 8 \cdot 9$ with an added margin of 1 row/columns in each direction. This margin is used to take into account the impact of the convolution windows at the borders of the macro-blocks. A margin of 1 is sufficient in the case of the Linear pipeline because the largest convolution kernel has a 3×3 window size. However, the Bosslike pipeline uses a 5×5 Lanczos kernel during its downsampling operation, leading to an inaccurate estimation of $\mathbf{H}$ on the borders. A better estimate would be obtained by using a margin of 2 instead.

The errors on the ALASKA pipeline are several orders of magnitude higher showing that the model cannot handle extreme deviations from our assumptions of linearity and stationarity.

7.7 Conclusion

In this chapter we proposed a model of the noise in natural images by taking into account (1) the heteroscedasticity of the noise in the RAW domain and (2) the processing pipeline under the linearity and stationarity assumptions of the previous chapter. This leads to a multivariate Gaussian model of (macro)-blocks of DCT coefficients which depends only on the camera, ISO parameter and processing pipeline thus providing a theoretical justification of our empirical definition of the source in Chapter 4.

However, our model does not automatically provide a model of dependencies between (macro)-blocks themselves. Therefore, we proposed two models of dependencies, the first, idealized, assumes (macro)-blocks to be independents whereas the second, closer to reality as was shown in [75], assumes blocks of DCT coefficients to be dependent only with direct neighboring blocks.

We then went on to provide two method of estimation for the covariances of each (macro)-block of DCT coefficients. The first method

assumes knowledge of both the RAW image and of the processing pipeline as a black box. It is a straightforward application of Eq 7.1.2 and is the simplest to implement. The second method only necessitates access to the processing pipeline as a black box and leverages a novel approximate model of the variances of the DC coefficients and their linear relationship with the variances of the AC coefficients.

We finally ended the chapter by validating our model by showing that the distance between a covariance estimated using Monte-Carlo simulation and a covariance estimated using our method is quite small as long as the pipeline does not diverge too much from the linearity and stationarity assumptions.

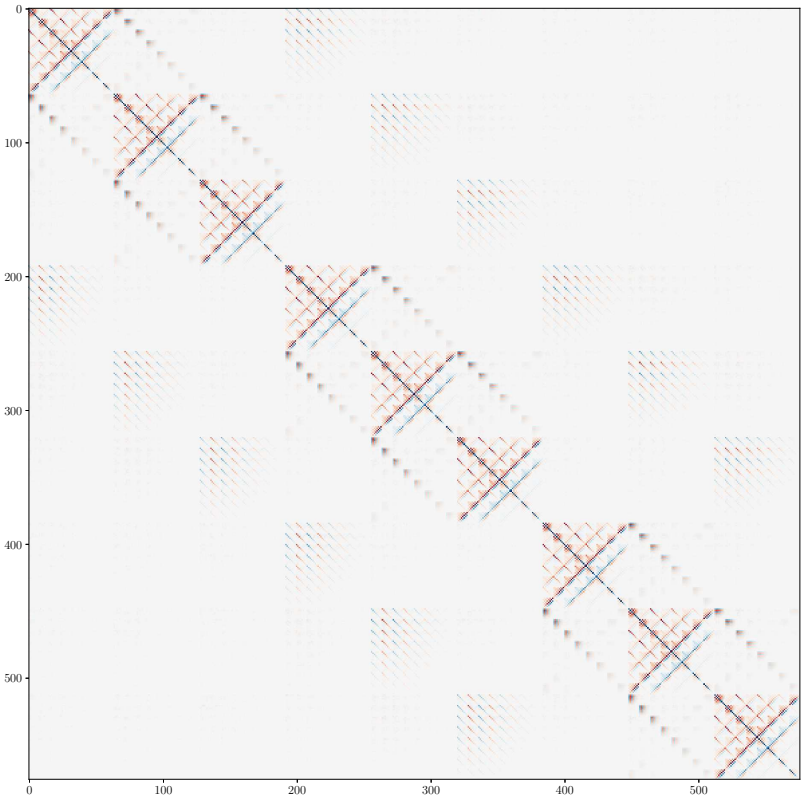


Figure 7.5: Covariance matrix Σ^{dev} of 24×24 blocks of DCT coefficients obtained for the Linear pipeline.

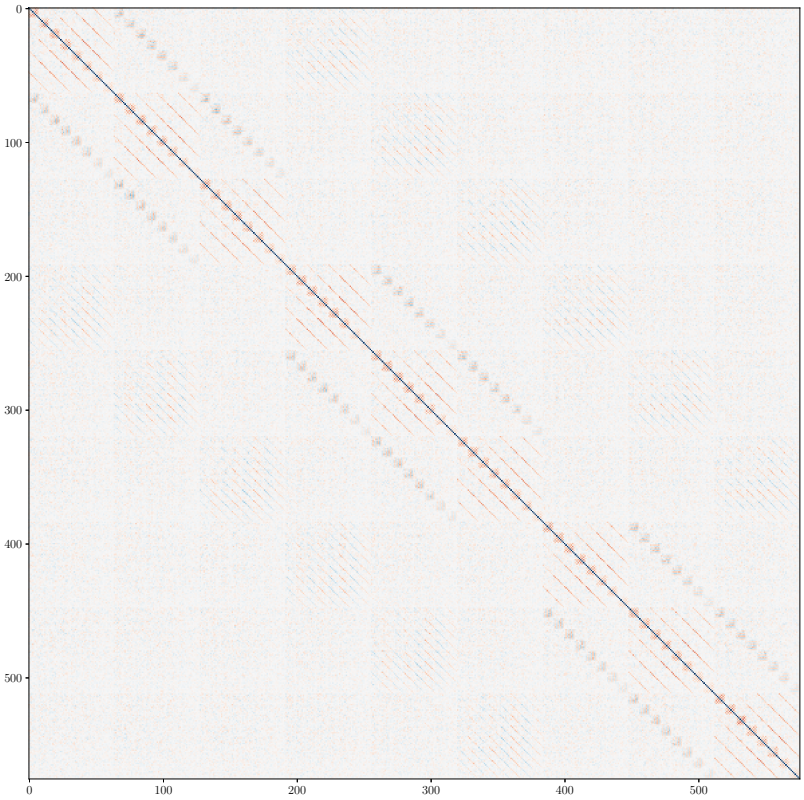


Figure 7.6: Covariance matrix Σ^{dev} of 24×24 blocks of DCT coefficients obtained for the Bosslike pipeline.

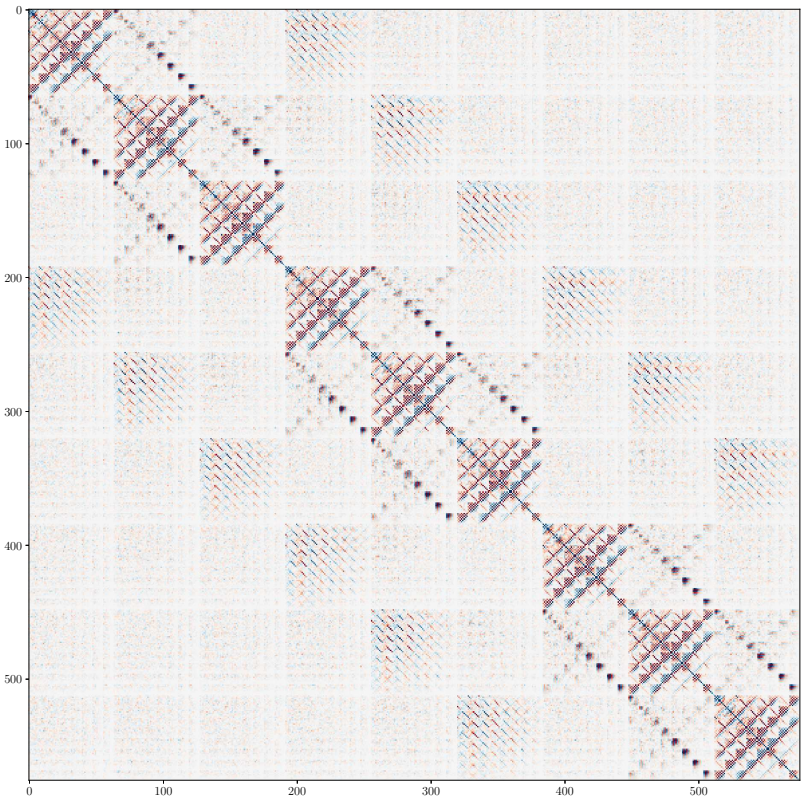


Figure 7.7: Covariance matrix Σ^{dev} of 24×24 blocks of DCT coefficients obtained for the ALASKA pipeline.

Part II

Steganography using a statistical model of the sensor noise

In the second part of this manuscript we leverage the theoretical model of the noise of natural images constructed in Part I to design steganographic algorithms in the JPEG domain able to provide guarantees of performance under a given set of hypotheses on the noise structure.

All chapters in this part follow an identical structure. First we describe the model of the cover and of the stego images which will be used by both the steganographer and the steganalyst. Then, we derive the optimal detector under these models and compute analytically, when possible, its statistical performances. Finally, we design the corresponding embedding scheme which minimizes the power of this optimal detector or an upper bound on this power.

This part is structured in such a way that we go from the simplest model requiring the least amount of knowledge about the cover image to the most complicated one requiring the full knowledge about the processing pipeline as well as access to the unquantized DCT coefficients:

- *Chapter 8* presents the design of JMiPOD, a steganographic algorithm which assumes that DCT coefficients are independent and follow quantized gaussian distributions with different variances. This algorithm does not assume access to the processing pipeline nor to the unquantized DCT coefficients of the cover image. This chapter is adapted from our contribution [14] which was itself a continuation of previous works in the spatial domain, namely MiPOD [69] and MG [29].

- *Chapter 9* presents the design of Gaussian Embedding, a steganographic algorithm which assumes that unquantized DCT coefficients are independent with different variances which can be estimated very precisely using the knowledge of the processing pipeline. It minimizes the power of the optimal detector in the continous domain under this model under a payload constraint in the quantized domain. Consequently it assumes both the knowledge of the processing pipeline as well as access to the unquantized DCT coefficients of the cover image. This Chapter is adapted from our two contributions [33, 32].

- *Chapter 10* presents the design of different variants of Multivariate Gaussian Embedding, a steganographic algorithm which fully leverages the noise model of Part I by assuming that macro-blocks of unquantized DCT coefficients follow a multivariate Gaussian distribution. It also minimizes the power of the optimal detector in the continous domain under this model under a payload constraint in the quantized domain. This Chapter is adapted from our two contributions [31, 30].

- *Chapter 11* provides numerical results and experimental comparisons between these algorithms as well as to the current state of the art in JPEG steganography.

8 Independent optimal steganography in the quantized domain

We start our study of the different possible methods of steganography in Gaussian noise with the most natural setting, where the steganographer is assumed to only have access to the cover image and nothing else. In this setting, the steganographer cannot use the processing pipeline and, as such, does not have access to the covariance of the noise. Furthermore, we will see that having access only to the quantized values of the cover imposes a major assumption on the structure of the noise: the fine quantization limit assumption.

8.1 Cover and stego model

8.1.1 *Cover model*

Since we assumed that the steganographer does not have access to the processing pipeline, we have to use an approximation of the noise model derived in Chapter 7. Furthermore, this model did not take the quantization step into account, something we address in this subsection.

Without access to the covariance of the noise [1], we model the noise as independent, but not identically distributed, Gaussian random variables. That is, the i-th unquantized DCT coefficient, y_i of the image is such that:

$$y_i \sim \mathcal{N}\left(\mu_i, \sigma_i^2\right). \tag{8.1.1}$$

However, the precover is quantized to a cover and as such each quantized DCT coefficient z_i follows a quantized Gaussian distribution:

$$\begin{aligned} z_i &= \text{round}\left(y_i\right), && (8.1.2)\\ &\sim \mathbf{N}\left(\mu_i, \sigma_i^2\right), && (8.1.3) \end{aligned}$$

with probability mass function:

$$\mathrm{f}_{\mu_i,\sigma_i}\left(x\right) = \Phi\left(\frac{x-\mu_i+0.5}{\sigma_i}\right) - \Phi\left(\frac{x-\mu_i-0.5}{\sigma_i}\right), x \in \mathbb{Z}. \tag{8.1.4}$$

Now such a distribution is quite difficult to manipulate in practice. The usual strategy to deal with this difficulty in the literature is to use the so-called fine quantization limit assumption [16, 29, 69, 15].

The fine quantization limit assumption states that the quantizer step is far smaller than the variance. In our case, the quantizer step is always 1[2]. This means that:

[1] The literature concerning the estimation of signal-dependent multivariate noise is very sparse and always make strong assumptions on either the information available to build the model or the structure of the covariance [3, 57].

[2] Without loss of generality, we assume in our model that σ_i^2 is the variance after the division by $Q_{k,l}$ in Eq (6.1.29) but before the rounding operation. The rounding operation does not multiply the resulting coefficient by $Q_{k,l}$, so the quantization step is always 1 in our model.

$$\sigma_i^2 \gg 1 \tag{8.1.5}$$

The analysis in [82, Chapter 4] shows that under this assumption, the pmf of z_i is well approximated by a Gaussian distribution such that:

$$z_i \approx \mathcal{N}\left(\mu_i, \sigma_i^2 + \frac{1}{12}\right) \tag{8.1.6}$$

$$\mathrm{f}_{\mu_i,\sigma_i}(x) \simeq \Phi\left(\frac{x-\mu_i}{\sqrt{\sigma_i^2+\frac{1}{12}}}\right). \tag{8.1.7}$$

The correction of $\frac{1}{12}$ of the variance is known from the so-called Sheppard's corrections, with a proof which can be found in [45]. In order to simplify the notation in the rest of this chapter, we omit this term and assume it to be included in σ_i^2

8.1.2 *Stego model*

In the quantized domain, $(2q+1)$-ary embedding is a popular choice with ternary embedding an excellent compromise between coding efficiency and security performance.

By $(2q+1)$-ary embedding we mean embedding using an alphabet with symbols taking integers value from -q to q

For our purpose, we consider that the steganographer uses a ternary alphabet $\mathcal{A} = \{-1, 0, +1\}$ where the probability β_i of adding $+1$ to the i-th cover element is equal to the probability of adding -1. Noting that the probability of adding 0 to the i-th cover element is consequently equal to $1-2\beta_i$, the pmf of the i-th stego element ζ_i is a mixture of quantized Gaussians:

$$\begin{aligned}\mathrm{f}_{\mu_i,\sigma_i,\beta_i}(x) = (1-2\beta_i)\,\mathrm{f}_{\mu_i,\sigma_i}(x) \\ + \beta_i\left(\mathrm{f}_{\mu_i,\sigma_i}(x+1) + \mathrm{f}_{\mu_i,\sigma_i}(x-1)\right),\end{aligned} \tag{8.1.8}$$

and we denote the distribution having this pmf as $\mathcal{N}_s$ so that we can write the stego object as:

$$\zeta_i \sim \mathcal{N}_s\left(\mu_i, \sigma_i^2, \beta_i\right). \tag{8.1.9}$$

Equation (8.1.8) means that if $\zeta_i = x$, then with probability $1-2\beta_i$, the value of the DCT coefficient before embedding was also x (no embedding); with probability β its value was either $x-1$ (added $+1$) or $x+1$ (added -1).

8.2 Optimal detector

Following the setting described in Section 2.3, we assume that the steganalyst has access to $\boldsymbol{\sigma}$, $\boldsymbol{\mu}$ and to $\boldsymbol{\beta}$. A sample image $\boldsymbol{\varsigma}$ is then presented to the steganalyst and they must choose between the two hypotheses:

$$\begin{cases}\mathcal{H}_0: & \{\varsigma_i \sim \mathcal{N}_s\left(\mu_i, \sigma_i^2, \mathbf{0}\right)\}, \forall i \\ \mathcal{H}_1: & \{\varsigma_i \sim \mathcal{N}_s\left(\mu_i, \sigma_i^2, \beta_i\right)\}, \forall i\end{cases} \tag{8.2.1}$$

The following theorem provides the optimal detector and its asymptotic statistical performance under this setting:

Theorem 8.2.1. *Under the settings of this chapter, the following holds:*

1. *The test maximizing the power function P_D for a prescribed false P_{FA} is the likelihood ratio test (LRT):*

$$\Lambda(\boldsymbol{\varsigma}, \boldsymbol{\sigma}, \boldsymbol{\beta}) = \sum_{i=1}^{n} \log\left(\frac{\mathrm{f}_{\mu_i,\sigma_i,\beta_i}(\boldsymbol{\varsigma})}{\mathrm{f}_{\mu_i,\sigma_i}(\boldsymbol{\varsigma})}\right) \underset{\mathcal{H}_1}{\overset{\mathcal{H}_0}{\lessgtr}} \tau \tag{8.2.2}$$

   where τ is a threshold fixed so that $\mathbb{P}[\Lambda > \tau \mid \mathcal{H}_0] = \alpha_0$, where α_0 is the chosen probability of false alarm.

2. *The limit distribution of $\Lambda - \mathbb{E}_{\mathcal{H}_0}[\Lambda]$ as $n \to \infty$ is given by:*

$$\Lambda(\boldsymbol{\varsigma}, \boldsymbol{\sigma}, \boldsymbol{\beta}) - \mathbb{E}_{\mathcal{H}_0}[\Lambda] \rightsquigarrow \begin{cases} \mathcal{N}\left(0, \sum_{i=1}^{n} \frac{2\beta_i^2}{\sigma_i^4}\right) & \text{under } \mathcal{H}_0 \\ \mathcal{N}\left(\sum_{i=1}^{n} \frac{2\beta_i^2}{\sigma_i^4}, \sum_{i=1}^{n} \frac{2\beta_i^2}{\sigma_i^4} + \mathcal{O}\left(\sigma_i^{-6}\right)\right) & \text{under } \mathcal{H}_1 \end{cases} \tag{8.2.3}$$

3. *Under the asymptotic regime, the power function P_D can be computed as:*

$$P_D(\alpha_0) = Q\left(Q^{-1}(\alpha_0) - \sqrt{\sum_{i=1}^{n} \frac{2\beta_i^2}{\sigma_i^4}}\right) \tag{8.2.4}$$

Proof. (1) follows directly from Theorem 2.3.5 combined with Proposition 2.3.6. (2) and (3) are proved in Appendix 8.A. □

The third point can be interpreted as the fact that areas with high variance don't contribute much to the power of the detector. This can be intuitively understood since the values of DCT coefficients in such areas are difficult to predict and are, as such, the most secure for the steganographer. We leverage precisely this observation in the next section where we finally design the first steganographic algorithm of this manuscript.

8.3 Embedding

The steganographer's problem is to minimize the power of the LRT under a given payload constraint R. This can be formulated as the following optimization problem:

$$\begin{cases} \min_{\boldsymbol{\beta}} & = \sum_{i=1}^{n} \frac{2\beta_i^2}{\sigma_i^4} \\ R & = \sum_{i=1}^{n} H_3(\beta_i) \end{cases} \tag{8.3.1}$$

with:

$$H_3(x) = -2x\log(x) - (1-2x)\log(1-2x). \tag{8.3.2}$$

This problem can be solved using the method of Lagrange multipliers. The Lagrangian of Eq (8.3.1) can be written as:

$$\mathcal{L} = \sum_{i=1}^{n} \frac{2\beta_i^2}{\sigma_i^4} + \lambda\left(R - \sum_{i=1}^{n} H_3(\beta_i)\right). \tag{8.3.3}$$

We then compute the parameters β_i and λ which make $\frac{\partial\mathcal{L}}{\partial\beta_i}$ null:

$$\frac{\partial\mathcal{L}}{\partial\beta_i} = 0 \iff \frac{4\beta_i}{\sigma_i^4} + 2\lambda\log\left(\frac{\beta_i}{1-2\beta_i}\right) = 0, \tag{8.3.4}$$

which needs to be solved numerically for each β_i. In practice, a interval halving search is performed on λ to find the resulting β_i which allow the payload constraint to be satisfied.

8.4 Variance estimation without the processing pipeline

8.4.1 *The noise-content tradeoff*

In the absence of the processing pipeline, the techniques presented in Chapter 7 cannot be used to estimate the variance. Consequently, an estimator of the variance must be chosen to perform the embedding of hidden data as described in Eq (8.3.1). A counter-intuitive fact to take into account is that the most accurate estimator of the variance might not, in fact, lead to the best security performance for this algorithm. This has been first discussed in [70, 69] and we have provided a long experimental study of this fact in [32] where we clearly showed that, against some steganalyzers, the use of perfect estimates of the variance is sub-optimal with respect to using estimates from more crude estimators.

In particular, we showed the existence of a noise-content trade-off in steganography: algorithms that use only perfect estimates of the variance don't leverage the fact that steganalyzer cannot perfectly model the content of an image. Such algorithms have consequently subpar performance against algorithms such as J-UNIWARD which precisely leverages such weakness of steganalyzers.

However, it is possible to integrate this local difficulty of modeling content by using an imperfect variance estimator that only uses the cover image. The estimation errors would typically occur in "non-smooth" areas and result in largely overestimated variance – indicating to the steganographic algorithm that these zones are "safe" to embed in.

8.4.2 *Wiener filtering and local approximation with a parametric model*

We perform the estimation of the variance in the spatial domain. That is, we first decompress the cover image $\mathbf{z}$ into the image $\tilde{\mathbf{z}}$. When referring to these images in their matrix form, we write $\mathbf{Z}$ and $\tilde{\mathbf{Z}}$ respectively.

Please note that the methodology of this subsection is mostly taken from the first work on the subject in steganography, that is [70]. Indeed, except for the change of some parameters, the overall methodology has been kept the same across all works based on the setting of this chapter [70, 69, 15, 14]. We still provide it in this manuscript for completeness.

The first step of the variance estimation procedure is to separate the noise from the content of the image since we are mainly interested in estimating the variance of the noise. To do so, we compute at first a crude estimate of the mean using a 2×2 Wiener filter W on $\tilde{\mathbf{Z}}$:

$$\mathbf{R} = \tilde{\mathbf{Z}} - W\left(\tilde{\mathbf{Z}}\right), \tag{8.4.1}$$

with $\mathbf{R}$ the residual noise.

To improve the estimation of the content, we fit a parametric linear model to the residuals using their surrounding neighbors. To do so, we first choose a neighborhood size p. For the i-th residual of the image, we note the $p \times p$ block centered on this residual, reshaped into a column vector of size p^2, as $\mathbf{r}_i$.

Using a linear parametric model for this residual amounts to write:

$$\mathbf{r}_i = \mathbf{G}\mathbf{a}_i + \boldsymbol{\xi}_i, \tag{8.4.2}$$

$$\boldsymbol{\xi}_i \sim \mathcal{N}\left(0, \operatorname{diag}\left(\boldsymbol{\varrho}_i^2\right)\right), \tag{8.4.3}$$

where $\mathbf{G}$ is a $p^2 \times q$ matrix defining the parametric model and $\mathbf{a}_i$ a vector of size q representing the actual coefficients of this model for the particular residual block $\mathbf{r}_i$.

Our goal here is to estimate ϱ_i^2. Whatever the chosen parametric model $\mathbf{G}$, under the assumption that all the variances in a given $\boldsymbol{\xi}$ are constant, the Gauss-Markov theorem [55, Theorem 4.12] states that the best linear unbiased estimation of $\mathbf{a}_i$ is given by:

$$\hat{\mathbf{a}}_i = \left(\mathbf{G}^T\mathbf{G}\right)^{-1}\mathbf{G}^T\mathbf{r}_i. \tag{8.4.4}$$

Using the well-known maximum likelihood estimator of Gaussian random variable, we obtain the individual variance estimates $\hat{\varrho}_i^2$:

$$\begin{aligned}\hat{\varrho}_i^2 &= \frac{||\mathbf{r}_i - \mathbf{G}\hat{\mathbf{a}}_i||_2^2}{p^2 - q} \\ &= \frac{||\mathbf{r}_i - \mathbf{G}\left(\mathbf{G}^T\mathbf{G}\right)^{-1}\mathbf{G}^T\mathbf{r}_i||_2^2}{p^2 - q}.\end{aligned} \tag{8.4.5}$$

From the pioneering paper [70] to the most recent [14], It appears that the empirical choice that leads to highest empirical security against current detectors is a trigonometric polynomial model containing l monomials:

$$\begin{aligned}G = &(\mathbf{1}, \cos(\mathbf{U}), \cos\left(\mathbf{U}^\mathbf{T}\right), \cos(\mathbf{U}) \cdot \cos\left(\mathbf{U}^\mathbf{T}\right), \cos(\mathbf{2U}), \\ &\cos\left(2\mathbf{U}^\mathbf{T}\right), \cos(2\mathbf{U}) \cdot \cos\left(2\mathbf{U}^\mathbf{T}\right), \cos(4\mathbf{U}), \ldots, \\ &\cos(l\mathbf{U}), \cos\left(l\mathbf{U}^\mathbf{T}\right), \cos(l\mathbf{U}) \cdot \cos\left(l\mathbf{U}^\mathbf{T}\right), \cos(l\mathbf{2U}))\end{aligned} \tag{8.4.6}$$

with:

$$\mathbf{U}_{i,j} = \frac{\pi(2j-1)}{2p}, \forall 1 \leq i, j \leq p. \tag{8.4.7}$$

For the moment, we only estimated the variance ϱ_i^2 in the spatial domain. To obtain the variances in the DCT domain, we leverage the

linearity of the DCT transform – see Eq (6.1.26). For the k-th 8×8 block of DCT coefficient, the block of variance $\boldsymbol{\sigma}_k^2$ in the DCT domain is given by:

$$\hat{\boldsymbol{\sigma}}_k^2 = \text{diag}\left(\mathbf{H}^{DCT}\text{diag}\left(\boldsymbol{\varrho_k}^2\right)\left(\mathbf{H}^{DCT}\right)^T\right), \tag{8.4.8}$$

where $\mathbf{H}^{DCT}$ is the DCT transform represented as a 64×64 matrix. Once every $\hat{\sigma}_i^2$ is estimated we compute the Fisher Information I_i :

$$\mathbf{I}_k = \hat{\boldsymbol{\sigma}}_k^{-4}. \tag{8.4.9}$$

We call this quantity the Fisher Information as it can be shown with simple calculations that $\mathbb{E}\left[\left(\frac{\partial f_{\mu_i,\sigma_i,\beta_i}}{\partial \beta_i}\right)^2\right]_{|\beta_i=0} = \frac{1}{\sigma_i^4}$

It has been observed that the security of the steganographic algorithm can be highly improved if the Fisher Information is smoothed using an ad-hoc filter:

$$\hat{\mathbf{I}} = \mathbf{I} \star \mathbf{W}. \tag{8.4.10}$$

We have found experimentally the following parameters to lead to good performance in the JPEG domain:

$$\mathbf{W} = \frac{1}{20}\begin{pmatrix} 1 & 3 & 1 \\ 3 & 4 & 3 \\ 1 & 3 & 1 \end{pmatrix}. \tag{8.4.11}$$

8.5 Performance evaluation

Since the algorithm presented in this chapter was designed for quite a different setting than the algorithms of the following chapters (no side-information and no use of the processing pipeline), we provide the performance evaluation of J-MiPOD in this section whereas other algorithms will be evaluated in Chapter 11. We compare the proposed method with the two state-of-the-art of non side-informed JPEG steganography: UERD [39] and J-UNIWARD [42].

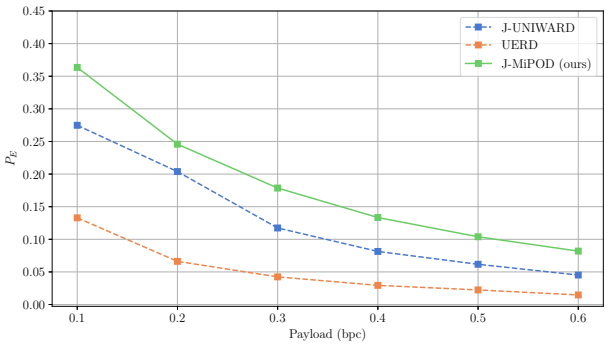
We have used two datasets: BossBase [5] and BOWS[3]. They are both made of $10,000$ grayscale images of size 512×512. As explained in Chapter 4 this dataset lacks the diversity of real-world datasets since all images have been processed in the same way. Therefore, to compare the security of these embedding schemes under more realistic conditions we have also used the recent ALASKA dataset [12]. The version we used is made of $80,000$ JPEG grayscale images of size 512×512. All the images from this dataset have been processed differently using a randomized process.

[3] Dataset from the watermarking competition organized by P. Bas and T. Furon – see http://bows2.ec-lille.fr/

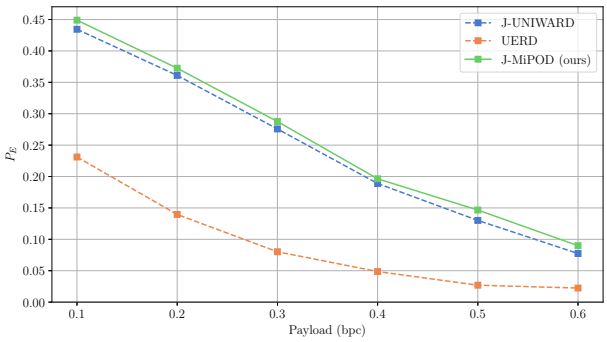
For steganalysis, we used Efficient-Net-b0 [76] as it has been shown to reach state of the art performance during the ALASKA2 competition. Results are presented in Figure 8.1-8.2.

8.6 Conclusion

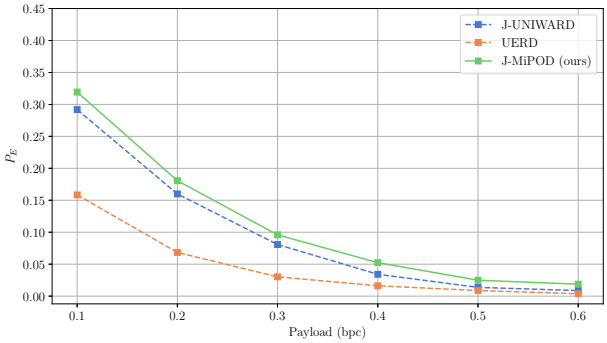
In this chapter, we presented the JMiPOD algorithm which is based on a very simplified model of the sensor noise in the quantized domain. We have used it to illustrate the main strategy of statistically based embedding schemes as presented in Section 2.3.



(a) QF100

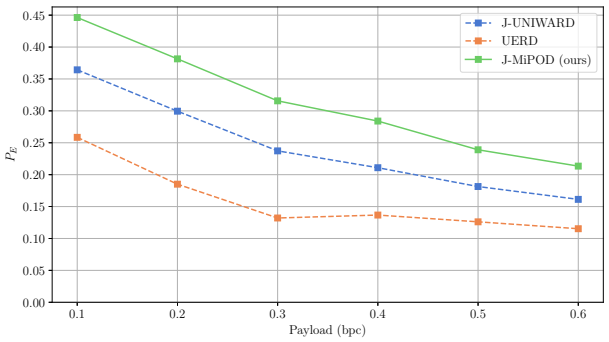


(b) QF95

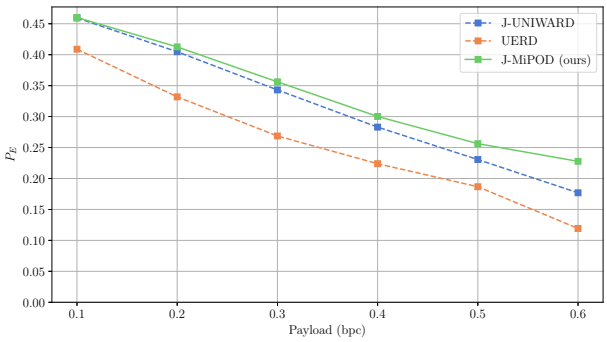


(c) QF75

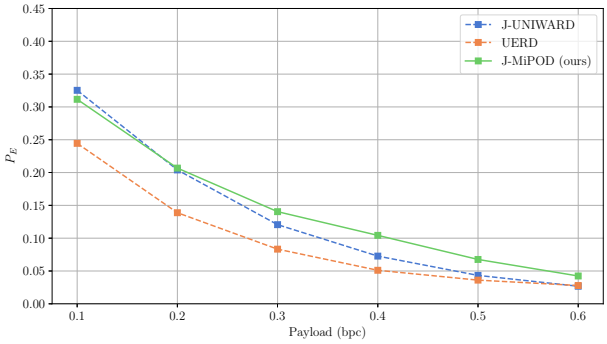
Figure 8.1: P_E as function of payload size for BossBase+BOWS steganalyzed using Efficient-Net-b0. Payload



(a) QF100



(b) QF95



(c) QF75

Figure 8.2: P_E as function of payload size for ALASKA steganalyzed using Efficient-Net-b0. Payload is in bits per non zero AC coefficients.

The algorithm is based on an independent quantized Gaussian model of DCT coefficients in the JPEG domain. The stego model uses a mixture of quantized Gaussian distribution, modeling the impact of ternary embedding. The power of the optimal detector under these models is shown to depend only on the variance of the DCT coefficients. JMiPOD then minimizes the power of this detector to locate the best DCT coefficients for embedding.

We provided a methodology to estimate the variance of DCT coefficients when only the JPEG file is available and showed that the method performed better than the state of the art of non side-informed JPEG steganography.

In the following chapters, we finally begin to leverage the model of the sensor noise developed in the previous part of this manuscript and combine it with the strategy outlined in this chapter.

Appendices

8.A Limit distribution of the LRT and asymptotic performance

Without loss of generality, we assume that $\mu_i = 0, \forall i$ for the rest of this appendix.

This appendix is largely adapted from the excellent presentation of the appendix of the original MiPOD paper [69]

We have that:

$$\begin{aligned}\Lambda(\boldsymbol{\varsigma}, \boldsymbol{\sigma}, \boldsymbol{\beta}) &= \sum_{i=1}^{n} \log\left(\frac{\mathrm{f}_{\sigma_i,\beta_i}(\boldsymbol{\varsigma})}{\mathrm{f}_{\sigma_i}(\boldsymbol{\varsigma})}\right) \\ &= \sum_{i=1}^{n} \log\left(\frac{(1-2\beta_i)\exp\left(\frac{\varsigma_i^2}{2\sigma^2}\right) + \beta_i\left(\exp\left(\frac{(\varsigma_i+1)^2}{2\sigma_i^2}\right) + \exp\left(\frac{(\varsigma_i-1)^2}{2\sigma_i^2}\right)\right)}{\exp\left(\frac{\varsigma_i^2}{2\sigma_i^2}\right)}\right) \\ &= \sum_{i=1}^{n} \log\left(1 - 2\beta_i + \beta_i\left(\exp\left(\frac{\varsigma_i - \frac{1}{2}}{\sigma_i^2}\right) + \exp\left(\frac{-\varsigma_i - \frac{1}{2}}{\sigma_i^2}\right)\right)\right)\end{aligned} \tag{8.A.1}$$

By using the fine quantization assumption, we have that $\sigma_i \gg 1$. We can then use the second-order Taylor approximation of the exponential function around 0:

$$\Lambda(\boldsymbol{\varsigma}, \boldsymbol{\sigma}, \boldsymbol{\beta}) \doteq \sum_{i=1}^{n} \log\left(1 + \beta_i\left(-\frac{1}{\sigma_i^2} + \frac{\varsigma_i^2 + \frac{1}{4}}{\sigma_i^4}\right)\right), \tag{8.A.2}$$

which we follow by the first-order Taylor approximation of $\log(1+x)$:

$$\Lambda(\boldsymbol{\varsigma}, \boldsymbol{\sigma}, \boldsymbol{\beta}) \doteq \sum_{i=1}^{n} \beta_i\left(-\frac{1}{\sigma_i^2} + \frac{\varsigma_i^2 + \frac{1}{4}}{\sigma_i^4}\right). \tag{8.A.3}$$

Using the invariance of the LRT under the addition of a constant we obtain the final expression:

$$\Lambda(\boldsymbol{\varsigma}, \boldsymbol{\sigma}, \boldsymbol{\beta}) = \sum_{i=1}^{n} \beta_i\left(-\frac{1}{\sigma_i^2} + \frac{\varsigma_i^2}{\sigma_i^4}\right). \tag{8.A.4}$$

The moments of Λ can be computed under both hypotheses. Under $\mathcal{H}_0$ we have:

$$\begin{aligned}\mathbb{E}_{\mathcal{H}_0}[\Lambda] &= \sum_{i=1}^{n} \beta_i \mathbb{E}_{\mathcal{H}_0}\left[\frac{\varsigma_i^2}{\sigma_i^4}\right] - \sum_{i=1}^{n} \frac{\beta_i}{\sigma_i^2} \\ &= \sum_{i=1}^{n} \frac{\beta_i}{\sigma_i^2} - \sum_{i=1}^{n} \frac{\beta_i}{\sigma_i^2} \\ &= 0.\end{aligned} \tag{8.A.5}$$

$$\begin{aligned}\mathrm{Var}_{\mathcal{H}_0}[\Lambda] &= \sum_{i=1}^{n} \beta_i^2 \mathrm{Var}_{\mathcal{H}_0}\left[\frac{\varsigma_i^2}{\sigma_i^4}\right] \\ &= \sum_{i=1}^{n} \frac{2\beta_i^2}{\sigma_i^4}.\end{aligned} \tag{8.A.6}$$

And conversely under $\mathcal{H}_1$:

$$\begin{aligned}
\mathbb{E}_{\mathcal{H}_1}[\Lambda] &= \sum_{i=1}^{n} \beta_i \frac{\mathbb{E}_{\mathcal{H}_1}\left[\varsigma_i^2\right]}{\sigma_i^4} - \sum_{i=1}^{n} \frac{\beta_i}{\sigma_i^2} \\
&= \sum_{i=1}^{n} \beta_i \frac{(1-2\beta_i)\mathbb{E}_{\mathcal{H}_0}\left[\varsigma_i^2\right] + \beta_i\left(\mathbb{E}_{\mathcal{H}_0}\left[(\varsigma_i+1)^2\right] + \mathbb{E}_{\mathcal{H}_0}\left[(\varsigma_i-1)^2\right]\right)}{\sigma_i^4} - \sum_{i=1}^{n} \frac{\beta_i}{\sigma_i^2} \\
&= \sum_{i=1}^{n} \beta_i \frac{(1-2\beta_i)\mathbb{E}_{\mathcal{H}_0}\left[\varsigma_i^2\right] + 2\beta_i\mathbb{E}_{\mathcal{H}_0}\left[\varsigma_i^2+1\right]}{\sigma_i^4} - \sum_{i=1}^{n} \frac{\beta_i}{\sigma_i^2} \\
&= \sum_{i=1}^{n} \beta_i \frac{(1-2\beta_i)\sigma_i^2 + 2\beta_i(\sigma_i^2+1)}{\sigma_i^4} - \sum_{i=1}^{n} \frac{\beta_i}{\sigma_i^2} \\
&= \sum_{i=1}^{n} \frac{\beta_i\sigma_i^2 + 2\beta_i^2}{\sigma_i^4} - \sum_{i=1}^{n} \frac{\beta_i}{\sigma_i^2} \\
&= \sum_{i=1}^{n} \frac{2\beta_i^2}{\sigma_i^4}.
\end{aligned} \tag{8.A.7}$$

$$\begin{aligned}
\mathrm{Var}_{\mathcal{H}_1}[\Lambda] &= \sum_{i=1}^{n} \beta_i^2 \mathrm{Var}_{\mathcal{H}_1}\left[\frac{\varsigma_i^2}{\sigma_i^4}\right] \\
&= \sum_{i=1}^{n} \beta_i^2 \frac{\mathbb{E}_{\mathcal{H}_1}\left[\varsigma_i^4\right] - \mathbb{E}_{\mathcal{H}_1}\left[\varsigma_i^2\right]^2}{\sigma_i^8} \\
&= \sum_{i=1}^{n} \beta_i^2 \frac{2\beta_i + 3\sigma_i^4 + 12\beta_i\sigma_i^2 - \sigma_i^4 - 4\beta_i^2 - 2\beta_i\sigma_i^2}{\sigma_i^8} \\
&= \sum_{i=1}^{n} \frac{2\beta_i^2}{\sigma_i^4} + \mathcal{O}\left(\sigma_i^{-6}\right) \\
&\simeq \mathrm{Var}_{\mathcal{H}_0}[\Lambda].
\end{aligned} \tag{8.A.8}$$

Since we assume DCT coefficients to be independent we can directly use Corollary 9.2.1 to obtain the asymptotic performance of the LRT:

$$\begin{aligned}
P_D &= Q\left(Q^{-1}(PFA) + \frac{0 - \sum_{i=1}^{n} \frac{2\beta_i^2}{\sigma_i^4}}{\sqrt{\sum_{i=1}^{n} \frac{2\beta_i^2}{\sigma_i^4}}}\right) \\
&= Q\left(Q^{-1}(PFA) - \sqrt{\sum_{i=1}^{n} \frac{2\beta_i^2}{\sigma_i^4}}\right).
\end{aligned} \tag{8.A.9}$$

9 Independent Gaussian steganography minimizing statistical detectability

9.1 Cover and stego model

In this chapter, the steganographer assumes that all the unquantized DCT coefficients y_i are independent and that they follow a Gaussian distribution such that:

$$y_i \sim \mathcal{N}\left(\mu_i, \sigma_i^2\right), \tag{9.1.1}$$

with the pdf denoted as p_{σ_i}.

However, contrary to Chapter 8, we now model the prestego signal, that is, the stego signal in the continuous domain before quantization. In this chapter, the steganographer chooses to use a prestego signal s_i, which is added to y_i such that:

$$s_i \sim \mathcal{N}\left(0, \epsilon_i^2\right). \tag{9.1.2}$$

This choice of distribution, though we do not prove its optimality, is motivated by two nice properties of the Gaussian distribution. First, the sum of two Gaussian random variable is also a Gaussian random variable which makes the mathematical derivations more tractable. Secondly, for a given expectation and variance, the Gaussian distribution is the maximum entropy distribution in the continuous domain [17], we can thus expect the distribution to maximize the embedded payload for a given power of the LRT.

We then define the prestego image element γ_i as the stego image element before rounding:

$$\gamma_i = y_i + s_i \tag{9.1.3}$$

$$\sim \mathcal{N}\left(\mu_i, \sigma_i^2 + \epsilon_i^2\right), \tag{9.1.4}$$

with pdf denoted as q_{σ_i,ϵ_i}.

9.2 Optimal detector

Under our setting, the goal of the steganalyst is to construct a test in the continuous domain. We assume they have access to $\boldsymbol{\mu}$ $\boldsymbol{\sigma}$, $\boldsymbol{\epsilon}$ as well as to the sample in the continuous domain $\boldsymbol{\xi}$. The steganalyst must choose between two hypotheses:

$$\begin{cases} \mathcal{H}_0: & \{\xi_i \sim \mathcal{N}\left(\mu_i, \sigma_i^2\right)\}, \forall i \\ \mathcal{H}_1: & \{\xi_i \sim \mathcal{N}\left(\mu_i, \sigma_i^2 + \epsilon_i^2\right)\}, \forall i \end{cases} \tag{9.2.1}$$

The following theorem provides the form of the optimal test under this setting as well as its performance:

Theorem 9.2.1 (Independent Gaussian LRT)**.** *Under the settings of this chapter, the following holds:*

1. *The test maximizing the power function P_D for a prescribed false α_0 is the log-likelihood ratio test (LRT):*

$$\Lambda(\boldsymbol{\xi}, \boldsymbol{\sigma}, \boldsymbol{\beta}) = \sum_{i=1}^{n} \log\left(\frac{q_{\sigma_i,\epsilon_i}}{p_{\sigma_i}}\right) \underset{\mathcal{H}_1}{\overset{\mathcal{H}_0}{\lessgtr}} \tau \tag{9.2.2}$$

   where τ is a threshold fixed so that $\mathbb{P}[\Lambda > \tau \mid \mathcal{H}_0] = \alpha_0$, where α_0 is the chosen probability of false alarm.

2. *The limit distribution of Λ as $n \to \infty$ is given by:*

$$\Lambda(\boldsymbol{\xi}, \boldsymbol{\sigma}, \boldsymbol{\epsilon}) \rightsquigarrow \begin{cases} \mathcal{N}\left(-\mathrm{D}_{KL}\left(p_{\sigma_i} \,||\, q_{\sigma_i,\epsilon_i}\right), \sum_{i=1}^{n} \frac{\epsilon_i^4}{2(\sigma_i^2+\epsilon_i^2)^2}\right) & \text{under } \mathcal{H}_0 \\ \mathcal{N}\left(\mathrm{D}_{KL}\left(q_{\sigma_i,\epsilon_i} \,||\, p_{\sigma_i}\right), \sum_{i=1}^{n} \frac{\epsilon_i^4}{2\sigma_i^4}\right) & \text{under } \mathcal{H}_1 \end{cases} \tag{9.2.3}$$

3. *Under the asymptotic regime and assuming $\sigma_i^2 \gg \epsilon_i^2$ for all i, the power function P_D can be computed as:*

$$P_D(\alpha_0) = Q\left(Q^{-1}(\alpha_0) - \sqrt{\sum_{i=1}^{n} \frac{\epsilon_i^4}{2\sigma_i^4}}\right) \tag{9.2.4}$$

Proof. (1) follows directly from Theorem 2.3.5 combined with Proposition 2.3.6. (2) and (3) are proved in Appendix 9.A and 9.B respectively. □

9.3 Embedding

The embedding problem under the setting of this chapter is peculiar because we express the power of the LRT in the continuous domain but the final message must be embedded in the quantized domain since the image will have to be shared as compressed JPEG. Consequently, the payload constraint must be expressed in the quantized domain despite the power of the LRT being expressed in the continuous domain. This results in the following optimization problem:

$$\begin{cases} \min_{\boldsymbol{\epsilon}} & \sum_{i=1}^{n} \frac{\epsilon_i^4}{2\sigma_i^4} \\ R = & \sum_{i=1}^{n} \sum_{k\in\mathbb{Z}} \beta_i^k \log \beta_i^k \end{cases} \tag{9.3.1}$$

where β_i^k is the probability of modifying the i-th DCT coefficient by $+k$ during embedding. It is computed as:

$$\beta_i^k = \Phi\left(\frac{k - r_i + 0.5}{\epsilon_i}\right) - \Phi\left(\frac{k - r_i - 0.5}{\epsilon_i}\right), \tag{9.3.2}$$

where $r_i = y_i - [y_i]$ is the rouding error of the i-th DCT coefficient.

Now, Eq (9.3.1) is a difficult system to solve directly but we can use the following proxy problem:

$$\begin{cases} \min_{\boldsymbol{\epsilon}} & \displaystyle\sum_{i=1}^{n} \frac{\epsilon_i^4}{2\sigma_i^4} \\ R^* = & \frac{1}{2}\displaystyle\sum_{i=1}^{n} \log\left(2\pi e \epsilon_i^2\right) \end{cases} \tag{9.3.3}$$

where R^* is the entropy of the signal in the continuous domain. Now observe that both R^* and R are increasing in ϵ_i^2. Consequently, we only have to find R^* such that $\sum_{i=1}^{n}\sum_{k\in\mathbb{Z}} \beta_i^k \log \beta_i^k = R$, with each β_i^k directly depending on R^* through ϵ_i^2.

We can solve Eq (9.3.3) using the method of Lagrange multipliers:

$$\mathcal{L} = \sum_{i=1}^{n} \frac{\epsilon_i^4}{2\sigma_i^4} + \lambda\left(R^* - \frac{1}{2}\sum_{i=1}^{n} \log\left(2\pi e \epsilon_i^2\right)\right) \tag{9.3.4}$$

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \epsilon_i} = 0, \iff & \frac{2\epsilon_i^3}{\sigma_i^4} + \frac{\lambda}{\epsilon_i} = 0 \\ \iff & \epsilon_i^2 = \sqrt{\frac{\lambda}{2}}\sigma_i^2. \end{aligned} \tag{9.3.5}$$

Finally we obtain the expression of λ by plugging in ϵ_i^2 in the payload constraint in the continuous domain:

$$\lambda = \exp\left(\frac{4R^*}{n} - \frac{4}{n}\sum_{i=1}^{n} \log\left(\sigma_i\right) - 2\log\left(2\pi e\right) + \log\left(2\right)\right). \tag{9.3.6}$$

Note then that the optimal variance of the prestego signal is proportional to the variance of the cover noise. Finding the optimal R can be done through a simple halving interval search on λ until the payload constraint *in the quantized domain* is met. This allows solving the original optimization problem in Eq (9.3.1). We provide the full algorithm in Algorithm 3.

9.4 Validation and limitations of the method

We end this chapter by studying how our steganographic algorithm behaves against the likelihood ratio test under different model assumptions on the cover model. First, we test the performance of the algorithm on synthetic images which follow the model of Section 9.1 and verify that it matches the performance predicted by Theorem 9.2.1. We then go on to test the method on natural images in order to show the limitations of not taking correlations between DCT coefficients into account.

9.4.1 *Synthetic images*

In order to validate the analysis of Section 9.2, we build a dataset made out of synthetic images with the parameters of the noise chosen as to

Algorithm 3: Gaussian Embedding

Data: $\mathbf{y}$: Precover, $\boldsymbol{\sigma}^2$: Cover noise variances, R: Payload size (in nats)
Result: $\mathbf{s}$: Prestego signal solving Eq (9.3.1)
$n \leftarrow \text{size}(y)$;
$\mathbf{r} \leftarrow \mathbf{y} - \text{round}(\mathbf{y})$;
// Interval bisection on R^* until the payload constraint is met
$l, r \leftarrow \text{InitializeIntervalBounds}()$;
// Compute λ using Eq (9.3.6) for $R^* = l$.
$\lambda \leftarrow \text{ComputeLambda}\,(l)$;
// Compute entropy using Eq (9.3.1) and Eq (9.3.2).
$fl \leftarrow \text{ComputeEntropy}\,(\lambda) - R$;
$\lambda \leftarrow \text{ComputeLambda}\,(r)$;
$fr \leftarrow \text{ComputeEntropy}\,(\lambda) - R$;
while error $\neq 0$ **do**
  $R^* \leftarrow \frac{l+r}{2}$;
  $\lambda \leftarrow \text{ComputeLambda}\,(R^*)$;
  error $\leftarrow \text{ComputeEntropy}\,(\lambda) - R$;
  if $\text{fl} \cdot \text{error} < 0$ **then**
    $r \leftarrow R^*$;
    $fr \leftarrow$ error;
  else
    $l \leftarrow R^*$;
    $fl \leftarrow$ error;
// Sample n independent gaussians with variance $\boldsymbol{\epsilon}^2$
$\mathbf{s} \leftarrow \mathcal{N}(0, \text{diag}\left(\sqrt{\frac{\lambda}{2}}\,\boldsymbol{\sigma}^2\right)$;

be identical to those of the camera used to capture the original image. For each dataset, we select a RAW image $\bar{\mathbf{x}}$ from BOSSBase which we then develop using the Linear pipeline, as defined in Section 7.6, to obtain the baseline image $\bar{\mathbf{y}}$. We then generate 1000 precover images $\mathbf{y}^{(l)}$ such that for all l:

$$y_i^{(l)} \sim \mathcal{N}\left(\mu_i, \sigma_i^2\right) \tag{9.4.1}$$

where $\mu_i = \bar{y}_i$ and σ_i^2 is the variance of $\bar{y}_i$ estimated using the method presented in Section 7.3. Notice that each $\mathbf{y}^{(l)}$ is simply the baseline image to which was added different realizations of a Gaussian noise.

We then embed each image using the Gaussian Embedding algorithm for different payload to obtain the prestego images $\boldsymbol{\gamma}^{(l)}$:

$$\boldsymbol{\gamma}^{(l)} \sim \mathcal{N}\left(\mu_i, \sigma_i^2 + \epsilon_i\right). \tag{9.4.2}$$

We finally compute the LRT under the model presented in this chapter:

$$\Lambda\left(\boldsymbol{\xi}, \boldsymbol{\sigma}, \boldsymbol{\beta}\right) = \sum_{i=1}^{n} \log\left(\frac{q_{\sigma_i, \epsilon_i}}{p_{\sigma_i}}\right) \tag{9.4.3}$$

We present the ROC curve of the empirical LRT against the ROC curve of the theoretical LRT in Figure 9.1. As expected the perfor-

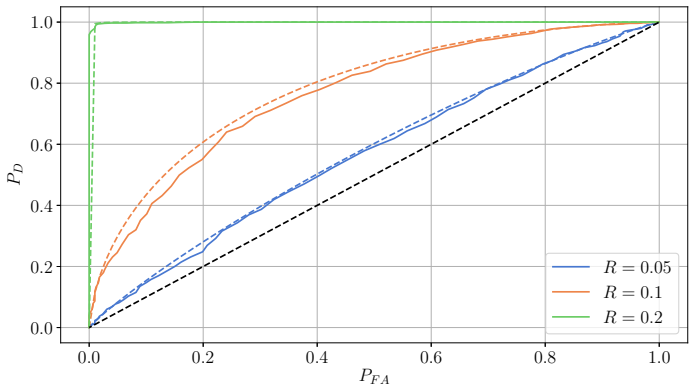


Figure 9.1: LRT on a dataset with synthetic independent Gaussian noise built with image 6551 in BossBase for different relative payloads R. The dotted lines represent the theoretical performance of the LRT whereas the full line represent the empirical performance.

mance of the empirical LRT match the expected performance for all the tested cameras, validating our theoretical analysis of the algorithm. We now go on to study how far the independent Gaussian model for natural images in order to highlight its limit and motivate the construction of the next chapter.

9.4.2 *Natural images under the Linear pipeline*

We repeat the experiment of the previous subsection, but this time, instead of generating noise directly in the developed domain, we generate noise in the RAW domain so that it follows all the operation of the processing pipeline. We first estimate the heteroscedastic model parameters c_1 and c_2 of the noise in the RAW domain of the RAW image $\bar{\mathbf{x}}$ using the method described in Section 5.3. We then generate 1000 noisy versions $\mathbf{x}^{(l)}$ such that:

$$x_i^l \sim \mathcal{N}\left(\bar{x}_i, c_1\bar{x}_i + c_2\right). \tag{9.4.4}$$

Notice that each noisy image should follow exactly the model of the noise of $\bar{\mathbf{x}}$ except that we use the baseline image as the mean value. We then process each image using the Linear pipeline to obtain the precover, with the l-th precover denoted as $\mathbf{y}^{(l)}$. From the analysis of Chapter 7, $\mathbf{y}^{(l)}$ should follow a multivariate Gaussian model:

$$\mathbf{y}_k^{(l)} \sim \mathcal{N}\left(\mathbf{H}\,\bar{\mathbf{x}}_k, \Sigma_k\right). \tag{9.4.5}$$

Note that for the Linear pipeline, the lattice model presented in Section 7.2 should be exact as is shown in [75]. As such, using the Linear pipeline provides us with an exact precover model to compare with the cruder model of this chapter.

Now, we embed each precover using Gaussian Embedding to obtain the prestego $\boldsymbol{\gamma}^{(l)}$ which should follow the following model:

$$\mathbf{y}_k^{(l)} \sim \mathcal{N}\left(\mathbf{H}\,\bar{\mathbf{x}}_k, \Sigma_k + \operatorname{diag}(\boldsymbol{\epsilon}_k)\right). \tag{9.4.6}$$

We finally compute the LRT for two different types of the Warden:

1. A **Lattice Warden** who uses the model of Eq (9.4.5) and Eq (9.4.6) as well as the lattice model as defined in Section 7.2. We refer the reader to Chapter 10 for the exact description of the LRT and how to compute the joint probabilities of the macro-blocks under the lattice model.

2. An **Ignorant Warden** which is defined exactly as in Section 9.2, in particular she does not use the covariance matrices Σ_k of the noise but only the variances $\boldsymbol{\sigma}^2$.

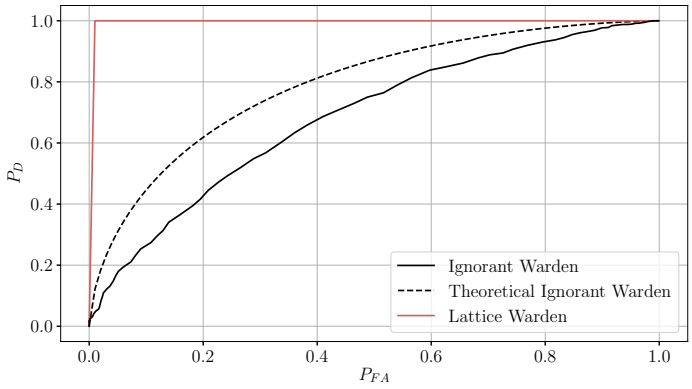


Figure 9.2: ROC curves of different LRTs on a dataset built using image 6551 in BossBase, to which independent Gaussian noise was added in the RAW domain and then developed using the Linear pipeline. Images were embedded using Gaussian Embedding at 0.1bpp. Note how the empirical LRT of the Lattice Warden has perfect detection despite the LRT of the ignorant Warden having a quite low performance.

A first observation is that there is a non-negligible discrepancy between the theoretical performance of the ignorant Warden's LRT and its actual empirical performance: we cannot provide any exact guarantee of security under this model as it is inexact. A second observation is that under the model of the Lattice Warden, the LRT always perfectly discriminates between precover and prestego images even though the LRT under the ignorant Warden's model is far from a perfect classification. This shows that taking into account correlations between DCT coefficients is extremely important in practice if we want to provide guarantees of performance. The final chapter of this manuscript will also demonstrate that, even for imperfect detectors such as neural networks, not using correlations leads to highly sub-optimal performance of the steganographic algorithm.

9.5 Conclusion

In this chapter we derived our first steganographic algorithm, Gaussian Embedding, based on the model of the sensor noise presented in the first part of this manuscript. We started in the simplest way possible by assuming no dependencies between DCT coefficients during the embedding phase. Note however that the variance estimation method we use does take into account these dependencies, leading to a superior estimation of the variances than the method presented in Chapter 8[1].

The steganographic algorithm then assumes an independent Gaussian model for both the precover and the prestego. The power of the optimal detector was then derived and we showed that in this case, the optimal prestego signal has a variance proportional to the variance of the precover noise. We then designed an embedding scheme which leverages this result while still being able to embed in the quantized domain, which is the domain of interest in our case.

We finished this chapter by validating our implementation by showing that the empirical performance are aligned with the theoretical predictions of Theorem 9.2.1 when using synthetic images following the noise model of this chapter. However, we also showed the limit of the simplified model of this chapter by showing that, when using natural images, the LRT that takes dependencies between DCT coefficients into account perfectly detects stego images generated by Gaussian Embedding.

The goal of the next chapter will be to fix this problem by designing an algorithm which fully leverages the dependencies between DCT coefficients.

[1] As was explained in Chapter 8 this is not necessarily a good thing since this "better" variance estimation method does not take into account the content of the image but only the noise. For the interested reader, we studied this question of the noise-content trade-off more thoroughly in the original publication [32].

APPENDICES

9.A LIMIT DISTRIBUTION OF THE LRT

In this appendix we derive the exact asymptotic performance of test derived in Section 9.2. Let :

$$p_{\sigma_i}(x) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(\frac{-x^2}{2\sigma_i^2}\right), \tag{9.A.1}$$

$$q_{\sigma_i,\epsilon_i}(x) = \frac{1}{\sqrt{2\pi(\sigma_i^2+\epsilon_i^2)}} \exp\left(\frac{-x^2}{2(\sigma_i^2+\epsilon_i^2)}\right), \tag{9.A.2}$$

$$\Lambda_i(\xi_i,\sigma_i,\epsilon_i) = \ln\left(\frac{p_{\sigma_i}(\xi_i)}{q_{\sigma_i,\epsilon_i}(\xi_i)}\right), \tag{9.A.3}$$

$$\Lambda(\boldsymbol{\xi},\sigma,\epsilon) = \sum_{i=0}^{n} \Lambda_i(\xi_i,\sigma_i,\epsilon_i), \tag{9.A.4}$$

Using Proposition 2.3.7, we write the expectation of the LRT under each hypothesis as the KL-divergence between the cover and stego distributions:

$$\begin{aligned}\mathbb{E}_{\mathcal{H}_0}[\Lambda_i] &= -D_{KL}\left(p_{\sigma_i} \,||\, q_{\sigma_i,\epsilon_i}\right) \\ &= \ln\left(\frac{\sqrt{\sigma_i^2+\epsilon_i^2}}{\sigma_i}\right) + \frac{\sigma_i^2}{2\left(\sigma_i^2+\epsilon_i^2\right)} - 0.5.\end{aligned} \tag{9.A.5}$$

and similarly for $\mathcal{H}_1$:

$$\mathbb{E}_{\mathcal{H}_1}[\Lambda_i] = D_{KL}\left(q_{\sigma_i,\epsilon_i} \,||\, p_{\sigma_i}\right) \tag{9.A.6}$$

The variances can also be computed analytically by identifying the moments of the Gaussian in the following integral:

$$\begin{aligned}Var_{\mathcal{H}_0}[\Lambda_i] = &\int_{-\infty}^{\infty} \ln^2\left(\frac{q_{\sigma_i,\epsilon_i}(z)}{p_{\sigma_i}(z)}\right) p_{\sigma_i}(z)\, dz \\ &- D_{KL}^2\left(p_{\sigma_i} \,||\, q_{\sigma_i,\epsilon_i}\right).\end{aligned} \tag{9.A.7}$$

Let:

$$\begin{aligned}c_1 &= \ln^2\left(\frac{\sigma_i}{\sigma_i^s}\right), \\ c_2 &= 2\ln\left(\frac{\sigma_i^s}{\sigma_i}\right)\frac{(\sigma_i^s)^2-\sigma_i^2}{2(\sigma_i^s)^2\sigma_i^2}, \\ c_3 &= \left(\frac{(\sigma_i^s)^2-\sigma_i^2}{2(\sigma_i^s)^2\sigma_i^2}\right)^2,\end{aligned}$$

with $(\sigma_i^s)^2 = \sigma_i^2 + \epsilon_i^2$.
We can rewrite the variance as:

$$\begin{aligned}Var_{\mathcal{H}_0}[\Lambda_i] = &\int_{-\infty}^{\infty} \left(c_3 z^4 + c_2 z^2 + c_1\right) p_{\sigma_i}(z)\, dz \\ &- D_{KL}^2\left(p_{\sigma_i} \,||\, q_{\sigma_i,\epsilon_i}\right).\end{aligned} \tag{9.A.8}$$

Finally, recognizing the second and fourth moment of the Gaussian distribution, we obtain:

$$Var_{\mathcal{H}_0}\left[\Lambda_i\right] = 3c_3\sigma_i^4 + c_2\sigma_i^2 + c_1 - D_{KL}^2\left(p_{\sigma_i} \,||\, q_{\sigma_i,\epsilon_i}\right), \tag{9.A.9}$$

and similarly under $\mathcal{H}_1$.

With some routine but tedious calculations, the second moments of the LRT can be simplified as:

$$\begin{cases} Var_{\mathcal{H}_0}\left[\Lambda_i\right] = \frac{\epsilon_i^4}{2\left(\epsilon_i^2+\sigma_i^2\right)^2} \\ Var_{\mathcal{H}_1}\left[\Lambda_i\right] = \frac{\epsilon_i^4}{2\sigma_i^4} \end{cases} \tag{9.A.10}$$

Finally, using the independence of the cover elements, the full moments are obtained as the sum of the individual moments:

	$\mathbb{E}\left[\Lambda\right]$	$Var\left[\Lambda\right]$
$\mathcal{H}_0$	$-\sum_{i=1}^n D_{KL}\left(p_{\sigma_i} \,||\, q_{\sigma_i,\epsilon_i}\right)$	$\sum_{i=1}^n \frac{\epsilon_i^4}{2\left(\epsilon_i^2+\sigma_i^2\right)^2}$
$\mathcal{H}_1$	$\sum_{i=1}^n D_{KL}\left(q_{\sigma_i,\epsilon_i} \,||\, p_{\sigma_i}\right)$	$\sum_{i=1}^n \frac{\epsilon_i^4}{2\sigma_i^4}$

As the number cover elements $n \to \infty$, Linderberg's central limit theorem implies that:

$$\Lambda(\mathbf{x}, \boldsymbol{\sigma}, \boldsymbol{\epsilon}) \rightsquigarrow \begin{cases} \mathcal{N}\left(\mathbb{E}_{\mathcal{H}_0}\left[\Lambda\right], \mathrm{Var}_{\mathcal{H}_0}\left[\Lambda\right]\right), & \text{under } \mathcal{H}_0 \\ \mathcal{N}\left(\mathbb{E}_{\mathcal{H}_1}\left[\Lambda\right], \mathrm{Var}_{\mathcal{H}_1}\left[\Lambda\right]\right), & \text{under } \mathcal{H}_1 \end{cases} \tag{9.A.11}$$

where $\rightsquigarrow$ denotes convergence in distribution.

9.B Power of the LRT

From the limiting distribution in Eq 9.A.11 and using Proposition 2.3.9, the asymptotic power of the LRT is given by:

$$P_D = \mathbb{P}\left(\delta\left(x\right) = \mathcal{H}_1 | \mathcal{H}_1\right) \tag{9.B.1}$$

$$= Q\left(\frac{Q^{-1}\left(P_{FA}\right)\sqrt{\mathrm{Var}_{\mathcal{H}_0}\left[\Lambda\right]} + \mathbb{E}_{\mathcal{H}_0}\left[\Lambda\right] - \mathbb{E}_{\mathcal{H}_1}\left[\Lambda\right]}{\sqrt{\mathrm{Var}_{\mathcal{H}_1}\left[\Lambda\right]}}\right), \tag{9.B.2}$$

where Q is the tail distribution function of the standard normal distribution.

Which we rewrite by plugging in the expression of each moment:

$$P_D = Q\left(\frac{Q^{-1}\left(P_{FA}\right)\sqrt{\sum_{i=1}^n \frac{\epsilon_i^4}{2\left(\epsilon_i^2+\sigma_i^2\right)^2}} - \sum_{i=1}^n \frac{\epsilon_i^4}{2\sigma_i^4+2\sigma_i^2\epsilon_i^2}}{\sqrt{\sum_{i=1}^n \frac{\epsilon_i^4}{2\sigma_i^4}}}\right), \tag{9.B.3}$$

Under the assumption that the power of the stego signal is negligible compared to the sensor noise, that is $\sigma_i^2 >> \epsilon_i^2$, we obtain the much more manageable expression:

$$
\begin{aligned}
P_D &\doteq Q\left(\frac{Q^{-1}\left(P_{FA}\right)\sqrt{\sum_{i=1}^{n}\frac{\epsilon_i^4}{2\sigma_i^4}}-\sum_{i=1}^{n}\frac{\epsilon_i^4}{2\sigma_i^4}}{\sqrt{\sum_{i=1}^{n}\frac{\epsilon_i^4}{2\sigma_i^4}}}\right), \\
&= Q\left(Q^{-1}\left(P_{FA}\right)-\sqrt{\sum_{i=1}^{n}\frac{\epsilon_i^4}{2\sigma_i^4}}\right).
\end{aligned}
\tag{9.B.4}
$$

10 Multivariate Gaussian steganography minimizing statistical detectability

This chapter is the culmination of the second part of this manuscript where we design the steganographic algorithm which fully leverages the model derived in Chapter 7 by taking all the dependencies between DCT coefficients into account.

10.1 Cover and stego model

10.1.1 *Precover model*

The precover model is exactly the model presented in Chapter 7, i.e. each macro-block $\mathbf{y}_k$ of the precover follows a multivariate Gaussian distribution with covariance Σ_k:

$$\mathbf{y}_k \sim \mathcal{N}\left(\boldsymbol{\mu}_k, \Sigma_k\right), \tag{10.1.1}$$

The dependency models presented in Section 7.2, namely the independent macro-block model and the lattice model will both be addressed in Section 10.3.

Before ending this subsection, we assume that we can write the whole cover $\mathbf{y}$ as multivariate Gaussian r.v:

$$\mathbf{y} \sim \mathcal{N}\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}\right), \tag{10.1.2}$$

with $\boldsymbol{\Sigma}$ being the covariance of the whole image.

This assumption is simple to prove if one remembers that photo-sites are all independant Gaussian random variables and that $\mathbf{H}$ is identical for each (macro)-block. One then can simply build a matrix $\mathbf{H}^*$ which takes the whole RAW image $\mathbf{x}$ as input and outputs the developed image $\mathbf{y}$. $\mathbf{H}^*$ is a block matrix built using copies of $\mathbf{H}$. $\mathbf{y}$ is then evidently also follows a multivariate Gaussian distribution.

10.1.2 *Prestego model*

We assume that the steganographer uses a centered multivariate Gaussian signal with covariance $\mathbf{E}_k$:

$$\mathbf{s}_k \sim \mathcal{N}\left(\mathbf{0}, \mathbf{E}_k\right). \tag{10.1.3}$$

The signal is then simply added to the k-th macro-block of the precover leading to the prestego macro-block $\boldsymbol{\gamma}_k$:

$$\begin{aligned} \boldsymbol{\gamma}_k &= \mathbf{y}_k + \mathbf{s}_k && (10.1.4)\\ &\sim \mathcal{N}\left(\boldsymbol{\mu}_k, \Sigma_k + \mathbf{E}_k\right) && (10.1.5) \end{aligned}$$

This choice of distribution is motivated by the same properties of the Gaussian distribution we used in Chapter 9; it also allows us to construct extremely efficient algorithms to compute the stego signal in the quantized domain – see Section 10.3.

10.2 Optimal detector

Under these precover and prestego models, the goal of the steganalyst is once again to construct a test in the continuous domain. We assume they have access to $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$, $\mathbf{E}$ as well as to the observations in the continuous domain $\boldsymbol{\xi}$. Furthermore we denote the covariance of the entire precover as $\boldsymbol{\Sigma}$ and the covariance of the entire prestego as $\mathbf{E}$. The steganalyst must choose between two hypotheses:

$$\begin{cases} \mathcal{H}_0 &= \boldsymbol{\xi} \sim \mathcal{N}\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}\right), \\ \mathcal{H}_1 &= \boldsymbol{\xi} \sim \mathcal{N}\left(\boldsymbol{\mu}, \boldsymbol{\Sigma} + \mathbf{E}\right). \end{cases} \tag{10.2.1}$$

This time, the optimal detector has a structure that prevents us to provide a simple asymptotic expression of its power. This is shown in the following theorem:

Theorem 10.2.1 (Multivariate Gaussian LRT)**.** *Under the settings of this section, the following holds:*

1. *The test maximizing the power function P_D for a prescribed false α_0 is the likelihood ratio test (LRT):*

$$\Lambda(\boldsymbol{\xi}, \boldsymbol{\Sigma}, \mathbf{E}) = \frac{q_{\boldsymbol{\Sigma},\mathbf{E}}\left(\boldsymbol{\xi}\right)}{p_{\boldsymbol{\Sigma}}\left(\boldsymbol{\xi}\right)} \underset{\mathcal{H}_1}{\overset{\mathcal{H}_0}{\lessgtr}} \tau, \tag{10.2.2}$$

   where τ is a threshold fixed so that $\mathbb{P}\left[\Lambda > \tau \mid \mathcal{H}_0\right] = \alpha_0$, where α_0 is the chosen probability of false alarm.

2. *Let $\mathbf{A} = \boldsymbol{\Sigma}^{-1} - \left(\boldsymbol{\Sigma} + \mathbf{E}\right)^{-1}$. An equivalent test can be written as:*

$$\Lambda^*(\boldsymbol{\xi}, \boldsymbol{\Sigma}, \mathbf{E}) = \boldsymbol{\xi}^T \mathbf{A}\, \boldsymbol{\xi} \underset{\mathcal{H}_1}{\overset{\mathcal{H}_0}{\lessgtr}} \tau', \tag{10.2.3}$$

   where $\tau' = 2\tau - 2\log\left(\frac{|\boldsymbol{\Sigma}|}{|\mathbf{E}|}\right)$.

3. *The likelihood ratio $\Lambda^*(\boldsymbol{\xi}, \boldsymbol{\Sigma}, \mathbf{E})$ is a sum of weighted chi-square random variables:*

$$\boldsymbol{\xi}^T \mathbf{A}\, \boldsymbol{\xi} = \sum_{i=0}^{M} \mathbf{K}_{ii} \xi_{u,i}^2. \tag{10.2.4}$$

   where $\xi_{u,i}^2 \sim \mathcal{X}\left(1\right)$ and where the $\mathbf{K}_{ii}$ are the eigenvalues of $\mathbf{A}\,\boldsymbol{\Sigma}$ or $\mathbf{A}\left(\boldsymbol{\Sigma} + \mathbf{E}\right)$ under $\mathcal{H}_0$ and $\mathcal{H}_1$ respectively.

Proof. (1) follows directly from the Neyman-Pearson Lemma as stated in Theorem 2.3.5.
(2) follows from Proposition 2.3.6 and some simple algebraic manipulations – see [80, Chapter 3, Section 3].
(3) is proven in Appendix 10.A. □

Sadly, there exists no general closed-form expression for the tails of a weighted sum of chi-square random variables [58]. Furthermore, we cannot use the central-limit theorem to approximate this distribution since some weights might be far larger than most other weights. Due

to this fact, it is difficult to design a steganographic algorithm with optimality guarantees in this setting. However, thanks to the Proposition 2.3.10, we can still *upper bound* the power of the LRT through the expression of the KL-divergence between the cover and stego distributions. This is the method we propose to follow in the next section to design a steganographic algorithm with security guarantees.

10.3 Embedding

For both dependency models, the goal of the steganographer is to minimize the power of the LRT under a given payload constraint R. However as we have seen in Section 10.2, no closed-form expression of P_D is available. Therefore, we simplify the problem by instead minimizing an upper bound on the power of the LRT. That is, we abandon claims of optimality in exchange for a tractable problem while, at the same time, keeping guarantees of security for a given steganographic signal. Moreover, we provide the best security guarantee available under the chosen bound.

As was shown with Proposition 2.3.10, the power of the LRT can be controlled by the following quantity:

$$\mathrm{D}_{KL}\left(p_{\mathbf{\Sigma}} \,||\, q_{\mathbf{\Sigma},\mathbf{E}}\right) \tag{10.3.1}$$

Consequently we can formulate the following optimization problem:

$$\begin{cases} \min\limits_{\mathbf{E}} & \mathrm{D}_{KL}\left(p_{\mathbf{\Sigma}} \,||\, q_{\mathbf{\Sigma},\mathbf{E}}\right) \\ R = & \sum\limits_{i=1}^{n} \sum\limits_{k\in\mathbb{Z}} \beta_i^k \log \beta_i^k \end{cases} \tag{10.3.2}$$

To solve this problem we use the following theorem:

Theorem 10.3.1 (Optimal covariance of the prestego)**.** *Let $R^* \in \mathbb{R}^+$. Let also the precover $\mathbf{y}$ and prestego $\boldsymbol{\gamma}$, both of size n, be such that:*

$$\mathbf{y} \sim \mathcal{N}\left(\mathbf{0}, \Sigma\right), \tag{10.3.3}$$

$$\boldsymbol{\gamma} \sim \mathcal{N}\left(\mathbf{0}, \Sigma + \mathbf{E}\right). \tag{10.3.4}$$

Then the optimization problem the system in Eq (11.5.22) has for solution a positive definite matrix $\mathbf{E}^*$ such that:

$$\mathbf{E}^* = \alpha\Sigma, \tag{10.3.5}$$

with $\alpha \in \mathbb{R}^+$.

Proof. See Appendix 10.B. □

In other words, the prestego signal has a covariance proportional to the covariance of the precover noise. In particular, α is the same for all macro-blocks of the precover.

Consequently, the system in Eq (11.5.22) is easily solved by a simple bisection search on α since $\sum_{i=0}^{n} \sum_{j\in\mathbb{Z}} \beta_i^{(j)} \log\left(\beta_i^{(j)}\right)$ is increasing in α.

However, to be able to compute the value of the payload size for a given α, it is necessary to compute all individual $\beta_i^{(j)}$ which is not an obvious task since we want to compute an individual value in the quantized domain from a multivariate signal in the continuous domain. The next subsection addresses this difficulty and presents a solution that avoids relying on expensive Monte-Carlo simulations to do so.

10.3.1 *Computing embedding probabilities*

First, let $\mathbf{s}_k$ be a prestego signal macro-block. It follows a centered multivariate Gaussian (MVG) random variable (rv) of N elements with a full-rank covariance $\mathbf{E}_k$. Denote the i-th element of $\mathbf{s}_k$ as $s_{k,i}$. Then we have, for all i:

$$s_{k,i}|s_{k,1}, s_{k,2}, ...s_{k,i-1} \sim \mathcal{N}\left(\bar{\eta}_{k,i}, \bar{\epsilon}_{k,i}^2\right). \tag{10.3.6}$$

In other words, every element of an MVG rv conditioned on all its previous elements follows a univariate Gaussian distribution with mean $\bar{\eta}_{k,i}$ and variance $\bar{\epsilon}_{k,i}^2$.

Secondly, since $\mathbf{E}_k$ is a full-rank covariance matrix, it is symmetric positive definite. Consequently it can be factorized uniquely by a lower triangular matrix $\mathbf{L}_k$ with positive diagonal entries [43, Corollary 7.2.9]:

$$\mathbf{E}_k = \mathbf{L}_k \mathbf{L}_k^T, \tag{10.3.7}$$

which corresponds to the Cholesky decomposition of the covariance matrix.

Finally, let $\mathbf{w}_k$ be a vector of M identical standard Gaussian rvs with zero mean and unit variance. We can correlate white noise by multiplying it by the Cholesky decomposition of a chosen covariance matrix:

$$\mathbf{L}_k \mathbf{w}_k \sim \mathcal{N}\left(\mathbf{0}, \mathbf{E}_k\right). \tag{10.3.8}$$

The parameters $\bar{\boldsymbol{\eta}}$ and $\bar{\boldsymbol{\epsilon}}$ can be computed efficiently using the Cholesky decomposition and the realization of $\mathbf{s}$ using the following equations – see Figure 10.1 for a graphical explanation:

$$\bar{\boldsymbol{\epsilon}}_k = \operatorname{diag}(\mathbf{L}_k), \tag{10.3.9}$$

$$\bar{\boldsymbol{\eta}}_k = \left(\mathbf{L}_k \mathbf{w}_k - \operatorname{diag}(\mathbf{L}_k)\mathbf{w}_k\right). \tag{10.3.10}$$

$$\begin{pmatrix} l_{11} & & \\ l_{21} & l_{22} & \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} l_{11}x_1 \\ l_{21}x_1 + l_{22}x_2 \\ l_{31}x_1 + l_{32}x_2 + l_{33}x_3 \end{pmatrix} \quad \begin{matrix} \bar{\epsilon}_i = l_{ii} \\ \\ \bar{\eta}_i = \sum_{j=1}^{i-1} l_{ij}x_j \end{matrix}$$

Figure 10.1: Illustration of how to compute the conditioned Gaussian parameters $\bar{\epsilon}_i$ (in red) and $\bar{\eta}_i$ (in cyan) using the Cholesky decomposition $\mathbf{L}$ of the covariance matrix. For the i-th line, the x_i on the diagonal is the random variable while the other $x_j, j < i$ are the realizations of the previous elements. One can see that the standard deviation $\bar{\epsilon}_i$ is simply l_{ii} whereas the mean $\bar{\eta}_i$ is the sum of the i-th line minus the last element.

Now observe that all of this methodology can be applied to every macro-block of the prestego signal $\mathbf{s}_k$ as defined in Eq (10.1.2). If we apply Eq (10.3.9)-(10.3.10) to every $\mathbf{s}_k$, we obtain a vector $\bar{\mathbf{s}}$ of n elements such that:

$$\bar{s}_i \sim \mathcal{N}(\bar{\eta}_i, \bar{\epsilon}_i)\,. \tag{10.3.11}$$

Finally, using the chain rule of probability on each $\beta_i^{(j)}$, and the quantization of the Gaussian distribution, the embedding probabilities $\beta_i^{(j)}$ are obtained by:

$$\beta_i^{(j)} = \mathbf{\Phi}\left(\frac{j - r_i - \bar{\eta}_i + 0.5}{\bar{\epsilon}_i}\right) - \mathbf{\Phi}\left(\frac{j - r_i - \bar{\eta}_i - 0.5}{\bar{\epsilon}_i}\right), \tag{10.3.12}$$

where $r_i = y_i - [y_i]$ denotes the rounding error of i-th DCT coefficient.

In practice, we perform a q-ary embedding, that is, the alphabet size of the embedding scheme is finite: j is thus constrained to a finite range between $-q$ and q and the $\beta_i^{(j)}$ must be normalized accordingly:

In practice, the impact of the normalization is negligible.

$$\beta_{i,\text{normalized}}^{(j)} = \frac{\beta_i^{(j)}}{\sum_{j=-q}^{q} \beta_i^{(j)}}. \tag{10.3.13}$$

To understand Equation (10.3.12), observe that after adding the prestego signal, the new value will either stay in the same integer bin after rounding or fall into a neighboring one. The probability of falling into one bin or another depends on the original rounding error r_i, the conditioned mean $\bar{\eta}_i$ and variance $\bar{\epsilon}_i^2$ of the prestego signal – see Figure 10.2. Computing $\beta_i^{(j)}$ is then simply a matter of computing the probability of falling into each bin which is simply the area under the curve of the prestego signal centered at the original rounding error inside each integer bin.

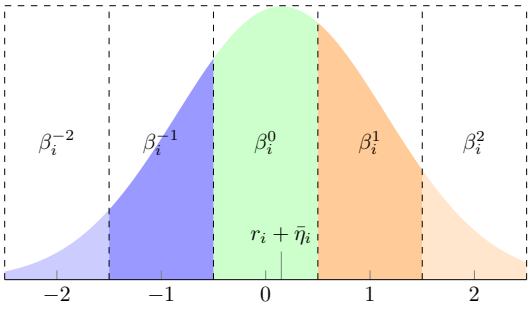


Figure 10.2: Figure explaining how $\beta_i^{(j)}$ are computed with a prestego signal centered at the original rounding error r_i. The dashed boxes represent the integer bins after rounding the DCT coefficient whereas the colored parts represent the area under the curve of the prestego signal which will be taken into account for each $\beta_i^{(j)}$.

10.3.2 *Independent macro-block model*

The first dependency model assumes that every elements in the same $\sqrt{N} \times \sqrt{N}$ macro-block are possibly dependent while elements in two different macro-blocks are considered independent. As shown in Figure 7.2, only non-overlapping macro-blocks are considered.

To build the macro-blocks in this model, one just has to split the whole image $\mathbf{y}$ into non-overlapping macro-blocks $\mathbf{y}_k$ of the same size so that every element y_i of the precover belongs to one and only one macro-block.

Consequently we have each $\mathbf{y}_k$ independent from every other $\mathbf{y}_k$ and following a MVG distribution:

$$\mathbf{y}_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \Sigma_k)\,. \tag{10.3.14}$$

To simulate the stego signal the steganographer first samples the prestego signal according to the precover distribution and secondly, scales it so that it matches the payload constraint. The signal is finally added to the precover and quantized.

We provide the full algorithm of the steganographic scheme that uses this dependency model, henceforth named Multivariate Gaussian Embedding, or MGE for short, in Algorithm 4.

Algorithm 4: Multivariate Gaussian Embedding

Data: N: Number of element in a macro-block, $\mathbf{y}$: Precover, $\mathbf{\Sigma}$: Covariance matrices, R: Payload size (in nats)
Result: $\mathbf{s}$: Prestego signal solving Eq (11.5.22)
$n \leftarrow \text{size}(y)$;
$m \leftarrow n/N$;
$\mathbf{r} \leftarrow \mathbf{y} - \text{round}(\mathbf{y})$;
for *k in 1..m* **do**
  `// Sample` N `standard gaussians`
  $\mathbf{w}_k \leftarrow \mathcal{N}(0, \mathbf{I})$;
  `// Correlate the noise using the Cholesky decomposition`
  $\mathbf{L}_k \leftarrow \text{Cholesky}(\Sigma_k)$;
  $\mathbf{s}_k \leftarrow \mathbf{L}_k \mathbf{w}_k$;
`// Interval bisection on` α `until the payload constraint is met`
$l, r \leftarrow$ InitializeIntervalBounds();
`// Compute entropy using Eq (11.5.22) and Eq (10.3.12). the argument of the function is` α`.`
$fl \leftarrow \text{ComputeEntropy}\,(l) - R$;
$fr \leftarrow \text{ComputeEntropy}\,(r) - R$;
while error $\neq 0$ **do**
  $\alpha \leftarrow \frac{l+r}{2}$;
  error $\leftarrow \text{ComputeEntropy}\,(\alpha) - R$;
  if fl $\cdot$ error < 0 **then**
    $r \leftarrow \alpha$;
    $fr \leftarrow$ error;
  else
    $l \leftarrow \alpha$;
    $fl \leftarrow$ error;
for *k in 1..m* **do**
  $\mathbf{s}_k \leftarrow \alpha \mathbf{s}_k$;

10.3.3 *Lattice model*

In the lattice model, we assume dependencies between DCT coefficients within the same block as well as among DCT coefficients with neighboring blocks. Recall that under the linear pipeline, such a model theoretically allows sampling the stego signal exactly, as was shown

in [75].

First, we introduce some notations that will be used throughout this subsection. We write $\mathbf{\Lambda}$ to denote a set of blocks, which we call a lattice, such that all blocks in the set are independent from one another.

For a given block $\mathbf{y}_k$ we write $\mathbf{y}_k^{\text{cardinal}}$ with cardinal $\in \{N, S, E, W, NE, NW, SE, SW\}$ to designate the block which is respectively above, below, right of, left of, etc... of $\mathbf{y}_k$.

Now, from Figure 7.2, observe that the lattice model assumptions lead to a natural decomposition of the image into four lattices $\mathbf{\Lambda}_1, \mathbf{\Lambda}_2, \mathbf{\Lambda}_3, \mathbf{\Lambda}_4$.

The steganographer's goal is, first, to sample a MVG signal $\mathbf{s}$ with the same covariance as the noise of the precover. To do so, we use the fact that the pdf of the whole image can be decomposed using the chain rule of probability so that the pdf of $\mathbf{s}$ is:

$$\begin{aligned} p(\mathbf{s}) =& p\left(\mathbf{s}^{\mathbf{\Lambda}_1}\right) p\left(\mathbf{s}^{\mathbf{\Lambda}_2}|\mathbf{s}^{\mathbf{\Lambda}_1}\right) p\left(\mathbf{s}^{\mathbf{\Lambda}_3}|\mathbf{s}^{\mathbf{\Lambda}_1}, \mathbf{s}^{\mathbf{\Lambda}_2}\right) \\ & p\left(\mathbf{s}^{\mathbf{\Lambda}_4}|\mathbf{s}^{\mathbf{\Lambda}_1}, \mathbf{s}^{\mathbf{\Lambda}_2}, \mathbf{s}^{\mathbf{\Lambda}_3}\right). \end{aligned} \tag{10.3.15}$$

Consequently, we can first sample the prestego signal in $\mathbf{\Lambda}_1$ by sampling from $p\left(\mathbf{s}^{\mathbf{\Lambda}_1}\right)$, then sample the prestego signal in $\mathbf{\Lambda}_2$ according to $p\left(\mathbf{s}^{\mathbf{\Lambda}_2}|\mathbf{s}^{\mathbf{\Lambda}_1}\right)$ and so on for the two other lattices.

Now let $\mathbf{\Lambda}_1^*, \mathbf{\Lambda}_2^*, \mathbf{\Lambda}_3^*, \mathbf{\Lambda}_4^*$ be such that:

$$\mathbf{\Lambda}_1^* = \mathbf{\Lambda}_1, \tag{10.3.16}$$

$$\mathbf{\Lambda}_2^* = \{[y^{NE}, y^{NW}, y^{SE}, y^{SW}, y] | y \in \mathbf{\Lambda}_2\}, \tag{10.3.17}$$

$$\mathbf{\Lambda}_3^* = \{[y^N, y^S, y^E, y^W, y] | y \in \mathbf{\Lambda}_3\}, \tag{10.3.18}$$

$$\mathbf{\Lambda}_4^* = \{[y^N, y^S, y^E, y^W, y^{NE}, y^{NW}, y^{SE}, y^{SW}, y | y \in \mathbf{\Lambda}_4\}. \tag{10.3.19}$$

In other words, each $\mathbf{\Lambda}^*$ contains vectors built from blocks in $\mathbf{\Lambda}$ along with the blocks from previous lattices on which they depend. See Figure 10.3 for a graphical representation of the dependency structure of each lattice used to construct each $\mathbf{\Lambda}^*$. In what follows, we refer to the covariance matrix of the k-th vector of the i-th lattice $\mathbf{\Lambda}_i^*$ as $\Sigma_k^{\mathbf{\Lambda}_i^*}$

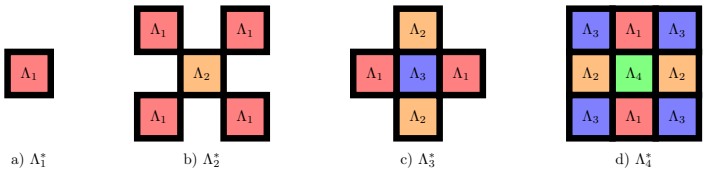


Figure 10.3: Graphical representation of the blocks used to construct each $\mathbf{\Lambda}_i^*$.

By using exactly the same reasoning as in Section 10.3.1, adapted to a multivariate setting, we can easily sample the blocks in each lattice by using the Cholesky decomposition of the covariance of each vector in each $\mathbf{\Lambda}_i^*$:

$$\mathbf{s}_k^{\mathbf{\Lambda}_i} = \mathbf{L}_k^{\mathbf{\Lambda}_i^*} \mathbf{w}_k^{\mathbf{\Lambda}_i}, \forall i \in \{1, 2, 3, 4\} \tag{10.3.20}$$

with $\mathbf{L}_k^{\mathbf{\Lambda}_i^*}$ being the rectangular submatrix of the Cholesky decomposition of $\mathbf{\Sigma}_k^{\mathbf{\Lambda}_i^*}$ referring to the central block of the k-th vector of $\mathbf{\Lambda}_i^*$.

In practice, using the convention of Eq (10.3.16-10.3.19), where the central block is always at the end of each vector of $\mathbf{\Lambda}_i^*$, $\mathbf{L}_k^{\mathbf{\Lambda}_i^*}$ would be constructed by taking the 64 last lines of the Cholesky decomposition of $\mathbf{\Sigma}_k^{\mathbf{\Lambda}_i^*}$.

Once every $\mathbf{s}_k^{\mathbf{\Lambda}_i}$ are computed, the algorithm to sample the prestego signal and perform the embedding is exactly the same as for the independent macro-block model.

10.3.4 *Embedding in practice*

The algorithms we have presented so far are useful for *simulating* extremely efficiently the stego signal in the quantized domain. However, in practice, when using Syndrom-Treillis Codes to embed a given message , we need to compute the embedding probabilities $\beta_i^{(j)}$ and convert them to the corresponding costs $\rho_i^{(j)}$ using:

$$\beta_i^{(j)} = \frac{e^{-\lambda\rho_i^{(j)}}}{1 + \sum_{j\neq 0} e^{-\lambda\rho_i^{(j)}}}. \tag{10.3.21}$$

where λ is the Lagrange multiplier solving the system in Eq. (11.5.22). However, the probability of embedding in the l-th coefficient of a given block depends on the actual embedding performed by the STC on all the previous coefficients in that block. The practical implementation will thus have to be performed iteratively.

In the first iteration, we compute the embedding probabilities of the first coefficient of each block. Here, we necessarily have $\bar{\nu}_i = 0$ for all coefficients; $\beta_i^{(j)}$'s can thus directly be computed and converted to costs. Once the payload has been embedded, the uncorrelated prestego signal $\mathbf{w}_i$ corresponding to each of these coefficients must be computed since it will be used to compute the $\beta_i^{(j)}$ of the next coefficient. To do so we can sample the correlated prestego signal $\mathbf{s}_i$ using rejection sampling until the rounded value of $\mathbf{s}_i$ matches the actual embedding in the quantized domain. We then compute $\mathbf{w}_i$ using the fact that:

$$\mathbf{w}_k = \mathbf{L}_k^{-1}\mathbf{s}_k \tag{10.3.22}$$

The l-th iteration is done in exactly the same manner for the l-th coefficient of each block, except this time we compute $\bar{\nu}_l$ using the previously computed $\mathbf{w}_i$ and Eq (10.3.10).

The only caveat with this approach is that α must be fixed before the embedding and be identical for each lattice. This can be done by simulating embedding on the image until an α big enough for the message to be embedded is found, then use this α on the actual STC embedding. However, since the actual entropy depends on the $\beta_i^{(j)}$, hence on the actual embedding performed, it is theoretically not possible to compute the minimal α that would allow reaching the payload size. In practice, this is not such a problem as we observed during simulations that the realizations of the $\mathbf{w}_i$ play a very small role on the entropy; it will usually make it change by 1 or 2 nats at most if at all. A good rule of thumb would then be to find the optimal α for a

slightly higher payload and to use it to ensure the message will fit on the first try.

10.4 Validation of the method

In the previous chapter, we showed that only using the diagonal of the covariance matrix led to a great loss of performance against a detector that took into account correlations between DCT coefficients. In this section, we show that Multivariate Gaussian Embedding does indeed solve this problem and has the performance predicted by Theorem 10.3.1. We follow the same methodology as the previous chapter: first, we test the performance of the algorithm on synthetic images and then go on to test the method on natural images developed with the linear pipeline.

10.4.1 *Synthetic images*

We build a dataset of synthetic images with the parameters of the noise taken from the camera that was used to capture the image. We selected a RAW image $\bar{\mathbf{x}}$ from BOSSBase which we then develop using the Linear pipeline, as defined in Section 7.6, to obtain the baseline image $\bar{\mathbf{y}}$. We then generated 1000 synthetic precover images $\mathbf{y}^{(l)}$ by sampling them directly such that for all l:

$$\mathbf{y}^{(l)} \sim \mathcal{N}\left(\boldsymbol{\mu}_k, \Sigma_k\right). \tag{10.4.1}$$

where $\boldsymbol{\mu}_k = \bar{\mathbf{y}}_k$ and Σ_k is the covariance of $\bar{\mathbf{y}}_k$ estimated using the method presented in Section 7.3. We also sample this signal so that macro-blocks follow the Lattice model defined in Section 7.2 by using the sampling method presented in Section 10.3.3

We then embed each image using the Multivariate Gaussian Embedding algorithm for different payloads to obtain the prestego images $\boldsymbol{\gamma}^{(l)}$:

$$\boldsymbol{\gamma}^{(l)} \sim \mathcal{N}\left(\boldsymbol{\mu}_k, \Sigma_k + \mathbf{E}_k\right). \tag{10.4.2}$$

We finally compute the LRT under the model presented in this chapter:

$$\Lambda(\boldsymbol{\xi}, \boldsymbol{\Sigma}, \mathbf{E}) = \frac{q_{\boldsymbol{\Sigma},\mathbf{E}}\left(\boldsymbol{\xi}\right)}{p_{\boldsymbol{\Sigma}}\left(\boldsymbol{\xi}\right)}. \tag{10.4.3}$$

We present the ROC curve of the empirical LRT against the ROC curve of the theoretical LRT in Figure 10.4.

Once again, the performance of the empirical LRT matches the expected performance for all the tested cameras, validating our theoretical analysis of the algorithm.

10.4.2 *Natural images under the Linear pipeline*

We repeat the experiment of Section 9.4.2[1] but we now embed each precover by using the Multivariate Gaussian Embedding algorithm to obtain the prestego $\boldsymbol{\gamma}^{(l)}$ which, if our theoretical model is correct, should follow:

[1] That is, the noise is now sampled in the RAW domain and is processed in the same way as the image instead of being sampled directly in the developed domain.

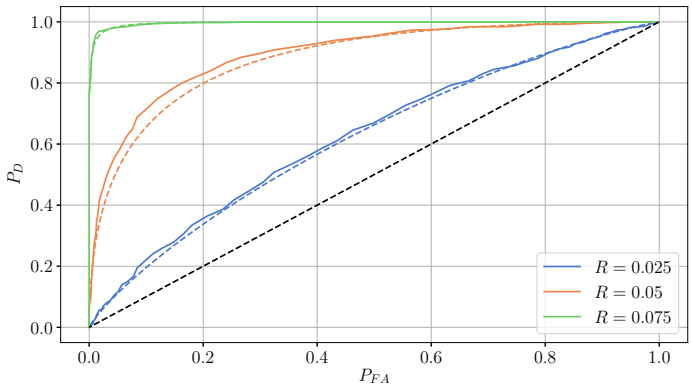


Figure 10.4: LRT on a dataset with synthetic multivariate Gaussian noise built with image 6551 in BossBase for different relative payloads R. The dotted lines represent the theoretical performance of the LRT whereas the full line represent the empirical performance.

$$\mathbf{y}_k^{(l)} \sim \mathcal{N}\left(\mathbf{H}\,\bar{\mathbf{x}}_k, \Sigma_k + \mathbf{E}_k\right). \tag{10.4.4}$$

Notice that this is the same model as for the synthetic case: if the empirical performance matches the theoretical one, this is strong evidence that our model of the noise is exact for the Linear pipeline.

The results are presented in Figure 10.5.

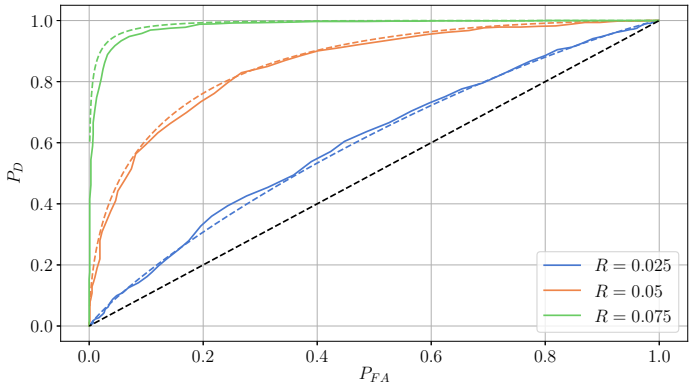
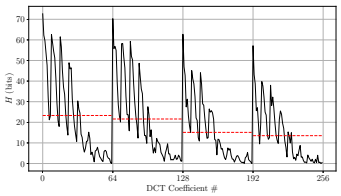


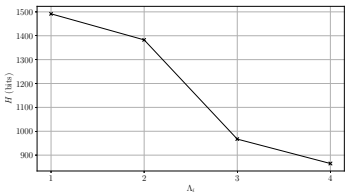
Figure 10.5: LRT on a dataset to which independent Gaussian noise was added in the RAW domain and then developed using the Linear pipeline with image 6551 in BossBase for different relative payloads R. The dotted lines represent the theoretical performance of the LRT whereas the full line represent the empirical performance.

Notice how Figure 10.5 is virtually indistinguishable from Figure 10.4 even though the former is, for all intent and purpose, a natural image. Note that the difference in the theoretical performance between the two figures is only due to the fact that the choice α leading to the right payload in the quantized domain is different for the synthetic and the natural images.

This strongly validates our noise model as a correct model of the noise of such images, at least for the Linear pipeline as per the assumption of our model.



(a) Payload allocation per DCT mode



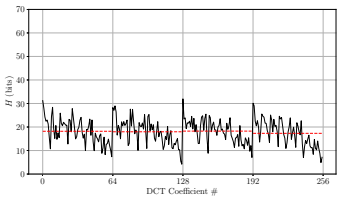
(b) Payload allocation per lattice

Figure 10.6: BossBase image 3492 developed with the Linear pipeline. On the left, how much entropy is allocated per mode for each macro-lattices, noting that each macro-lattice contains 64 modes. The red dotted lines represented the average entropy allocated per macro-lattice. On the right, how much entropy is allocated per macro-lattice.

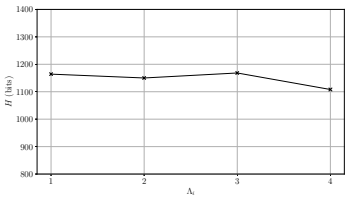
10.4.3 *Information hidden per lattice*

When using Multivariate Gaussian Embedding, cover images are effectively decomposed into 4×64 lattices when using the 4 Lattice model. In this subsection, we investigate how the payload is allocated across each lattice. Two main parameters play a role here: the value of the variance of the DCT coefficients and the act of conditioning DCT coefficients on realizations of previous lattices. On one hand the higher the variance, the higher the allocation of the payload will be. On the other hand, conditioning a DCT coefficient on previous lattices will tend to decrease the payload allocated to it (due to the loss of entropy), depending on the strength of the correlations with DCT coefficients in these lattices.

To verify these heuristics, we embed some images developed with either the Linear or the Bosslike pipeline with 4 Lattice MGE at different payloads and compute the entropy on each lattice in the quantized domain – see the results in Figures 10.6 to 10.9.

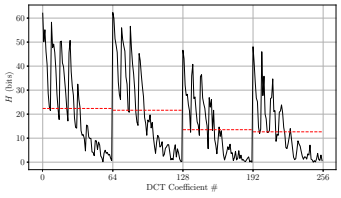


(a) Payload allocation per DCT mode

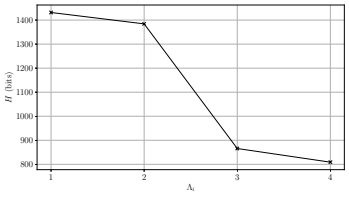


(b) Payload allocation per lattice

Figure 10.7: BossBase image 3492 developed with the Bosslike pipeline.



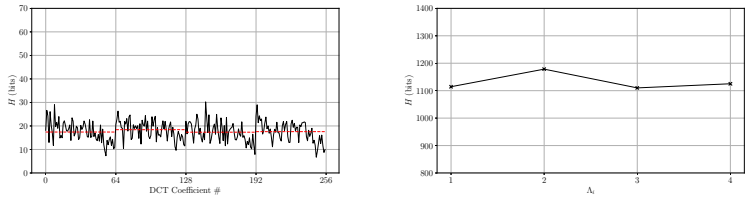
(a) Payload allocation per DCT mode



(b) Payload allocation per lattice

Figure 10.8: BossBase image 3799 developed with the Linear pipeline.

In the case of the Linear pipeline, we can observe the two parameters in action: for a given block of DCT coefficient – which we might call a "macro-lattice" here, the payload allocated decreases steadily for DCT coefficients in the higher frequency modes. This is both due to the fact these coefficients variances decrease for higher modes due to



(a) Payload allocation per DCT mode (b) Payload allocation per lattice

Figure 10.9: BossBase image 3799 developed with the Bosslike pipeline.

the DCT transform as well as for the fact that each DCT coefficient is conditioned on every coefficient of a lower mode.

If we look at the entropy per macro-lattice, we indeed observe that the allocated payload decreases consistently as the index of the lattice increases. More precisely, we see a small loss for Λ_2 since the central block is only conditioned on diagonal neighboring blocks – see Figure 10.3 – where the correlations are small. The loss is far stronger for Λ_3 and Λ_4 where the central block are now dependent with horizontal and vertical neighboring blocks with far stronger correlations.

For the case of the Bosslike pipeline, the observations for a given block are the same but we can also see that the entropy per macro-lattices is more or less constant between each Λ_i. This comes from the fact that, because of the resizing operations in this pipeline, inter-block dependencies are actually negligible.

This observation will also explain the results in Chapter 11 where we will see that the independent macro-block model has security performances equivalent to the 4 lattice model for the Bosslike pipeline.

10.5 Conclusion

In this chapter, we derived a steganographic algorithm, Multivariate Gaussian Embedding, which is able to leverage the full model that was presented in the first part of this manuscript. To get there, we proved a key result which states that the optimal covariance of the prestego signal is proportional to the covariance of precover noise.

We then provided an algorithm that allows simulating this optimal signal extremely efficiently by leveraging the Cholesky decomposition of covariance matrices. Two variants of the algorithm were presented depending on the chosen dependency model.

We finished the chapter by validating the implementation of our steganographic algorithm by showing that its empirical performance match the theoretical performance given by Theorem 10.2.1. We also experimentally showed that our noise model is exact under the Linear pipeline when using the four lattices dependency model.

In the next and last chapter of this manuscript we evaluate the performance of the Gaussian Embedding algorithms presented in the second part of this manuscript as well as further experimental analysis.

Appendices

10.A Likelihood ratio is a weighted sum of chi-square random variable

To be as general as possible, we will not consider any particular dependency model in this appendix. As such we will consider that the sample $\boldsymbol{\xi}$ is the whole image under scrutiny. Consequently we try to derive the optimal test which discriminate between the two following hypotheses:

$$\begin{cases} \mathcal{H}_0 &= \{\boldsymbol{\xi} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)\}, \\ \mathcal{H}_1 &= \{\boldsymbol{\xi} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma + \mathbf{E})\}. \end{cases} \tag{10.A.1}$$

Notice that we consider here the covariance of the whole image Σ and not the covariance of the blocks. Now recall that the optimal test under our setting is the likelihood-ratio test given by:

$$\Lambda(\boldsymbol{\xi}, \boldsymbol{\Sigma}, \mathbf{E}) = \frac{q_{\boldsymbol{\Sigma},\mathbf{E}}(\boldsymbol{\xi})}{p_{\boldsymbol{\Sigma}}(\boldsymbol{\xi})} \underset{\mathcal{H}_1}{\overset{\mathcal{H}_0}{\lessgtr}} \tau, \tag{10.A.2}$$

To compute the power of this test, we need to compute the distribution of the likelihood ratio under both hypotheses. To do so, we use the fact that the statistic of the LRT can be written as a quadratic form – see [80][Chapter 3, Section 3] for a derivation:

$$\frac{1}{2}\left(\boldsymbol{\xi}^T\left(\boldsymbol{\Sigma}^{-1} - (\boldsymbol{\Sigma}+\mathbf{E})^{-1}\right)\boldsymbol{\xi} + \log\left(\frac{|\Sigma|}{|\boldsymbol{\Sigma}+\mathbf{E}|}\right)\right) \underset{\mathcal{H}_1}{\overset{\mathcal{H}_0}{\lessgtr}} \tau. \tag{10.A.3}$$

This test can be simplified as:

$$\hat{\Lambda}(\boldsymbol{\xi}, \boldsymbol{\Sigma}, \mathbf{E}) = \boldsymbol{\xi}^T\left(\boldsymbol{\Sigma}^{-1} - (\boldsymbol{\Sigma}+\mathbf{E})^{-1}\right)\boldsymbol{\xi} \underset{\mathcal{H}_1}{\overset{\mathcal{H}_0}{\lessgtr}} \tau'. \tag{10.A.4}$$

by putting the contribution of the constant values in the threshold.

We will show that this test can be rewritten as a sum of weighted independent standard chi-squared rv.s. We will develop only the case under $\mathcal{H}_0$; the case for $\mathcal{H}_1$ follows exactly from the same method.

Let:

$$\mathbf{A} = \boldsymbol{\Sigma}^{-1} - (\boldsymbol{\Sigma}+\mathbf{E})^{-1}, \tag{10.A.5}$$

and decompose Σ using the symmetric square-root matrix :

$$\Sigma = \Sigma^{1/2}\Sigma^{1/2}. \tag{10.A.6}$$

Using the spectral theorem we can write:

$$\Sigma^{1/2}\mathbf{A}\Sigma^{1/2} = \mathbf{U}\mathbf{K}\mathbf{U}^T, \tag{10.A.7}$$

where $\mathbf{U}$ is an orthogonal matrix and $\mathbf{K}$ a diagonal matrix.

Finally, let $\boldsymbol{\xi}_u = \mathbf{U}^T\Sigma^{-1/2}\boldsymbol{\xi}$ and note that it follows a centered multivariate Gaussian distribution with covariance the identity matrix (by the orthogonality of $\mathbf{U}$).

Now write:

$$\boldsymbol{\xi}^T \mathbf{A}\, \boldsymbol{\xi} = \boldsymbol{\xi}^T\, \Sigma^{-1/2} \mathbf{U}\mathbf{K}\mathbf{U}^T \Sigma^{-1/2}\, \boldsymbol{\xi} \tag{10.A.8}$$

$$= \boldsymbol{\xi}_u^T\, \mathbf{K}\, \boldsymbol{\xi}_u = \sum_{i=0}^{M} \mathbf{K}_{ii} \xi_{u,i}^2. \tag{10.A.9}$$

The statistic in Eq (10.A.4) is thus a realization of a sum of **independent** standard chi-squared random variable weighted by the eigenvalues of $\mathbf{A}\Sigma$ and $\mathbf{A}\,(\boldsymbol{\Sigma} + \mathbf{E})$ under $\mathcal{H}_0$ and $\mathcal{H}_1$ respectively.

10.B Scaling of noise covariance minimizes KL-divergence

Computing the power of the LRT when the distribution under both hypotheses are weighted sum of chi-square distribution is difficult in the general case. Furthermore, we cannot appeal to the central limit theorem since we have no information on the distribution of the weights $\mathbf{K}_{ii}$.

If we cannot give the form of the optimal pre-stego signal due to this fact, we can however give security guarantees for a given pre-stego signal. To do so, we use the result of Cachin's work on steganographic security [11], which shows, using a simple data-processing inequality, that the performance of an optimal detector is upper bounded by the KL divergence between the distributions of the two hypotheses.

Instead of minimizing the power of LRT under an entropy constraint we will thus minimize the KL-divergence $D_{\mathrm{KL}}\,(p||q)$ under an entropy constraint:

$$\begin{cases} \min\limits_{\mathbf{E}} & D_{\mathrm{KL}}\,(p_{\boldsymbol{\Sigma}}||q_{\boldsymbol{\Sigma},\mathbf{E}}) \\ R & = \frac{1}{2}\log\,(2\pi e|\mathbf{E}|) \end{cases} \tag{10.B.1}$$

where R is the payload constraint in the continuous domain and $|.|$ the matrix determinant. Recall that we assume here $\boldsymbol{\Sigma}$ and $\mathbf{E}$ to be both positive definite matrices.

Note that the KL divergence between p and q is given by:

$$\frac{1}{2}(\mathrm{trace}\left((\boldsymbol{\Sigma}+\mathbf{E})^{-1}\Sigma\right) + \log\left(\frac{|\,\boldsymbol{\Sigma}+\mathbf{E}\,|}{|\Sigma|}\right) - n). \tag{10.B.2}$$

The main idea of this proof is that we can simplify the problem by working in a basis where the covariance of the cover noise is the identity matrix. This is done by showing that the KL-divergence is invariant when changing to this basis. We then express the Lagrangian of the system as a function of the eigenvalues of $\mathbf{E}$ and show that there exists a unique minimum for our system.

First, let $\mathbf{L}$ be the Cholseky decomposition of Σ such that:

$$\Sigma = \mathbf{L}\mathbf{L}^T. \tag{10.B.3}$$

Let us also define $\mathbf{E}_w$ as:

$$\mathbf{E}_w \triangleq \mathbf{L}^{-1}\,\mathbf{E}\,\left(\mathbf{L}^{-1}\right)^T. \tag{10.B.4}$$

We now want to show the following equality:

$$D_{\mathrm{KL}}\left(p_{\mathbf{\Sigma}} \,||\, q_{\mathbf{\Sigma},\mathbf{E}}\right) = D_{\mathrm{KL}}\left(p_{\mathbf{I}} \,||\, q_{\mathbf{I},\mathbf{E}_w}\right), \tag{10.B.5}$$

where I is the identity matrix (with relevant dimensions).

To do so, we begin by showing the equality for the trace term:

$$\begin{aligned}
\operatorname{trace}\left((\Sigma+\mathbf{E})^{-1}\Sigma\right) &= \operatorname{trace}\left(\mathbf{L}^T(\Sigma+\mathbf{E})^{-1}\mathbf{L}\right) \\
&= \operatorname{trace}\left(\left(\left(\mathbf{L}^T(\Sigma+\mathbf{E})^{-1}\mathbf{L}\right)^{-1}\right)^{-1}\right) \\
&= \operatorname{trace}\left(\left(\mathbf{L}^{-1}\left(\Sigma+\mathbf{E}\right)\left(\mathbf{L}^{-1}\right)^T\right)^{-1}\right) \\
&= \operatorname{trace}\left(\left(\mathbf{I}+\mathbf{L}^{-1}\,\mathbf{E}\left(\mathbf{L}^{-1}\right)^T\right)^{-1}\right) \\
&= \operatorname{trace}\left((\mathbf{I}+\mathbf{E}_w)^{-1}\right).
\end{aligned} \tag{10.B.6}$$

and secondly for the determinant term:

$$\begin{aligned}
|\Sigma+\mathbf{E}\,|\cdot|\Sigma^{-1}| &= |\mathbf{I}+\mathbf{E}\,\Sigma^{-1}| \\
&= |\mathbf{I}+\mathbf{E}\,\mathbf{L}(\mathbf{L}^{-1})^T| \\
&= |\mathbf{I}+\mathbf{E}_w\,|,
\end{aligned} \tag{10.B.7}$$

with the last line obtained using Sylvester's Law of determinant. This validates the invariance of the KL-divergence to our change of basis.

Using Eq (10.B.5), we rewrite the system in Eq (10.B.1) as:

$$\begin{cases}
\min\limits_{\mathbf{E}_w} & D_{\mathrm{KL}}\left(p_{\mathbf{I}} \,||\, q_{\mathbf{I},\mathbf{E}_w}\right) \\
R & = \frac{1}{2}\left(\log\left(2\pi e|\mathbf{E_w}|\right) - \log\left(2\pi e|\Sigma^{-1}|\right)\right).
\end{cases} \tag{10.B.8}$$

Now, let $\mathrm{k_i}$ be the i-th eigenvalues of $\mathbf{E}_w$, we have that:

$$\begin{aligned}
D_{\mathrm{KL}}\left(p_{\mathbf{I}} \,||\, q_{\mathbf{I},\mathbf{E}_w}\right) &= \frac{1}{2}\left(\operatorname{trace}\left((\mathbf{I}+\mathbf{E}_w)^{-1}\right)\right) \\
&\quad + \frac{1}{2}\left(\log\left(|\mathbf{I}+\mathbf{E}_w\,|\right) - n\right) \\
&= \frac{1}{2}\left(\sum_{i=1}^{n}\frac{1}{1+\mathrm{k_i}} + \sum_{i=1}^{n}\log\left(1+\mathrm{k_i}\right) - n\right)
\end{aligned} \tag{10.B.9}$$

Consequently the Lagrangian of Eq (10.B.8) is given by:

$$\begin{aligned}
\mathcal{L} &= \frac{1}{2}\left(\sum_{i=1}^{n}\frac{1}{1+\mathrm{k_i}} + \sum_{i=1}^{n}\log\left(1+\mathrm{k_i}\right) - n\right) \\
&\quad - \frac{\lambda}{2}\left(R - \log\left(2\pi e|\Sigma^{-1}|\right) - \sum_{i=1}^{n}\log\left(2\pi e\,\mathrm{k_i}\right)\right),
\end{aligned} \tag{10.B.10}$$

The derivative with respect to the eigenvalues $\mathrm{k_i}$ is given by:

$$\frac{\partial\mathcal{L}}{\partial\,\mathrm{k_i}} = \frac{\lambda}{2\,\mathrm{k_i}} + \frac{\mathrm{k_i}}{2\left(\mathrm{k_i}+1\right)^2}. \tag{10.B.11}$$

Solving for 0 and assuming $\lambda \neq -1$:

$$\begin{aligned} & \frac{\partial \mathcal{L}}{\partial \mathrm{k_i}} = 0 \\ \Leftrightarrow \quad & \frac{\lambda}{\mathrm{k_i}} + \frac{\mathrm{k_i}}{(\mathrm{k_i}+1)^2} = 0 \\ \Leftrightarrow \quad & \begin{cases} \mathrm{k_i} & = \frac{-\lambda - \sqrt{-\lambda}}{\lambda+1} \\ \mathrm{k_i} & = \frac{-\lambda + \sqrt{-\lambda}}{\lambda+1} \end{cases} \end{aligned} \tag{10.B.12}$$

Since $\mathbf{E}_w$ is positive definite, an acceptable solution must be both real and strictly positive. Therefore, we necessarily have $\lambda < 0$. Furthermore, notice that if $\lambda < -1$, then neither solutions are positive. Consequently we must have $-1 < \lambda < 0$ and the only positive solution is given by:

$$\alpha = \frac{-\lambda + \sqrt{-\lambda}}{\lambda + 1} \in \mathbb{R}^+, \tag{10.B.13}$$

with λ obtained by plugging α in the entropy constraint.

In conclusion, the Lagrangian admits a unique stationary point, which is a matrix with eigenvalues which are all equal. Since $\mathbf{E}_w$ is positive definite (hence normal), this implies that $\mathbf{E}_w$ is proportional to the identity matrix:

$$\mathbf{E}_w = \alpha \mathbf{I}. \tag{10.B.14}$$

To show that this solution is indeed a minimum, we compute the bordered Hessian $\mathbf{F}$ of the Lagrangian. Starting with the second derivative when $i = i$:

$$\frac{\partial^2 \mathcal{L}}{\partial \mathrm{k_i}^2} = -\frac{\lambda}{\mathrm{k_i}^2} - \frac{1}{(\mathrm{k_i}+1)^2} + \frac{2}{(\mathrm{k_i}+1)^3}. \tag{10.B.15}$$

At $\mathrm{k_i} = \alpha$, the first two terms are equal so we have:

$$\frac{\partial^2 \mathcal{L}}{\partial \mathrm{k_i}^2}|_{\mathrm{k_i}=\alpha} = 2(\alpha+1)^{-3} > 0. \tag{10.B.16}$$

Then, for all $i \neq j$:

$$\frac{\partial^2 \mathcal{L}}{\partial \mathrm{k_i}\, k_j} = 0. \tag{10.B.17}$$

and finally:

$$\frac{\partial^2 \mathcal{L}}{\partial \mathrm{k_i}\, \lambda} = \frac{\partial^2 \mathcal{L}}{\partial \lambda\, \mathrm{k_i}} = \frac{1}{\mathrm{k_i}} > 0. \tag{10.B.18}$$

When $\mathrm{k_i} = \alpha$, the bordered Hessian has a special structure:

$$\mathbf{F}|_{k_i=\alpha} = \begin{pmatrix} 0 & \alpha^{-1} & \alpha^{-1} & \dots & \alpha^{-1} \\ \alpha^{-1} & 2(\alpha+1)^{-3} & 0 & \dots & 0 \\ \alpha^{-1} & 0 & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ \alpha^{-1} & 0 & \dots & 0 & 2(\alpha+1)^{-3} \end{pmatrix} \tag{10.B.19}$$

By induction, this allows us to express the i-th leading principal minor as follow:

$$|\mathbf{F}_i| = -(i-1)\left(\frac{2}{(\alpha+1)^3}\right)^{i-2} \cdot \frac{1}{\alpha^2} < 0, \text{ for } i > 1. \tag{10.B.20}$$

Therefore, each leading minor is strictly negative. Since we have only one constraint, the solution is indeed the unique minimum as was to be shown.

Going back to the original system in Eq (10.B.1), the solution is obtained by:

$$\begin{aligned} \mathbf{E} &= \mathbf{L}\,\mathbf{E}_w\,\mathbf{L}^T \\ &= \mathbf{L}\alpha\mathbf{I}\mathbf{L}^T \\ &= \alpha\Sigma. \end{aligned} \tag{10.B.21}$$

11 Numerical results and analysis for Gaussian embedding schemes

We end the second part of this manuscript by comparing the performance of each of the algorithms we developed against a state-of-the-art steganalyzer as well as to the current state-of-the-art in side-informed JPEG steganography: SI-UNIWARD.

From this experimental study, we show when the use of more complex models of the noise of natural images brings more security *in practice* and when simpler models can be used without much loss. Finally, we provide a set of experiments to understand the impact of different type of model misspecification on empirical security, such as when the heteroscedastic model or the processing pipeline matrix are badly estimated.

11.1 Experimental setting

In this section, we present the experimental setting that will be used for all the experiments of this chapter.

The overall structure of standard performance evaluation experiments in steganalysis is as follows.

First, from each cover dataset, we generate 5 stego datasets, each with different but fixed payloads. For each dataset, note that the processing pipeline, including the quality factor of the JPEG compression, is fixed. The final datasets are then built by merging the cover dataset with each stego dataset. Secondly, we separate each final dataset into a training and testing set with respectively 70% and 30% of the images. We train one steganalyzer for each training set and test it on the corresponding testing set. The performance of each steganalyzer is quantified by using the minimum probability of error P_E under the assumption that the cover and stego classes are balanced:

$$P_E = \min_{P_{FA}} \frac{1}{2} \left(P_{FA} + P_{MD} \right). \tag{11.1.1}$$

Dataset We use the BOSSBase RAW dataset excluding the images taken with the M9 camera because of the peculiar distribution of its photonic noise (see [20, Figure 2]) which would lead to an imprecise estimation of the covariance matrix. From this dataset comprising 7240 RAW images taken with 6 different cameras, we produce two new datasets using two different processing pipelines: a linear processing pipeline and a processing pipeline close to what is used to obtain the standard BOSSBase dataset. Both these pipelines output 264×264

greyscale JPEG images. The details are exposed in Table 7.1. The cropping operation was not performed blindly: we used the Edge crop[1] algorithm that selects crops containing the greatest number of edges – that is the zone that should contain the most textured areas. This choice of cropping was made because it is known that SI-UNIWARD does not perform well on smooth images [32, Section V.A]. Therefore, using such a cropping algorithm allows us to compare our embedding schemes in a situation where SI-UNIWARD is not disfavored due to a suboptimal content choice.

[1] Available at *https://alaska.utt.fr/#material*

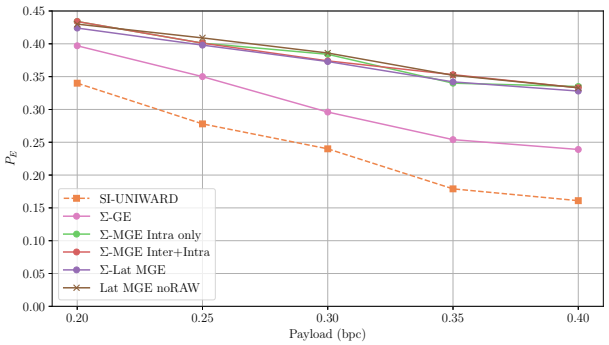
Steganography The different embedding schemes used are described in Table 11.1.

Paramater estimation for stegaography In Chapter 7, we presented two methods to estimate the covariance matrix of the noise model. The most precise method, presented in Section 7.3 requires access to the RAW file. When using this method, we prefix the steganographic scheme's name with Σ. When using the other method, presented in Section 7.4 which does not require access to the RAW file, we add the suffix *noRAW* to the scheme's name. Both these methods require an estimate of the parameters c_1 and c_2 of the heteroscedastic model described in Eq (5.2.8). They are estimated using the method described in Section 5.3. Finally, the matrix $\mathbf{H}$ representing the processing pipeline is estimated once for each processing pipeline using the method described in Section 7.4.2.

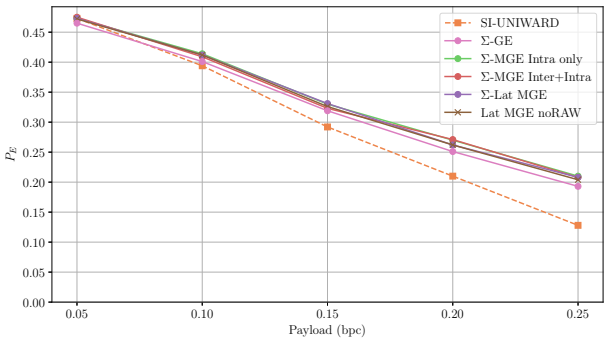
Steganalysis Steganalysis was performed using the current state-of-the-art in practical steganalyzers, namely neural networks using pretrained weights on ImageNet – see Section 3.2. More precisely, we used the Efficient-Net-b3 [77] architecture which showed to have excellent performance while also being extremely fast to train during the ALASKA2 competition [12]. We modified the architecture slightly so that the stride of the stem is equal to 1. The network was trained by training from the highest payload for 30 epochs, then 10 epochs for other payloads. The model for the highest payload was initialized with ImageNet weights. The base learning rate was fixed at 0.0005 and divided by 2 on loss plateau. The batch size was fixed to 24. The rest of the parameters are initialized in the same way as the original paper.

11.2 Performance against the state of the art

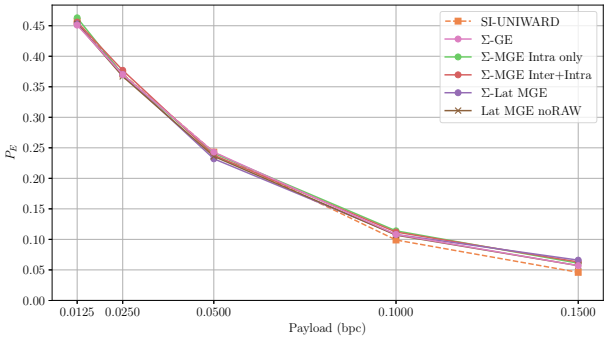
The results of the experiments are presented in Figure 11.1 for the Bosslike pipeline and in Figure 11.2 for the Linear pipeline. Beginning with QF100, regardless of the processing pipeline, there is a clear gap between the performance of SI-UNIWARD and the different variants of MGE. For the linear pipeline, the gap is, on average, of 6% and 9% in terms of absolute P_E for Σ-MGE Intra only and Σ-MGE intra



(a) Bosslike/QF100

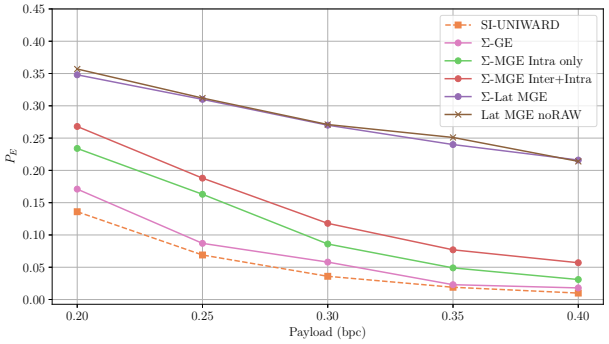


(b) Bosslike/QF95

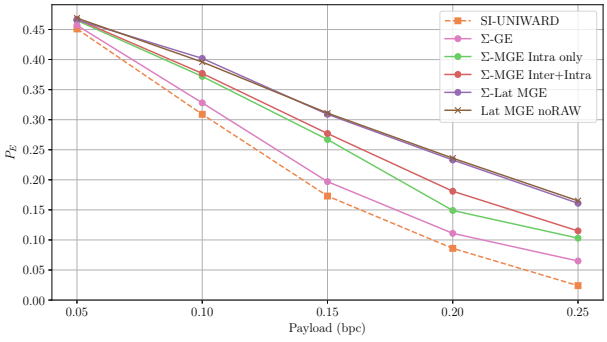


(c) Bosslike/QF75

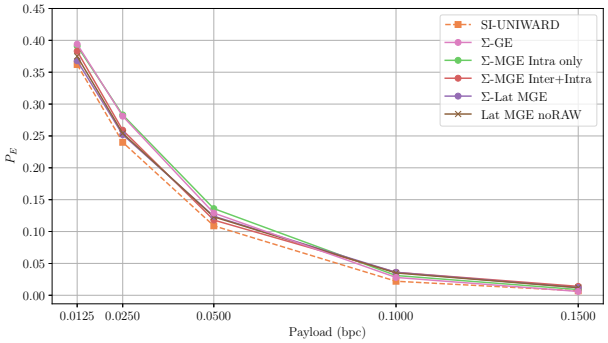
Figure 11.1: P_E as function of payload size for BossBase developed with the BOSS pipeline using Efficient-Net-b3.



(a) LIN/QF100



(b) LIN/QF95



(c) LIN/QF75

Figure 11.2: P_E as function of payload size for BossBase developed with the Linear pipeline using Efficient-Net-b3.

Name	Meaning
GE	Minimizes the power of of the MP detector in the continuous domain supposing every DCT coefficients to be independent as described in Chapter 9.
MGE Intra Only	Minimizes the KL divergence detector in the continuous domain supposing 8×8 DCT blocks to be independent.
MGE Intra+Inter	Minimizes the KL divergence in the continuous domain supposing 24×24 DCT macro-blocks to be independent.
Lat MGE	Minimizes the power of the MP detector in the continuous domain using lattice embedding as described in Section 10.3.3.
SI-UNIWARD	Side informed distortion based schemes as described in [42].

Table 11.1: Nomenclature of the embedding schemes

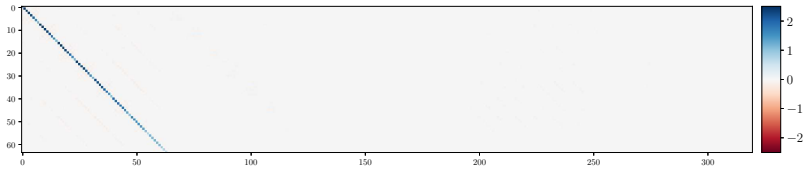
+ inter respectively. However, it is of 22% on average when using Σ-Lat MGE showing the importance of using the most precise model for this pipeline. The difference between the different models is less pronounced for the Bosslike pipeline with an average gain of 13.5% in terms of absolute P_E wrt SI-UNIWARD for the MGE schemes whatever the chosen correlation model. This is due to the fact that the downsampling operation removes most correlations between blocks due to the removal of neighboring pixels before the DCT transform.

At QF95, the difference between the different schemes becomes less pronounced for the Linear pipeline. However, there is still a gap in performance between these schemes and SI-UNIWARD. Compared with Σ-Lat MGE, there is an average gain of 10.5% wrt to SI-UNIWARD for the Linear pipeline and 4% for the BOSS pipeline. At QF75, every scheme performs approximately the same irrespective of the pipeline. This is most likely due to the fact that, at such a low QF, most of the variances are now close to 0. As such most of the performance of the steganography is likely due to the side-information related to the rounding errors.

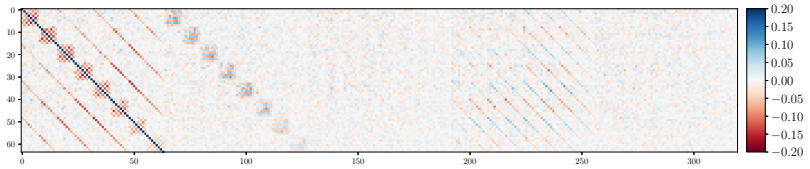
We also note that the implementation of Σ-Lat MGE which does not use the RAW file has a performance virtually identical with the original implementation showing that the assumptions on the processing pipeline we used in Section 7.4 are quite practical for a steganography context.

Finally, note that when using Σ-GE, which does not use any correlation between DCT coefficient, there is always a small gain with respect to SI-UNIWARD for both QF100 and QF95. However its performance are always subpar even compared to Σ-MGE intra only, showing the importance of taking these correlations into account.

One interesting thing to note here is the impact of the processing pipeline on the correlation structure necessary to obtain good performances. In the Bosslike case, the downsampling operation has made most inter-block dependencies very small compared to intra-block dependencies – see Figure 11.3. Consequently, using a more sophisticated model of dependencies does not bring any gain in performance. On the other hand, in the Linear pipeline case where all dependencies are preserved until the end of the pipeline since no downsampling is performed, there is quite a substantial gain when using the most sophisticated model.



(a) Original covariances



(b) Covariances after thresholding the dynamic range

Figure 11.3: Covariances of a central block (leftmost 64×64 block in the matrix) of DCT coefficients developed with the Bosslike pipeline with its neighboring blocks. The uppermost figure shows that the covariances are negligible compared to the variances. If we threshold the dynamic range we observe that the intra-block covariances dominate other inter-block covariances.

11.3 Impact of the quality factor on the covariance matrix

As we observed in the performance evaluation experiments, as we use lower quality factors for compressing a dataset, the lower is the gain in security when using models taking into account dependencies between DCT coefficients. This trend is so strong that at QF75, for a given pipeline, all the tested algorithms perform exactly the same despite wildly different models.

To explain this phenomenon we study the impact of the quantization of the covariance matrix. To illustrate our analysis, we provide in Figure 11.4 a series of covariance matrices of the same image block as the quality factor decreases. Note that these covariances are obtained **after** the rounding of DCT coefficients. One can observe that most of of the covariances become negligible as soon as quantization with QF95 is performed. Furthermore, most variances and covariances become zero since they are far smaller than the quantization step of 1 after being divided by the quantization matrix.

11.4 Impact of model misspecification

In this subsection, we study the impact of errors on the estimation of the covariance matrix. Three main types of errors can occur on the estimation:

1. errors on the heteroscedastic parameters c_1 and c_2,

2. errors on the pipeline matrix $\mathbf{H}$

3. and errors due to the saturation of the pixels.

Error on the heteroscedastic parameters Error on the heteroscedastic parameters can be studied analytically. Let c_1 and c_2 be the true parameters; $\hat{c}_1$ and $\hat{c}_2$ are the estimated parameters. The variances σ_i and estimated variances $\hat{\sigma}_i$ in the RAW domain are given by

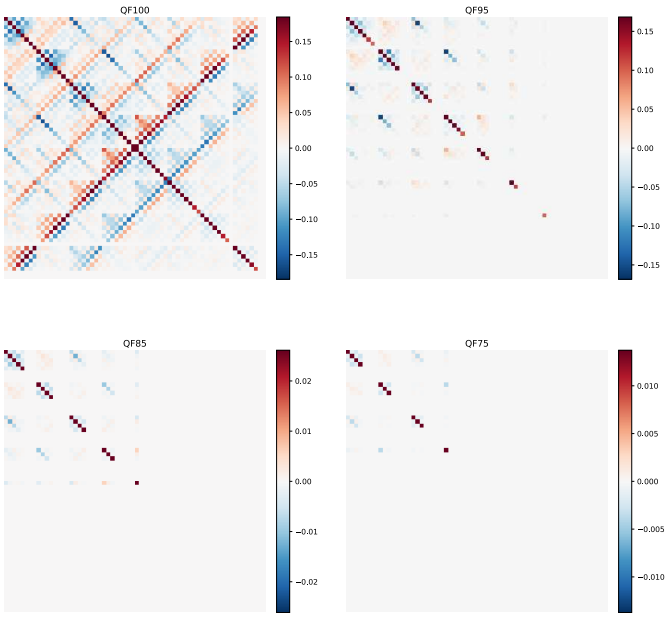


Figure 11.4: Covariance matrix of a block of rounded DCT coefficient of image *1200* in BOSSBase with the linear pipeline for different quality factors.

$$\sigma_i = c_1 \mu_i + c_2, \tag{11.4.1}$$

$$\hat{\sigma}_i = \hat{c}_1 \mu_i + \hat{c}_2. \tag{11.4.2}$$

Let α' be relative estimation error on the c_1 parameter:

$$\alpha' = \frac{\hat{c}_1}{c_1}. \tag{11.4.3}$$

We have that:

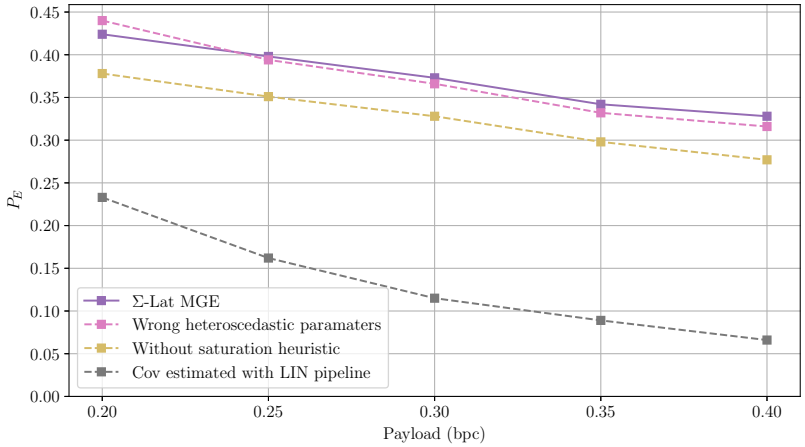
$$\hat{\sigma}_i = \alpha' \sigma_i + C, \tag{11.4.4}$$

where $C = \hat{c}_2 - \alpha' c_2$ is a constant which does not depend on the photo-site. Consequently, the resulting estimated covariance $\hat{\Sigma}_k$ in the developed domain is given by:

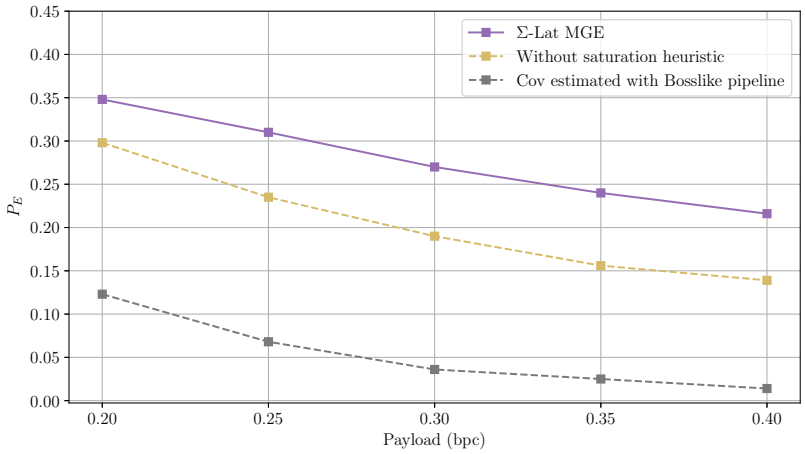
$$\hat{\Sigma}_k = \alpha' \Sigma_k + C\mathbf{H}\mathbf{H}^T. \tag{11.4.5}$$

As a particular case, note that if we have $\frac{\hat{c}_1}{\hat{c}_2} = \frac{c_1}{c_2}$, then the estimation error has no impact on the MGE steganography schemes, because a multiplicative error on the covariance does not change the optimal solution.

As a working example, let's take the Bosslike pipeline. We computed $\mathbf{H}\mathbf{H}^T$ for each camera and found that the average absolute value of the non-diagonal entries is of the order of 10^{-7} and 10^{-4} for the diagonal elements. If we set $\hat{c}_2 = 0$, the relative estimation error would



(a) Bosslike/QF100



(b) LIN/QF100

Figure 11.5: P_E as function of payload size for different the different heuristics and design decisions made in this paper.

then have to be very high – i.e, in the order of at least 100 – to begin to have an impact. To validate this analysis on the Bosslike pipeline, we repeated the experiments of Section 11.2 except that the covariance matrices were estimated by fixing $c_1 = 0.5$ and $c_2 = 0$ for every image. The results are given in Figure 11.5. As expected, they are extremely close to the original Σ-Lat-MGE results on this pipeline.

Error on the estimation of the processing pipeline and not taking clipping into account In this manuscript, we always assumed the steganographer had access to the correct processing pipeline for the estimation of the covariance matrix, even in the "no RAW" case. It would then be interesting to see the impact on security when using mismatched pipelines for the covariance estimation. To do so, we repeated the experiments of Section 11.2, but this time using the covariance matrices estimated with the Bosslike pipeline on the Linear dataset and vice versa. Similarly, we also repeated the experiments when using and not using the heuristic presented in Section 7.5 to take the clipping effect into account. The results are presented in Figure 11.5.

The results are clear: using strongly mismatched pipelines or not taking the saturation of pixels into account leads to near useless schemes,

showing the importance of having a good model of the pipeline and of the sensor in the first place.

11.5 Conclusion

In this final chapter, we evaluated all the Gaussian Embedding algorithms developed in the two previous chapters and compared them to SI-UNIWARD, the current state of the art of JPEG steganography.

We observed that, whatever the version of Gaussian Embedding we used, whatever the datasets, the algorithms performed better than SI-UNIWARD. However there was a large gap of performance between the different versions of the algorithms depending on how well the dependencies were taken into account. In the case of the Linear pipeline, the performance greatly increases (up to $+30\% in P_E$) the more dependencies are taken into account. On the other hand, due to the fact that inter-block dependencies are negligible in the case of the Bosslike pipeline, the Multivariate Gaussian Embedding taking only intra-block dependencies into account is sufficient to obtain the best performance.

We also observed that the difference between SI-UNIWARD tended to get smaller as we decrease the JPEG quality factor of the datasets. This was explained by observing that dependencies between DCT coefficients get destroyed due to the quantization and rounding operations.

Finally, we provided some analysis of the impact of model misspecification on the security performance of our algorithms.

General conclusion and perspectives

The two main tasks of this thesis could be summarized:

- The derivation of a realistic, yet simple, model of the sensor noise of natural images in the developed domain.
- A suite of steganographic algorithm which leverage this model of the noise to provide guarantees of performance.

In the first part of this manuscript, we tackled the first task by starting from first principles. We described how RAW images are captured and, following the classic work of A. Foi, we derived the heteroscedastic Gaussian noise model of the sensor noise in the RAW domain. We then went on to describe common operations in image processing. We provided a general of the full processing pipeline by observing that most of these operations could be well approximated by linear and stationary operations. We consequently modeled the full processing pipeline as a matrix that takes as input blocks of photo-sites and outputs blocks of DCT coefficients. By combining the model of RAW sensor noise and this model of the processing pipeline we obtained a multivariate Gaussian model of the noise in the developed domain. The parameters of interest of this model only depend on the camera sensor, ISO parameter and processing pipeline which were used to capture the image. For all the models presented in this first part, we also provided methods for estimating their parameters.

In the second part, we leveraged this model of the noise in the domain to solve the second task. We started by showing how to build a steganographic based on a statistical measure of detectability in the most simple setting. To do so, one needs a statistical model of the cover and the stego images. We started with a simple quantized, independent Gaussian model of the noise for the cover and a mixture of quantized Gaussian for the stego. We derived the optimal detector which discriminates between these two distributions, the likelihood ratio test, and computed analytically its power as a function of the cover and stego parameters. We then designed a steganographic algorithm that minimizes the power of this optimal detector. Following this step, we went on to design algorithms that follow this strategy but use the model derived in the first part of the manuscript. We began with an independent Gaussian model for the precover and the prestego but used the full multivariate model for the estimation of the variances of the model. The second algorithm used the full multivariate Gaussian model. In this most general case, we showed that the optimal

covariance of the prestego signal is proportional to the covariance of the precover noise.
We finally provided numerous results showing that these algorithms surpass the current state-of-the-art by a large margin in terms of security performance. We also showed that their performance matched the theoretical performance predicted by our theoretical analysis as long as the assumptions of the models were met. This allows for guarantees of security performance against the worst-case adversary (the optimal detector) under the linearity and stationarity assumptions of the processing pipeline. However, we also showed that even if these assumptions are not met, our algorithms still largely surpass the state-of-the-art.

The perspectives of this work are numerous and go from simple extensions to more theoretical questions on the link with other steganography methods. First, the theoretical analysis for the linear and stationary case is mostly complete. One missing part is in the proof of optimality of the covariance of the stego signal. Indeed, we only proved that a covariance matrix proportional to the cover noise covariance minimizes the KL-divergence, which itself bounds the power of the LRT. However, we conjecture that such a covariance should actually minimize the power of the LRT. At the very least, sharper bounds than the simple data processing inequality bound we used should be obtainable, using, for example, a Hoeffding-type inequality. Regarding extensions of our model to the non-linear and non-stationary case, we have already experimentally observed some interesting properties of the downsampling operation which could simplify the problem for all pipelines using this operation. Indeed, it seems that downsampling "linearizes" most of the non-linear operations, making the noise in the DCT domain close to Gaussian. Furthermore, the non-stationarity of the pipeline seems, usually, to result in a bias on the noise of individual DCT coefficients. These are some leads that could be followed in the future to try to make this model as general as possible and to truly provide a steganographic algorithm able to guarantee security performance in most cases.

Regarding some interesting connections with recent work, the breakthrough work on the Backpack algorithm showed that cost maps generated by this algorithm produced a stego signal with correlations reminiscent of the correlations produced by the processing pipeline studied in this manuscript. Seeing how our algorithm claims optimality for certain cases, it would be interesting to see if Backpack converges to costs leading to correlations identical to those of our algorithm or not. Another question of interest comes to mind after the work of Y. Yousfi in [85] which illustrated how different steganographic algorithms are detected differently by a deep neural network. In the case of J-UNIWARD, the DNN seems to make a decision by using the global image. In the case of J-MiPOD, on the other hand, the DNN seems

able to leverage local artifacts and make a decision based solely on a single block. Such information should be of interest for the design of our algorithms, especially in cases where we cannot claim optimality. Preliminary experiments on this question seem to show that a DNN will favor smooth areas to make a decision. This might be because these are the only areas where it is "easy" to estimate variances and maybe covariances.Finally, it should be noted that we emphasized the use of our noise model for applications in steganography. However one can easily imagine ways to improve steganalysis or even forensics methods if we assume the knowledge of the processing pipeline available. In the steganalysis case, this would allow verifying, per block, a deviation from the expected correlations between DCT coefficients. Another direction, in line with the observations of Chapter 4, would be the design of a classifier that, for a given image, automatically selects the best images in a training set in order to train itself to classify this image. This selection would be made by selecting images that have a covariance structure close to the covariance structure of the image to classify. This would ensure that the classifier is specialized for the right processing pipeline to minimize cover-source mismatch.

Résumé en français

Introduction générale

La stéganographie est la discipline s'intéressant à la conception d'algorithmes permettant la dissimulation d'information dans un support considéré comme anodin, appelé média de couverture, ou plus simplement cover.

La discipline s'est à ce jour principalement concentré sur l'utilisation de support numériques pour la dissimulation d'information, en particulier les images numériques. Ce choix résulte de la conjonction de trois propriétés intéressantes pour le stéganographe: l'omniprésence de ce médium, la facilité de son partage ainsi que sa facilité de modification (au contraire de la vidéo par exemple). Une fois l'information dissimulée dans une image, on appelle cette dernière *objet stéganographique*, ou plus simplement stégo. On suppose ensuite que cet objet stégo est envoyé à un tiers en passant par canal publique non destructif et non bruité.

En pratique la stéganographie cherche donc à concevoir des algorithmes permettant d'insérer de l'information dans une image numérique sans en l'altérer visuellement. Cependant, la minimisation de l'impact visuel de l'insertion est très insuffisante en pratique. En effet, on considère que le stéganographe est confrontée à une adversaire, la stéganalyste, capable d'employer diverses techniques statistiques pour permettre la détection d'information dans les images. Nous référons au Chapitre 1 pour un exemple d'attaque concrète sur un schéma d'insertion naïf ne prenant en compte que l'impact visuel. Le but du stéganographe sera donc très précisément de minimiser la détectabilité de son schéma d'insertion lorsque confronté à un certain type d'adversaire.

Comme nous venons de l'indiquer, la discipline adverse de la stéganographie est la stéganalyse dont le rôle est développer des techniques permettant la détection d'information dissimulée. Le contexte classique d'étude de la stéganalyse et de la stéganographie suppose que que la stéganalyste est *passive*, c'est à dire qu'elle ne modifiera pas les images envoyées par le stéganographe pour empêcher un tiers d'en extraire le message. De plus, il est important de noter que le rôle de la stéganalyste est la *détection d'information cachée* et non pas son extraction. De plus, on supposera que le but de la stéganalyste est de déterminer pour chaque image qui lui est présentée si celle ci est cover ou stego. Son rôle ne sera donc pas ici de savoir si un acteur donné est stéganographe ou non.

11.5.1 *État de l'art en stéganographie*

Les travaux actuels en stéganographie reposent sur la conception de fonctions de coût efficaces. Il existe actuellement deux voies principales pour leur conception : heuristique et statistique.

La voie heuristique a été de loin la plus populaire, l'approche étant basée sur la conception de fonctions de coût évaluées sur le stéganalyste le plus efficace du moment. Les performances de ces systèmes ne sont donc validées qu'empiriquement. Cette approche a donné lieu à des schémas très efficaces dans le domaine spatial [42, 56] et dans le domaine JPEG [42, 39].

Malgré ce succès et le fait que la famille UNIWARD, par exemple, soit toujours compétitive à ce jour, cette approche présente plusieurs limites. Tout d'abord, le fait que ces techniques ne soient validées qu'empiriquement conduit parfois à d'énormes différences de performance au fur et à mesure que les standards d'évaluation des performances évoluent et que les techniques de stéganalyse deviennent plus sophistiquées - voir par exemple l'observation récente sur UERD contre DNNs [8] ou l'impact du choix des jeux de données sur la performance de ces schémas [69, 35]. Deuxièmement, et plus généralement, cette approche utilise des fonctions de coût qui n'ont pas *apriori* de lien clair avec une détectabilité théorique ou empirique. Par conséquent, cette approche donne très peu d'indications sur la raison pour laquelle une stratégie fonctionne et pourquoi une autre ne fonctionne pas. Pire encore, cela rend l'approche incapable de donner des garanties théoriques en termes de détectabilité.

La deuxième voie est basée sur la minimisation d'une quantité qui est directement liée *apriori* à la détectabilité théorique et/ou empirique. La stratégie la plus récente qui suit cette voie, l'insertion adverserielle, tente directement de minimiser la détectabilité empirique sur une base d'image donnée en modifiant itérativement la fonction de coût afin de minimiser les performances d'un classificateur. Ce classificateur se met également à jour à chaque étape afin de minimiser la performance du sténographe. Bien que cette approche ai largement fait ses preuves [77, 6], elle a ses propres limites. Celles-ci sont : un coût de calcul élevé et le besoin de grandes bases d'images, nécessaires pour obtenir une fonction de coût de haute qualité. De plus, elle ne résout pas le problème des garanties théoriques pour bases d'images non testées.

Pour ces raisons, nous travaillons dans ce rapport avec une autre stratégie qui fonctionne en minimisant la détectabilité théorique d'un détecteur optimal. Cette approche est apparue pour la première fois dans [29] et a abouti à la conception de l'algorithme MiPOD [69]. Le cadre conçu pour MiPOD est basé sur la théorie des tests d'hypothèse. L'idée principale est de représenter le problème de la stégananalyse

comme un simple test entre deux hypothèses : $\mathcal{H}_0$, l'image examinée est une couverture ou $\mathcal{H}_1$, c'est un stego. En utilisant cette théorie, on peut montrer sous certaines conditions qu'il existe un détecteur optimal et calculer ses performances. Le but du stéganographe est alors de minimiser la puissance de ce détecteur sous la contrainte d'insérer un message d'une taille donnée dans le média de couverture.

Toutefois, pour être en mesure de poser le problème dans ce contexte, il faut disposer d'un modèle des images de couverture et un modèle des images stégo. Les limites de MiPOD et de l'algorithme MG qui l'a précédé, provenaient principalement de leur choix de modèle de bruit. En effet, ces deux algorithmes modélisaient les images naturelles comme des pixels corrompus par un bruit gaussien indépendant, mais non identique. De plus, ils s'appuyaient sur l'hypothèse de la limite de quantification fine, qui stipule que les variances des pixels sont supérieures à un. Ces deux hypothèses sont toutes deux erronées en pratique. Premièrement, il est connu que les pixels voisins sont, le plus souvent, corrélés en raison de l'impact de la chaîne de traitement [75, 32, 31]. Deuxièmement, la limite de quantification fine est souvent violée dans les zones sombres d'une image en raison de la nature hétéroscédatique du bruit [27], [82, Chapitre 5, Section 5] qui implique que la variance dans ces zones est très faible. Cette hypothèse est encore plus problématique si l'on veut étendre l'approche dans le domaine JPEG, car la quantification des coefficients DCT peut considérablement réduire la variance avant l'opération d'arrondi de la compression JPEG. Malgré ces limitations, MiPOD obtient des performances proches de l'état de l'art dans le domaine spatial, démontrant ainsi le mérite de l'approche.

État de l'art en stéganalyse

Comme nous venons de le voir, la stéganographie a développé des techniques de plus en plus puissantes afin de réduire le nombre d'éléments modifiés dans une image et de rendre ces modifications les moins détectables possibles. Ces développements ont commencé à avoir lieu au cours de l'année 2010 et se poursuivent aujourd'hui avec l'avènement de techniques d'apprentissage des coûts basées sur les réseaux de neurones [6]. Comme on peut s'y attendre, le développement de la stéganalyse reflète cette histoire et est alimenté par le développement de nouvelles techniques de stéganographie.

La méthodologie qui marque l'entrée de la stéganalyse dans sa phase moderne est l'utilisation combinée de l'ensemble de caractéristiques (feature sets) SPAM [65] et de Support Vector Machine (SVM). Les caractéristiques SPAM, développées en 2010, sont basées sur un modèle de chaîne de Markov des dépendances entre les pixels et ont servi de base à l'algorithme stéganographique HUGO. Bien que largement obsolète de nos jours, cette technique contient les deux principaux éléments qui ont été perfectionnés jusqu'en 2016 : (1) l'utilisation d'un

modèle heuristique de haute dimension des dépendances entre les pixels permettant l'élaboration (manuelle) d'un ensemble de caractéristiques qui sont ensuite transmises à (2) un classificateur formé dans un cadre d'apprentissage supervisé. Cet ensemble de techniques est présenté, dans le manuscrit, dans la section 3.1.

L'année 2015 marque l'entrée de la stéganalyse dans une nouvelle phase. En effet, celle-ci commence à tirer parti des récentes avancées en matière de traitement de l'image pour construire le premier réseau de neurones convolutifs conçu explicitement pour la stéganalyse : le CNN à neurones gaussiens [68]. Cependant, ses piètres performances par rapport aux modèles à haute dimension de l'époque ont eu pour conséquence que l'approche par apprentissage profond n'est pas devenue populaire avant l'avènement de Xu-Net [84] en 2016 et de son extension au domaine JPEG en 2017 [83]. Les premières itérations de Xu-Net se contentaient d'intégrer les meilleures pratiques de conception de réseaux de neurones : l'utilisation de la normalisation par lots [44] et de la fonction d'activation ReLU [59]. L'extension au domaine JPEG a permis d'ajouter des connexions raccourcies entre les couches afin d'éviter le problème du gradient évanescent, un problème courant dans les réseaux profonds où le gradient en un point du réseau devient si faible que le réseau cesse de converger. Cependant, Xu-Net a conservé une particularité de conception du CNN à neurones gaussiens, à savoir l'utilisation d'une couche de prétraitement. Cette couche consiste en une série de filtres passe-haut qui rappellent ceux utilisés pour construire les ensembles de caractéristiques à haute dimension en stéganalyse. Ils sont destinés à séparer le contenu de l'image du bruit pour faciliter la détection d'un signal stégo.

Ce choix de conception a été entièrement abandonné par la grande étape suivante en stéganalyse basée sur les réseaux de neurones : SRNet [8]. L'idée majeur derrière SRNet était de disposer d'une architecture entièrement automatique et universelle pour la stéganalyse. Par conséquent, SRNet est principalement fondé sur une architecture ResNet classique [40] sans aucune couche de prétraitement. Son principal choix de conception est d'empêcher tout *pooling* dans les premières couches du réseau afin de conserver autant d'informations que possible sur les variations locales de l'image. Ce choix est motivé par le fait que, contrairement aux problèmes classiques de la vision par ordinateur, la stéganalyse est confrontée à la détection de signaux très faibles et localisés. Au moment de la rédaction de ce manuscrit, le dernier développement en stéganalyse peut être observé dans les résultats du concours de stéganalyse ALASKA2 [12] qui a eu lieu en 2020. Lors de ce concours, la plupart des équipes de tête ont utilisé l'architecture EfficientNet [76] qui, contrairement à toutes les autres architectures de réseaux de neurones utilisées jusqu'alors en stéganalyse, n'était pas spécifiquement conçue pour la stéganalyse. La principale nouveauté était l'utilisation de l'apprentissage par transfert. EfficientNet a d'abord été initialisé avec des poids obtenus en l'entraînant sur ImageNet pour d'autres tâches. Il a ensuite été affiné en l'entraînant à classifier des images cover et des images stégo. Cela a conduit à des per-

formances exceptionnelles, dépassant SRNet. D'autres améliorations ont été obtenues par des modifications spécifiques de l'architecture – voir [86].

Problématique et contributions

Le but fixé de cette thèse était de concevoir un algorithme de stéganographie possédant des garanties en terme de détectabilité. Donner de telles garanties impose d'être en mesure quantifier exactement la détectabilité d'une image dans laquelle de l'information a été cachée.

Pour ce faire, nous nous somme dirigés vers une modélisation du problème utilisant les tests d'hypothèses. En effet, il est possible d'interpréter le problème d'un stéganalyste souhaitant classifier en deux classes d'images, cover ou stégo, comme un test d'hypothèse simple entre deux classes. Une fois ce test correctement spécifié, il est possible pour le stéganographe de construire un signal minimisant la puissance de ce test sous contrainte de cacher un message de taille donnée.

Cependant la construction d'un tel test nécessite l'accès à des modèles bien spécifiés pour les deux classes d'images cover et stégo. Bien plus, pour obtenir de réelles garanties en terme de détectabilité, ces modèles doivent êtres suffisamment fins pour que la puissance du test soit effectivement lié à une détectabilité empirique.

La problématique de ce manuscrit est donc triple :

- Déterminer les aspects clés d'une image naturelle qui ont un impacts dans les performances des algorithmes de stéganographie et de stéganalyse.
- Modéliser ces aspects clés pour obtenir un modèle fin du bruit des images naturelles.
- Exploiter ce modèle pour la construction d'un schéma d'insertion donnant des garanties de détectabilité.

Ce manuscrit est donc le fruit de trois contributions principales – chacune liée à un point de la problématique – que nous résumons très succinctement ici, le lecteur ou la lectrice intéressé·e pourra se référer aux chapitres qui leurs sont dédiées pour plus de détails.

Contribution 1 : Une étude empirique d'envergure de l'impact des différents paramètres d'acquisition et de développement des images sur les performances en stéganalyse – voir Chapitre 4.

Cette étude a mené à une définition concrète du concept de **source** en stéganographie.

Ce concept avait historiquement pour fonction d'expliquer les différences de performances de la stéganalyse sur des photographies prises dans des conditions différentes. Cependant, les paramètres d'importances n'avaient jamais été identifiés. Cette étude a eu pour résultat de rapporter le concept de sources à trois paramètres : le capteur de l'appareil

de photo utilisé, l'ISO avec lequel la photo a été prise et la chaine de développement utilisée.

Comme nous le montrons dans la seconde contribution ces trois paramètres caractérisent pleinement la structure du bruit du capteur. Cette structure est le point crucial pour la conception d'une stéganographie possédant des garanties de détectabilité.

Contribution 2 : Une modélisation générale du bruit de capteur des images naturelles ainsi qu'une méthode pour en estimer les paramètres – voir la Partie I de ce manuscrit.

Suite à l'étude de la première contribution, l'effort s'est concentré sur la modélisation du bruit de capteurs des images du domaine RAW jusqu'au domaine JPEG, qui est le domaine d'intérêt en stéganographie au vu de la prévalence de ce format. Cette modélisation part d'un modèle classique en débruitage des images RAW, fondé sur un modèle gaussien hétéroscedastique [27] – voir le Chapitre 5. Ainsi le bruit de chaque photo-site du capteur est indépendant mais pas identiquement distribué par rapport aux autres photo-sites. Les paramètres de ce modèle sont complètement déterminés par le type de capteur ainsi que l'ISO utilisé. Pour passer de ce modèle du bruit, dans le domaine RAW, à un modèle général du bruit dans le domaine JPEG, il est nécessaire de modéliser ensuite la chaine de développement – voir le Chapitre 6. En partant d'une hypothèse de linéarité et de stationnarité de la chaine de développement, on peut modéliser le bruit dans le domaine JPEG par un modèle gaussien multivarié – voir le Chapitre 7. Ce modèle pose toute l'importance de la covariance du bruit en stéganographie et donc de la nécessité de prendre en compte la corrélations entre des coefficients DCT voisins lors de l'insertion – un aspect jusque là peu étudié dans la discipline.

Contribution 3 : Un schéma d'insertion fondé sur le modèle de la seconde contribution, donnant des garanties de détectabilité pour une large classe de chaines de développement et dépassant largement les performances de l'état de l'art – voir la Partie II de manuscrit.

Pour utiliser ce nouveau modèle de bruit dans le cadre d'une stéganographie donnant des garanties de détectabilité, il est d'abord nécessaire de choisir un adversaire – un stéganalyste. Dans le cadre des tests d'hypothèse, l'adversaire le plus naturelle est un adversaire du type "pire-cas" qui connais *apriori* tous les paramètres du modèle, ainsi que les paramètres du signal stéganographique.

Dans ce contexte nous montrons que le signal optimal pour le stéganographe est un signal gaussien multivarié dont la covariance est directement proportionnelle à la covariance du bruit de l'image utilisé – le facteur de proportionnalité dépendent uniquement de la taille du message souhaitant être dissimulé.

Enfin nous montrons comment échantillonner rapidement ce signal stégo, sous contrainte que le message final aie une taille donnée, en

utilisant une décomposition de Cholesky de la matrice de covariance du bruit de l'image cover.

Impact de la source sur les performances en stéganalyse

Nous résumons dans cette section la première contribution de cette thèse, traitant de l'impact du phénomène de *cover-source mismatch* ou CSM en stéganalyse ainsi que de la définition d'une source d'image en stéganographie et stéganalyse.

Le phénomène du cover-source mismatch a été documenté pour la première fois dans [37] où il a été observé que lorsqu'un un classificateur est entrainé sur un ensemble de données contenant des images prises uniquement avec un appareil photo *AP1* puis que ce même classificateur est testé sur un second ensemble de données construit uniquement à l'aide d'un appareil *AP2*, on observe une énorme perte de performance par rapport à celles attendues si le classificateur était testé sur un ensemble de données construit uniquement avec *AP1*.

Ce problème est devenu encore plus évident lors de la compétition BOSS où les organisateurs ont ajouté des images dans l'ensemble de test qui ont été prises avec un appareil photo absent de la base d'entraînement. Cela a entraîné une forte baisse des performances de stéganalyse sur ces mêmes images. Ce qui est souvent moins mis en évidence, c'est que ces images aberrantes n'ont pas seulement été prises avec un appareil photo inconnu, mais qu'elles ont toutes subi une double compression JPEG, contrairement aux autres images qui ont simplement été compressées une fois. Cela montre que la chaîne traitement d'image peut également jouer un rôle important dans les performances de la stéganalyse.

Le travail de [50] étudie ce phénomène en se concentrant principalement sur l'impact de la diversité des appareils photos dans une base d'image. Cette étude montre cependant l'impact plus important des différentes chaînes de traitement d'image sur les effets du CSM. Les mêmes auteurs ont également étudié l'effet de différents algorithmes de redimensionnement sur la sécurité stéganographique, démontrant ainsi le rôle clé de la chaîne de traitement.

Cependant, jusqu'à notre série de travaux [34, 35], il n'y avait pas eu d'étude systématique de l'impact des différentes propriétés des images naturelles sur le CSM. Ce manque de connaissances sur les causes sous-jacentes de ce phénomène a conduit à l'impossibilité de concevoir une méthode théoriquement solide pour construire un ensemble d'entraînement qui permettrait à la fois de (1) limiter la perte de performance sur des données absentes de la base d'entrainement tout en (2) limitant le nombre d'échantillons nécessaires à cet entrainement. Il était du moins important de comprendre sur quels types d'images on pouvait s'attendre à ce qu'un classificateur entraîné sur une base d'image donné soit performant.

Notre stratégie de départ pour limiter l'impact du CSM consistait à

identifier la *source* des images dans un ensemble de test donné afin de construire un ensemble d'entraînement sur mesure. Mais cela soulève la question suivante : quelle est la source d'une image ?

11.5.2 *Définition d'une source*

La première étape dans la définition de la source d'une image est de lister et de catégoriser tous les paramètres jouant un rôle lors de la génération d'une image naturelle. Nous retiendrons seulement trois catégories pour décrire tous ces paramètres potentiels :

- *Contenu* : Il s'agit de la scène représentée par l'image. Plus précisément, il s'agit du signal émis et reçu sous forme de lumière par le capteur de l'appareil photo.
- *Paramètres d'acquisition* : Il s'agit de tous les paramètres de l'appareil photo qui sont fixés afin de capturer le signal d'une scène donnée. On pensera par exemple au modèle de capteur de l'appareil photo, le réglage ISO, le choix de l'objectif, le temps d'exposition ou l'ouverture.
- *Paramètres de traitement* : Il s'agit de tous les paramètres de tous les algorithmes utilisés pour transformer l'image RAW capturée par l'appareil photo en image finale traitée.

Cette catégorisation nous amène à la définition suivante de la source d'une image :

Source Une *source* peut être définie comme un ensemble de paramètres d'acquisitions combiné à un ensemble de paramètres de traitement qui génèrent des images tels que, pour un contenu donné, la succession des acquisitions forme un signal stationnaire.

Cette définition étant beaucoup trop générique pour aider le stéganalyste à concevoir ses bases d'images, nous avons effectué une étude expérimentale systématique de chacun de ces paramètres pour étudier leur impact sur la performance d'un classificateur à l'état de l'art – voir le Chapitre 4 de manuscrit.

De ces expériences, nous avons vu que l'appareil photo et l'ISO ont un fort impact sur le phénomène du CSM et que même deux appareils photo ayant le même réglage ISO ne seront pas forcément cohérents entre eux. Deuxièmement, l'impact de la chaîne de traitement de l'image semble être le plus importants sur l'ensemble des paramètres étudiés.

Par conséquent, nous proposons une définition empirique d'une source comme étant :

Definition 11.5.1 (Source II)**.** *La source d'une image est la combinaison d'un capteur d'appareil photo, d'un réglage ISO et d'une chaîne de traitement utilisée pour capturer et produire l'image finale traitée.*

La première partie de ce manuscrit vise très justement à donner une justification théorique à cette définition par la construction d'un modèle statistique.

Modèle de bruit des images naturelles

Nous développons succinctement dans cette section la seconde contribution de cette thèse, à savoir une modélisation fine du bruit de capteur des images naturelles ainsi que l'estimation des paramètres de ce modèle.

En partant des enseignements de la première contribution, nous savons que le bruit de capteur des images naturelles ne dépend que de trois paramètres : le capteur de l'appareil de photo, l'ISO avec lequel la photo a été prise et la chaine de développement.

Nous commençons par modéliser la chaine de développement avant de dériver le modèle de bruit dans le domaine JPEG à partir du modèle de bruit dans le domaine RAW.

Nous nous référerons à des bloc de taille 8×8 comme des **blocs de coefficients DCT** tandis que les groupes de blocs de coefficients DCT comme des **macro-blocs**. Notez également que nous regroupons les différents termes photo-sites (domaine RAW), pixels (domaine développé avant compression) et coefficients DCT (domaine JPEG) sous le terme général d'éléments d'image pour faciliter la présentation.

Hypothèses pour la modélisation de la chaine de développement

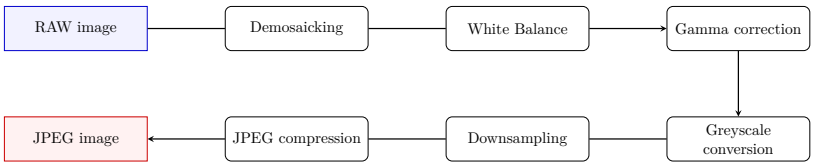


Figure 11.6: Ordre des différentes opérations de traitement d'images présentées dans ce chapitre. Pour un exemple de chaîne traitement d'image plus complet – comme celui utilisé dans le logiciel open-source Rawtherapee – voir https://rawpedia.rawtherapee.com/Toolchain_Pipeline.

Pour pouvoir modéliser le bruit de capteur dans le domaine JPEG, nous avons besoin d'un modèle de la chaine de développement. Pour conserver une certaine facilité de traitement mathématique tout en restant aussi général que possible, nous modélisons la chaine de développement comme un opérateur linéaire et stationnaire, agissant sur des macro-blocs d'éléments de l'image

Les opérations linéaires sont des fonctions f de la forme :

$$f(\sum_k \mathbf{x}_k) = \sum_k f(\mathbf{x}_k), \tag{11.5.1}$$

$$\forall \alpha \in \mathbb{R}, f(\alpha \mathbf{x}) = \alpha f(\mathbf{x}). \tag{11.5.2}$$

Les opérations stationnaires sont des opérations qui sont appliquées de manière identique pour chaque entrée de l'opération. Par exemple, si l'opération prend en entrée des blocs de sites photographiques, alors chaque bloc doit être traité de la même manière. L'algorithme de dématriçage PPG, décrit dans le Chapitre 6, est un exemple d'algorithme non stationnaire, car il effectue différentes opérations sur les blocs en fonction du gradient du photo-site central par rapport à ses voisins.

En partant de ces hypothèses, nous pouvons représenter la chaine de développement comme une matrice $M^2 \times N^2$, notée $\mathbf{H}$ pour le

reste de ce rapport. Dans ce rapport, $\mathbf{H}$ modélise la chaine de traitements jusqu'à la transformée DCT de la compression JPEG mais avant l'opération d'arrondis – qui est non-linéaire.

Le choix de M et N sont à la discrétion du modélisateur. Plus ces paramètres sont élevés, plus le modèle de bruit sera fin, car il prendra en compte les corrélations entre coefficients DCT de plus en plus éloignés.

Modèle de bruit des images RAW

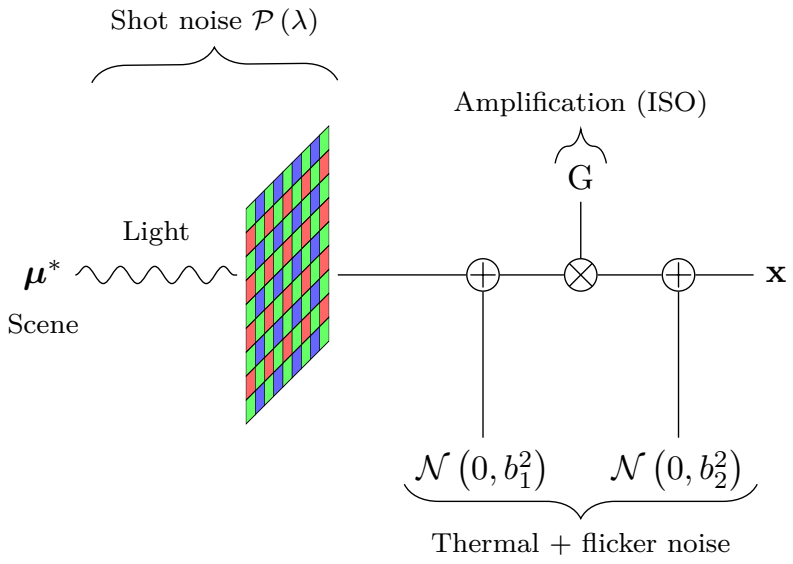


Figure 11.7: Diagramme du modèle d'acquisition d'image dans le domaine RAW. La lumière de la scène est capturée par les photo-sites du capteur d'image. La nature quantique de la lumière introduit du bruit sous la forme de "bruit de grenaille" (shot noise) qui suit une distribution de Poisson. D'autres bruits sont introduits par le circuit sous la forme de bruits thermiques. Le signal électrique est ensuite amplifié par une valeur G contrôlée par le réglage ISO de l'appareil photo.

Nous modélisons maintenant le bruit des images naturelles en partant du domaine RAW pour arriver jusqu'au domaine DCT (avant arrondis).

Pour commencer, étudions comment une image RAW, c'est à dire une image en sortie de capteur, avant tout traitement, est capturée. La chaîne d'acquisition des images RAW est résumé dans la figure 11.7.

Les capteurs d'appareils photo usuellement utilisés fonctionnent tous sur le même principe : convertir un signal lumineux (photons) en un signal électrique (électrons). Ils s'appuient donc sur une grille de photo-diodes, que nous appelons *photo-sites*, pour détecter les photons arrivant à différentes positions dans l'image. Chaque photo-site ne peut capturer qu'une seule couleur, généralement l'une des couleurs rouge, verte ou bleue (RVB). Cela est dû à un motif de masquage disposé sur les photo-sites qui prend généralement la forme du motif de la grille de Bayer.

Le nombre moyen d'électrons générés par un photon est appelé *efficacité quantique* et dépend des caractéristiques du capteur d'images. Une autre caractéristique dépendant du matériel est un offset sur chaque photo-site : les charges collectées de chaque site photo sont toujours compensées par une valeur positive $p_0 \in \mathbb{R}^+$. Ce courant électrique est ensuite envoyé dans différents circuits selon le type de

capteur – voir [60, 51] pour les références. Au cours de cette opération, le signal est amplifié par une valeur positive $G > 1$ qui dépend du réglage ISO choisi par l'utilisateur au moment de la capture de l'image.

À chaque étape de ce processus, du bruit est introduit dans le signal. Deux sources majeures de bruit peuvent être déduites de la description du système : le bruit lié à la nature quantique de la lumière lors de son arrivée sur les sites de prise de vue (bruit de grenaille) et le bruit introduit par les différents composants du circuit du capteur (bruit thermique).

A partir de ce modèle, il est possible de construire un modèle statistique du bruit dans le domaine RAW. Nous suivons pour cela les travaux maintenant classiques de Foi [27] en adoptant un modèle hétéroscedastique du bruit de capteur – voir le Chapitre 5 pour une dérivation complète du modèle.

Nous représentons donc les photo-sites d'une image RAW comme une séquence de m variables aléatoires suivant une loi gaussienne :

$$\mathbf{x}_i \sim \mathcal{N}\left(\mu_i, \sigma_i^2\right) \ ; \ \sigma_i^2 = c_1 \mu_i + c_2, \tag{11.5.3}$$

où μ_i serait la i-ième valeur du photo-site s'il n'avait pas été corrompu par un bruit d'acquisition. Dans ce modèle, la variance dépend linéairement μ_i à travers deux paramètres c_1 et c_2. Ces deux paramètres dépendent exclusivement du capteur et de l'ISO de l'appareil photo utilisé.

Pour faciliter la suite de la présentation, nous réinterprétons Eq (11.5.3) en modélisant conjointement les macro-blocs de photo-sites de tailles $M \times M$:

$$\mathbf{x}_k \sim \boldsymbol{N}(\boldsymbol{\mu}_k, \mathrm{diag}(\boldsymbol{\sigma}_k)), \tag{11.5.4}$$

où diag est la fonction prenant en entrée un vecteur et retournant la matrices diagonales des valeurs de ce vecteur.

Modèle de bruit dans le domaine développée

L'image RAW est ensuite développée en appliquant la chaine de développement $\mathbf{H}$ sur chaque macro-bloc.

En supposant la stationnarité de la chaîne de traitement, nous pouvons écrire le macro-bloc développé comme suit :

$$\mathbf{y}_k = \mathbf{H}\,\mathbf{x}_k\,. \tag{11.5.5}$$

De plus, en supposant la linéarité de la chaîne de traitement et en utilisant les propriétés de la distribution gaussienne, il est immédiat que $\mathbf{Y}_k$ suit une distribution gaussienne multivariée :

$$\mathbf{y}_k \sim \mathcal{N}\left(\mathbf{H}\boldsymbol{\mu}_k, \Sigma_k\right), \tag{11.5.6}$$

avec la covariance Σ_k obtenue par l'équation :

$$\Sigma_k = \mathbf{H}\mathrm{diag}\left(\boldsymbol{\sigma}_k\right)\mathbf{H}. \tag{11.5.7}$$

En résumé, sous les hypothèses de notre modèle, le bruit des macro-blocs dans le domaine développé suit un modèle gaussien multivarié. Ce modèle dépend de seulement trois paramètres : les paramètres hétéroscédastiques c_1 et c_2, qui ne dépendent que du capteur d'image et du réglage ISO, et le pipeline de traitement, ici modélisé par une matrice $\mathbf{H}$. Cela nous permet de donner une justification théorique de la définition de la source donnée lors de notre première contribution.

Estimation de la matrice représentant la chaîne de développement

Dans ce manuscrit, nous supposons toujours que le stéganographe a un accès de type "boîte noire" à la chaîne de traitement qui a été utilisé pour générer l'image cover qu'il utilisera pour cacher de l'information. En d'autres termes, le stéganographe n'a aucune connaissance directe du fonctionnement interne de cette chaîne de traitement mais peut l'utiliser pour développer des images.

À partir de cette hypothèse, nous souhaitons estimer la matrice $\mathbf{H}$ de cette chaîne de traitement.

Pour ce faire, nous générons d'abord une image constante $\bar{\mathbf{x}}$ corrompue par un bruit gaussien centré indépendant et identiquement distribué avec une variance σ^2 :

$$\bar{x}_i \sim \mathcal{N}\left(c, \sigma^2\right). \tag{11.5.8}$$

La valeur constante c peut, en théorie, être choisie arbitrairement. Cependant, la plupart des logiciels de traitement d'images coupent les valeurs inférieures à zéro. Par conséquent, nous choisissons toujours une valeur constante proche de la valeur médiane de la plage d'une image RAW donnée, soit généralement 2000 ou 8000.

L'image $\bar{\mathbf{x}}$ est ensuite développée, en utilisant l'accès à la chaîne de traitement. L'image ainsi développée est notée $\bar{\mathbf{y}}$. En suivant nos hypothèses de stationnarité et de linéarité, nous pouvons exprimer $\bar{\mathbf{y}}$ comme une fonction de $\bar{\mathbf{x}}$ et de $\mathbf{H}$:

$$\bar{\mathbf{y}}_k = \mathbf{H}\,\bar{\mathbf{x}}_k, \tag{11.5.9}$$

c'est-à-dire que chaque macro-bloc $\bar{\mathbf{y}}_k$ de l'image développée est obtenu en multipliant le macro-bloc correspondant dans le domaine RAW $\bar{\mathbf{x}}_k$ par $\mathbf{H}$.

À partir de l'équation (11.5.9), nous pouvons réduire le problème de l'estimation de $\mathbf{H}$ à un simple problème de régression linéaire. Par exemple, nous pouvons résoudre Eq (11.5.9) pour $\mathbf{H}$ en utilisant la méthode des moindres carrés. Soit m_x le nombre de macro-blocs de taille M^2 dans $\bar{\mathbf{x}}$ et m_y le nombre de macro-blocs correspondants de taille N^2 dans $\bar{\mathbf{y}}$ (sous forme de vecteur). Soit $\bar{\mathbf{X}}$ la matrice $m_x \times M^2$ construite en empilant chaque macro-blocs de $\bar{\mathbf{x}}$ (sous forme vectorielle). De même pour $\bar{\mathbf{Y}}$. La solution des moindres carrés de $\mathbf{H}$ est alors donnée par :

$$\mathbf{H} = \bar{\mathbf{Y}}\bar{\mathbf{X}}^T\left(\bar{\mathbf{X}}\bar{\mathbf{X}}^T\right)^{-1}. \tag{11.5.10}$$

11.5.3 *Estimation de la covariance du bruit de l'image cover*

Nous terminons en présentant une méthode pour estimer la matrice de covariance du bruit dans le domaine développé. La méthode que nous présentons ici est simple et exacte tant que les hypothèses du modèle sont valides. Cependant, elle suppose que l'on a accès à l'image RAW $\mathbf{x}$ ainsi qu'à un accès boîte noire au pipeline de traitement. Pour une méthode approximative mais ne nécessitant pas l'accès à l'image RAW, nous renvoyons à la Section 7.4 de manuscrit.

La première étape consiste à estimer la matrice $\mathbf{H}$ représentant la chaîne de traitement avec la méthode présentée précédemment. Ensuite, nous estimons les paramètres c_1 et c_2 du modèle hétéroscédastique en suivant la méthode décrite dans la Section 5.3 de manuscrit.

Une fois que tous ces paramètres ont été estimés, nous pouvons calculer la variance de chaque photo-site x_i en utilisant :

$$\sigma_i^2 = c_1 \mu_i + c_2. \tag{11.5.11}$$

La valeur moyenne du photo-site μ_i peut être estimée en utilisant n'importe quel algorithme de débruitage dans le domaine RAW tel que [27]. Cependant, nous avons constaté que, dans la pratique, l'utilisation de la valeur réelle du site de la photo comme estimation de la valeur moyenne n'entraîne aucune perte pour la stéganographie.

Nous réécrivons maintenant Eq (11.5.11) comme des macro-blocs de variance :

$$\boldsymbol{\sigma}_k^2 = c_1 \mu_k + c_2. \tag{11.5.12}$$

La covariance Σ_k de chaque macro-bloc est alors simplement obtenue en utilisant:

$$\Sigma_k = \mathbf{H} \mathrm{diag}\left(\boldsymbol{\sigma}_k\right) \mathbf{H}. \tag{11.5.13}$$

Stéganographie utilisant un modèle statistique du bruit de capteur

Cette section décrit la construction d'un schéma d'insertion fondé sur le modèle développé dans la seconde contribution.

Nous commençons par formuler le problème de détection du point de vue du stéganalyste avant d'en construire le détecteur optimal. Une fois cela fait, le but du stéganographe est de trouver le signal minimisant la puissance de ce détecteur.

Nous modélisons la précover $\mathbf{y}$ – l'image cover avant l'opération d'arrondis de la compression JPEG – comme un vecteur contenant m macro-blocs indépendants de taille $M \times M$ suivant une distribution gaussienne multivariée. L'image "pré-stego" $\boldsymbol{\gamma}$ est quant à elle l'image $\mathbf{y}$ à laquelle a été rajouté un signal gaussien multivarié de moyenne nulle et de covariance $\boldsymbol{\epsilon}_k$.

Formellement, soit :

$$\mathbf{y}_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \Sigma_k), \tag{11.5.14}$$

$$\boldsymbol{\gamma}_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \Sigma_k + \mathbf{E}_k). \tag{11.5.15}$$

En suivant la méthodologie présentée dans [69], nous cherchons à trouver les paramètres du signal stégo qui minimisent la puissance du détecteur le plus puissant. Pour ce faire, nous travaillons sous l'hypothèse que le stéganalyste connait $\mathbf{E} = (\mathbf{E}_1, \mathbf{E}_2, \ldots, \mathbf{E}_n)$ ainsi que les paramètres du modèles $\boldsymbol{\mu} = (\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_n)$ et $\boldsymbol{\Sigma} = (\Sigma_1, \Sigma_2, \ldots, \Sigma_n)$ les blocs d'images avant l'opération d'arrondis sont stéganalysés : $\boldsymbol{xi} = (\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_n)$.Le stéganalyste doit alors décider entre deux hypothèses $\forall k \in \{1, 2, \ldots, n\}$:

$$\begin{cases} \mathcal{H}_0 & = \{\boldsymbol{\xi}_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \Sigma_k)\}, \\ \mathcal{H}_1 & = \{\boldsymbol{\xi}_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \Sigma_k + \mathbf{E}_k)\}. \end{cases} \tag{11.5.16}$$

La distribution du bruit sous $\mathcal{H}_0$, $p_{\Sigma_k}(x)$, et celle de $q_{\Sigma_k, \mathbf{E}_k}(x)$ sous $\mathcal{H}_1$ sont données par :

$$p_{\Sigma_k}(x) = \frac{\exp\left((x-\mu_k)^T \Sigma_k^{-1} (x-\mu_k)\right)}{\sqrt{2\pi|\Sigma_k|}} \tag{11.5.17}$$

$$q_{\Sigma_k, \mathbf{E}_k}(x) = \frac{\exp\left((x-\mu_k)^T (\Sigma_k + \mathbf{E}_k)^{-1} (x-\mu_k)\right)}{\sqrt{2\pi|\Sigma_k + \mathbf{E}_k|}}. \tag{11.5.18}$$

Nous utilisons pour la suite le critère d'optimalité de Neyman-Pearson. Dans ce contexte, le stéganalyste construit un test $\delta : \mathbb{R} \to \{\mathcal{H}_0, \mathcal{H}_1\}$ qui maximise la puissance du test $P_D \triangleq \mathbb{P}(\delta(x) = \mathcal{H}_1 | \mathcal{H}_1)$ sous une probabilité de fausse-alarme fixée $P_{FA} \triangleq \mathbb{P}(\delta(x) = \mathcal{H}_1 | \mathcal{H}_0)$.

Sous ces hypothèses, le problème du stéganalyste (11.5.16) est réduit à un choix entre deux hypothèses simples. Dans ce contexte, le lemme de Neyman-Pearson indique que le test le plus puissant existe et que ce test est le test du rapport de vraisemblance (TRV) défini comme :

$$\Lambda_k(\boldsymbol{\xi}, \Sigma_k, \mathbf{E}_k) = \ln\left(\frac{p_{\Sigma_k}(\boldsymbol{\xi})}{q_{\Sigma_k, \mathbf{E}_k}(\boldsymbol{\xi})}\right), \tag{11.5.19}$$

$$\Lambda(\boldsymbol{\xi}, \boldsymbol{\Sigma}, \mathbf{E}) = \sum_{i=0}^{n} \Lambda_i(\boldsymbol{\xi}_k, \Sigma_k, \mathbf{E}_k) \underset{\mathcal{H}_1}{\overset{\mathcal{H}_0}{\lessgtr}} \tau, \tag{11.5.20}$$

où la dernière ligne se déduit de l'indépendance des macro-blocs.

Insertion par minimisation de la puissance du test

L'étude des performances du TRV que nous effectuons dans le Chapitre 10 nous permet de démontrer que le la matrice de covariance qui minimise la divergence de Kullback-Leibler entre la distribution de la précover et celle de la préstego à la forme suivante :

$$\mathbf{E}_k = \alpha \Sigma_k, \tag{11.5.21}$$

avec $\alpha \in \mathbf{R}^+$ indépendant de l'indice k.

Formellement, le stéganographe cherche alors à trouver le signal qui minimiser la divergence de Kullback-Leibler sous contrainte d'insérer un message de taille donnée :

$$\begin{cases} \min\limits_{\mathbf{E}} & \mathrm{D}_{KL}\left(p_{\boldsymbol{\Sigma}} \,||\, q_{\boldsymbol{\Sigma},\mathbf{E}}\right) \\ R = & \sum_{i=1}^{n} \sum_{k \in \mathbb{Z}} \beta_i^k \log \beta_i^k \end{cases} \tag{11.5.22}$$

où $\beta_i^{(j)}$ est la probabilité d'ajouter $+j$ eu i-ième coefficient, n est le nombre total de coefficients DCT et R est la taille du message à insérer en bits. Il est important de noter que l'algorithme minimise la divergence de Kullback-Leibler dans le domaine continue, tandis que la contrainte est donné dans le domaine quantifié.

Pour minimiser P_D sous cette contrainte, une simple recherche par dichotomie du α optimal suffit.

Calcul des probabilités d'insertion

Reste à calculer les probabilités d'insertion $\beta_i^{(j)}$ en fonction de la covariance de la préstego dans le domaine continue. Pour ce faire nous utilisons la factorisation de Cholesky de $\mathbf{E}$:

$$\mathbf{E}_k = \mathbf{L}_k \mathbf{L}_k^T. \tag{11.5.23}$$

Tout d'abord, soit $\mathbf{s}_k$ un macro-bloc de la préstego. Il suit une variable aléatoire (rv) gaussienne multivariée centrée (MVG) de N éléments avec une covariance de rang complet $\mathbf{E}_k$. Dénotons le i-ième élément de $\mathbf{s}_k$ comme $s_{k,i}$. On a alors, pour tout i :

$$s_{k,i} | s_{k,1}, s_{k,2}, ...s_{k,i-1} \sim \mathcal{N}\left(\bar{\eta}_{k,i}, \bar{\epsilon}_{k,i}^2\right). \tag{11.5.24}$$

En d'autres termes, chaque élément d'une gaussienne multivariée conditionné par tous ses éléments précédents suit une distribution gaussienne univariée de moyenne $\bar{\eta}_{k,i}$ et de variance $\bar{\epsilon}_{k,i}^2$. Ensuite, soit $\mathbf{w}_k$ un vecteur de M v.a gaussiennes standard de moyenne nulle et de variance unitaire. Nous pouvons corréler le bruit blanc en le multipliant

par la décomposition de Cholesky d'une matrice de covariance choisie :

$$\mathbf{L}_k \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{E}_k). \tag{11.5.25}$$

Les paramètres $\bar{\boldsymbol{\eta}}$ et $\bar{\boldsymbol{\epsilon}}$ peuvent être calculés efficacement en utilisant la décomposition de Cholesky et la réalisation de la préstegos:

$$\bar{\boldsymbol{\epsilon}}_k = \mathrm{diag}(\mathbf{L}_k), \tag{11.5.26}$$

$$N\bar{\boldsymbol{\eta}}_k = (\mathbf{L}_k \mathbf{w}_k - \mathrm{diag}(\mathbf{L}_k)\mathbf{w}_k). \tag{11.5.27}$$

Si l'on applique Eq (11.5.26)-eq11.5.27) à chaque $\mathbf{s}_k$, on obtient un vecteur $\bar{\mathbf{s}}$ de n éléments tel que :

$$\bar{s}_i \sim \mathcal{N}(\bar{\eta}_i, \bar{\epsilon}_i). \tag{11.5.28}$$

Enfin, en utilisant la formule des probabilités composées sur chaque $\beta_i^{(j)}$, et la quantification de la distribution gaussienne, les probabilités d'insertion $\beta_i^{(j)}$ sont obtenues par :

$$\beta_i^{(j)} = \mathbf{\Phi}\left(\frac{j - r_i - \bar{\eta}_i + 0.5}{\bar{\epsilon}_i}\right) - \mathbf{\Phi}\left(\frac{j - r_i - \bar{\eta}_i - 0.5}{\bar{\epsilon}_i}\right), \tag{11.5.29}$$

où $r_i = y_i - [y_i]$ représente l'erreur d'arrondi du i-ième coefficient DCT.

Résultats

Nous donnons-ici les résultats obtenus avec les algorithmes développés durant cette thèse. La description des conditions expérimentales restera très succinctes dans ce rapport, nous renvoyons à [30] pour plus de détails.

Les performances de nos algorithmes sont mesurées par rapport à l'état de l'art de la stéganographie JPEG – SI-UNIWARD [42]. Nous donnons une rapide description de chaque algorithme dans le Tableau 11.2.

Pour évaluer les performances de ces algorithmes, nous utilisons la base d'image BOSS [5], standard en stéganographie. La base d'image a été développée avec deux chaine de développement (CD) différentes – voir le tableau ci-dessous. La première CD est linéaire et correspond aux hypothèse de notre modèle, tandis que la CD BOSS correspond à la CD utilisée dans les benchmarks standard en stéganographie. Les performances sont évaluées empiriquement en utilisant l'état de l'art de la stéganalyse actuelle – Efficient-Net [76]. Efficient-Net-b3 est entrainé sur 5000 images et testé sur les 2342 images restantes. Le score utilisé est la probabilité d'erreur sous hypothèse de classes cover/stégo équilibrées :

$$P_E = \min_{P_{FA}} \frac{P_{FA} + P_{MD}}{2} \tag{11.5.30}$$

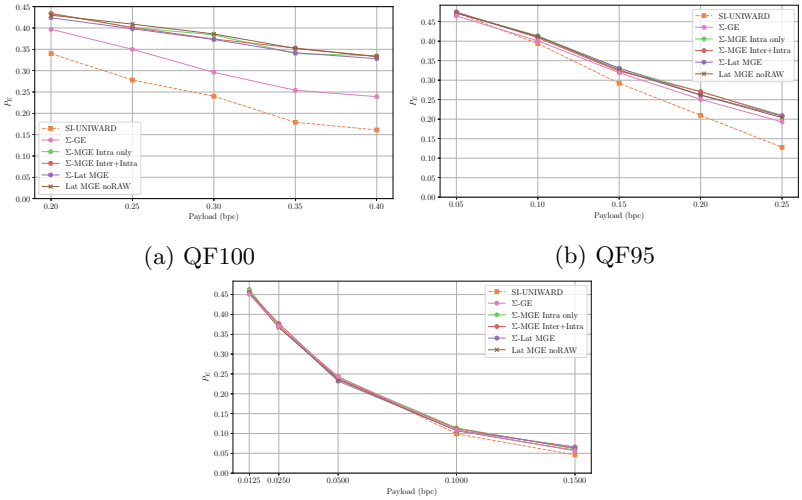
où P_{FA} est le taux de fausse alarme du détecteur sur la base de test, P_{MD} le taux de mauvaise détection.

Les résultats sont présentés dans les Figures 11.8-11.9.

CD	Dématriçage	Balance des blancs	Sous-échantillonage
Linéaire	Bilinéaire	Non	Rognage, 264×264
BOSS	PPG	Oui	Redimensionnements 792×792 à 264×264

Name	Meaning
GE	Minimise la puissance du détecteur le plus puissant en supposant que les coefficients DCT sont indépendants – voir le Chapitre 9.
MGE Intra Only	Minimise la divergence KL en supposant que les macro-blocs DCT 8×8 sont indépendants.
MGE Intra+Inter	Minimise la divergence KL en supposant que les macro-blocs DCT 24×24 sont indépendants.
Lat MGE	Minimise la divergence KL sous les hypothèses du modèle par lattice tel que présenté dans la Section 10.3.3.
SI-UNIWARD	Schéma d'insertion employant l'information adjacente tel que décrit dans [42].

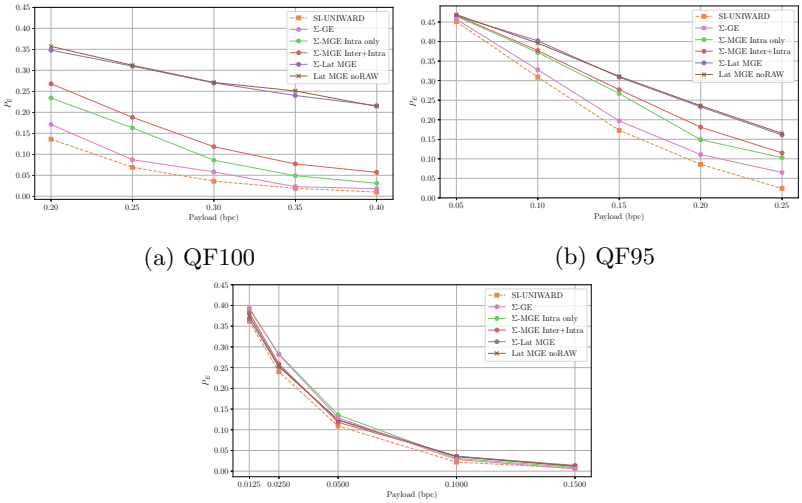
Table 11.2: Nomenclature of the embedding schemes



(a) QF100 (b) QF95

(c) QF75

Figure 11.8: P_E en fonction de la taille du message inséré sur la base d'image BOSS développée avec la CD BOSS et stéganalysée avec EfficientNet-b3.



(a) QF100 (b) QF95

(c) QF75

Figure 11.9: P_E en fonction de la taille du message inséré sur la base d'image BOSS développée avec la CD linéaire et stéganalysée avec EfficientNet-b3.

Les résultats des expériences sont présentés dans la Figure 11.1 pour la CD Bosslike et dans la Figure 11.2 pour la CD linéaire. À partir de QF100, quel que soit la CD de traitement, il existe un écart net entre les performances de SI-UNIWARD et les différentes variantes de MGE. Pour la CD linéaire, l'écart est, en moyenne, de $6\% et 9\%$ en termes de P_E absolu pour Σ-MGE Intra seulement et Σ-MGE intra + inter respectivement. Cependant, il est de 22% en moyenne lors de l'utilisation de Σ-Lat MGE montrant l'importance d'utiliser le modèle le plus précis pour ce pipeline. La différence entre les différents modèles est moins prononcée pour la CD Bosslike avec un gain moyen de 13.5% en termes de P_E absolu par rapport à SI-UNIWARD pour les schémas MGE quel que soit le modèle de corrélation choisi. Cela est dû au fait que l'opération de sous-échantillonnage supprime la plupart des corrélations entre les blocs en raison de la suppression des pixels voisins avant la transformée DCT.

À QF95, la différence entre les différents schémas devient moins prononcée pour la CD linéaire. Cependant, il y a toujours un écart de performance entre ces schémas et SI-UNIWARD. Par rapport à Σ-Lat MGE, il y a un gain moyen de $10,5\%$ par rapport à SI-UNIWARD pour la CD linéaire et de 4% pour la CD BOSS. À QF75, tous les schémas ont à peu près les mêmes performances, quel que soit la CD. Cela est probablement dû au fait qu'à un QF aussi bas, la plupart des variances sont maintenant proches de 0. Ainsi, la plupart des performances de la stéganographie sont probablement dues à l'information adjacente liée aux erreurs d'arrondi.

Nous notons également que l'implémentation de Σ-Lat MGE qui n'utilise pas le fichier RAW a une performance pratiquement identique à l'implémentation originale, ce qui montre que les hypothèses sur la CD que nous avons utilisées dans la section 7.4 sont tout à fait pratiques dans un contexte de stéganographie.

Enfin, notez que lorsque vous utilisez Σ-GE, qui n'utilise aucune corrélation entre les coefficients DCT, il y a toujours un petit gain par rapport à SI-UNIWARD pour QF100 et QF95. Cependant, ses performances sont toujours inférieures, même par rapport à Σ-MGE intra seulement, ce qui montre l'importance de prendre en compte ces corrélations.

Une chose intéressante à noter ici est l'impact du pipeline de traitement sur la structure de corrélation nécessaire pour obtenir de bonnes performances. Dans le cas de Bosslike, l'opération de sous-échantillonnage a rendu la plupart des dépendances inter-blocs très petites par rapport aux dépendances intra-blocs – voir Figure 11.3. Par conséquent, l'utilisation d'un modèle plus sophistiqué des dépendances n'apporte aucun gain de performance. Par contre, dans le cas du pipeline linéaire où toutes les dépendances sont préservées jusqu'à la fin du pipeline puisqu'aucun sous-échantillonnage n'est effectué, il y a un gain assez substantiel en utilisant le modèle le plus sophistiqué.

Impact du facteur de qualité sur la matrice de covariance

Comme nous venons de l'observer, plus nous utilisons des facteurs de qualité faibles pour compresser nos images, plus le gain en sécurité est faible lorsqu'on utilise des modèles tenant compte des dépendances entre les coefficients DCT. Cette tendance est si forte qu'à QF75, pour une CD donnée, tous les algorithmes testés ont exactement les mêmes performances malgré des modèles très différents. Il est malgré tout important de noter qu'en général, le stéganographe devrait éviter les couvertures avec des facteurs de qualité faibles car une quantification plus forte entraîne une variance plus faible du coefficient DCT et par conséquent, *mutatis mutandi*, une sécurité plus faible [9].

Pour expliquer ce phénomène, nous étudions l'impact de la quantification de la matrice de covariance.

Pour illustrer notre analyse, nous fournissons dans la Figure 11.10 une série de matrices de covariance du même bloc d'image à mesure que le facteur de qualité diminue. Notez que ces covariances sont obtenues **après** l'arrondi des coefficients DCT. On peut observer que la plupart des covariances deviennent négligeables dès que la quantification avec QF95 est effectuée. De plus, la plupart des variances et des covariances deviennent nulles car elles sont bien plus petites que le pas de quantification après avoir été divisées par la matrice de quantification.

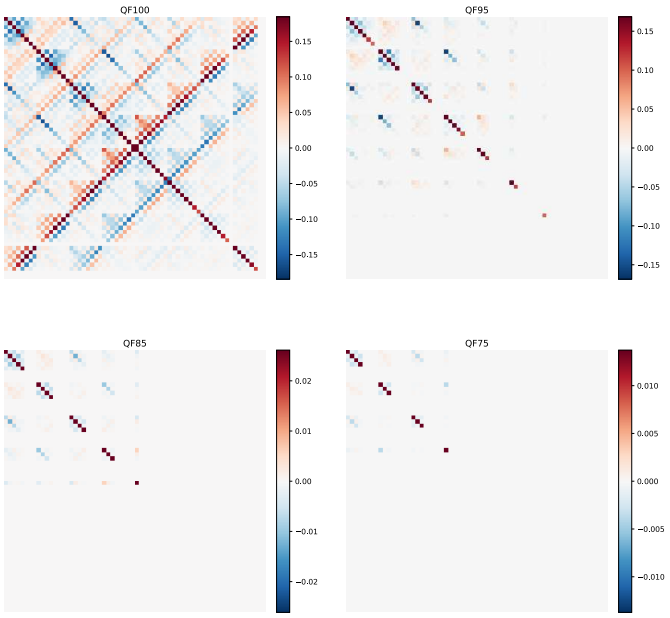


Figure 11.10: Matrice de covariance d'un bloc de coefficient DCT arrondi de l'image *1200* dans BOSSBase avec le pipeline linéaire pour différents facteurs de qualité.

Bibliography

[1] N. Ahmed, T. Natarajan, and K. Rao. *Discrete Cosine Transform*. In: IEEE Transactions on Computers (Jan. 1974) (see p. 79).

[2] N. Ahmed. *How I came up with the discrete cosine transform*. In: Digital Signal Processing (Jan. 1991) (see p. 79).

[3] L. Azzari, L. R. Borges, and A. Foi. *Modeling and Estimation of Signal-Dependent and Correlated Noise*. In: Denoising of Photographic Images and Video. Ed. by M. Bertalmío. Series Title: Advances in Computer Vision and Pattern Recognition. Springer International Publishing, 2018 (see pp. 65, 103).

[4] L. Azzari and A. Foi. *Gaussian-Cauchy mixture modeling for robust signal-dependent noise estimation*. In: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). ICASSP 2014 - 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, May 2014 (see p. 92).

[5] P. Bas, T. Filler, and T. Pevný. *"Break Our Steganographic System": The Ins and Outs of Organizing BOSS*. In: Information Hiding. Ed. by T. Filler et al. Series Title: Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011 (see pp. 47, 74, 108, 168).

[6] S. Bernard et al. *Explicit Optimization of min max Steganographic Game*. In: IEEE Transactions on Information Forensics and Security (2021) (see pp. 35, 154, 155).

[7] R. Böhme and R. Böhme. *Advanced statistical steganalysis*. Information security and cryptography. Springer-Verlag, 2010 (see p. 19).

[8] M. Boroumand, M. Chen, and J. Fridrich. *Deep Residual Network for Steganalysis of Digital Images*. In: IEEE Transactions on Information Forensics and Security (May 2019) (see pp. 35, 154, 156).

[9] J. Butora and J. Fridrich. *Effect of JPEG Quality on Steganographic Security*. In: Proceedings of the ACM Workshop on Information Hiding and Multimedia Security. IH&MMSec '19: ACM Information Hiding and Multimedia Security Workshop. ACM, July 2, 2019 (see p. 171).

[10] J. Butora and J. Fridrich. *Minimum Perturbation Cost Modulation for Side-Informed Steganography*. In: Electronic Imaging (Jan. 26, 2020) (see pp. 31, 32).

[11] C. Cachin. *An Information-Theoretic Model for Steganography*. In: Information Hiding. Ed. by D. Aucsmith. Series Title: Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1998 (see pp. 24, 29, 136).

[12] R. Cogranne, Q. Giboulot, and P. Bas. *ALASKA#2: Challenging Academic Research on Steganalysis with Realistic Images*. In: 2020 IEEE International Workshop on Information Forensics and Security (WIFS). 2020 IEEE International Workshop on Information Forensics and Security (WIFS). IEEE, Dec. 6, 2020 (see pp. 36, 93, 108, 142, 156).

[13] R. Cogranne et al. *Is ensemble classifier needed for steganalysis in high-dimensional feature spaces?* In: 2015 IEEE International Workshop on Information Forensics and Security (WIFS). 2015 IEEE International Workshop on Information Forensics and Security (WIFS). IEEE, Nov. 2015 (see p. 41).

[14] R. Cogranne, Q. Giboulot, and P. Bas. *Efficient Steganography in JPEG Images by Minimizing Performance of Optimal Detector*. In: IEEE Transactions on Information Forensics and Security (2021) (see pp. 31, 101, 106, 107).

[15] R. Cogranne, Q. Giboulot, and P. Bas. *Steganography by Minimizing Statistical Detectability: The cases of JPEG and Color Images*. In: Proceedings of the 2020 ACM Workshop on Information Hiding and Multimedia Security. IH&MMSec '20: ACM Workshop on Information Hiding and Multimedia Security. ACM, June 22, 2020 (see pp. 103, 106).

[16] R. Cogranne and F. Retraint. *An Asymptotically Uniformly Most Powerful Test for LSB Matching Detection*. In: IEEE Transactions on Information Forensics and Security (Mar. 2013) (see p. 103).

[17] T. M. Cover and J. A. Thomas. *Elements of information theory*. 2nd ed. OCLC: ocm59879802. Wiley-Interscience, 2006. 748 pp. (see pp. 22, 113).

[18] T. Denemark and J. Fridrich. *Side-informed steganography with additive distortion*. In: 2015 IEEE International Workshop on Information Forensics and Security (WIFS). 2015 IEEE International Workshop on Information Forensics and Security (WIFS). IEEE, Nov. 2015 (see p. 31).

[19] T. Denemark and J. Fridrich. *Steganography With Multiple JPEG Images of the Same Scene*. In: IEEE Transactions on Information Forensics and Security (Oct. 2017) (see p. 30).

[20] T. Denemark, P. Bas, and J. Fridrich. *Natural Steganography in JPEG Compressed Images*. In: Electronic Imaging (Jan. 28, 2018) (see pp. 47, 141).

[21] T. Denemark and J. Fridrich. *Model Based Steganography with Precover*. In: Electronic Imaging (Jan. 29, 2017) (see p. 32).

[22] J. Deng et al. *ImageNet: A large-scale hierarchical image database*. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops). IEEE, June 2009 (see p. 47).

[23] R. Durrett. *Probability: theory and examples*. 4th ed. Cambridge series in statistical and probabilistic mathematics. OCLC: ocn607573997. Cambridge University Press, 2010. 428 pp. (see pp. 28, 64).

[24] T. Filler, J. Judas, and J. Fridrich. *Minimizing Additive Distortion in Steganography Using Syndrome-Trellis Codes*. In: IEEE Transactions on Information Forensics and Security (Sept. 2011) (see p. 23).

[25] T. Filler and J. Fridrich. *Gibbs Construction in Steganography*. In: IEEE Transactions on Information Forensics and Security (Dec. 2010) (see pp. 19, 22).

[26] A. Foi et al. *Practical Poissonian-Gaussian Noise Modeling and Fitting for Single-Image Raw-Data*. In: IEEE Transactions on Image Processing (Oct. 2008) (see pp. 59, 65, 66, 69, 71).

[27] A. Foi. *Clipped noisy images: Heteroskedastic modeling and practical denoising*. In: Signal Processing (Dec. 2009) (see pp. 59, 69, 88, 92, 155, 158, 163, 165).

[28] J. Fridrich. *Steganography in digital media: principles, algorithms, and applications*. OCLC: 1144198829. 2010 (see p. 17).

[29] J. Fridrich and J. Kodovsky. *Multivariate gaussian model for designing additive distortion for steganography*. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP 2013 - 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, May 2013 (see pp. 24, 101, 103, 154).

[30] Q. Giboulot, P. Bas, and R. Cogranne. *Multivariate Side-Informed Gaussian Embedding Minimizing Statistical Detectability*. In: IEEE Transactions on Information Forensics and Security (submitted) (2022) (see pp. 101, 168).

[31] Q. Giboulot, P. Bas, and R. Cogranne. *Synchronization Minimizing Statistical Detectability for Side-Informed JPEG Steganography*. In: 2020 IEEE International Workshop on Information Forensics and Security (WIFS). 2020 IEEE International Workshop on Information Forensics and Security (WIFS). IEEE, Dec. 6, 2020 (see pp. 101, 155).

[32] Q. Giboulot, R. Cogranne, and P. Bas. *Detectability-Based JPEG Steganography Modeling the Processing Pipeline: The Noise-Content Trade-off*. In: IEEE Transactions on Information Forensics and Security (2021) (see pp. 59, 101, 106, 119, 142, 155).

[33] Q. Giboulot, R. Cogranne, and P. Bas. *JPEG Steganography with Side Information from the Processing Pipeline*. In: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, May 2020 (see pp. 59, 101).

[34] Q. Giboulot, R. Cogranne, and P. Bas. *Steganalysis into the Wild: How to Define a Source?* In: Electronic Imaging (Jan. 28, 2018) (see pp. 49, 159).

[35] Q. Giboulot et al. *Effects and solutions of Cover-Source Mismatch in image steganalysis*. In: Signal Processing: Image Communication (Aug. 2020) (see pp. 49, 54, 56, 154, 159).

[36] G. Goh. *Why Momentum Really Works.* In: Distill (Apr. 4, 2017) (see p. 45).

[37] M. Goljan, J. Fridrich, and T. Holotyak. *New blind steganalysis and its implications*. In: Electronic Imaging 2006. Ed. by E. J. Delp III and P. W. Wong. Feb. 2, 2006 (see pp. 48, 159).

[38] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. Adaptive computation and machine learning. The MIT Press, 2016. 775 pp. (see p. 41).

[39] L. Guo et al. *Using Statistical Image Model for JPEG Steganography: Uniform Embedding Revisited*. In: IEEE Transactions on Information Forensics and Security (Dec. 2015) (see pp. 20, 108, 154).

[40] K. He et al. *Deep Residual Learning for Image Recognition*. In: arXiv:1512.03385 [cs] (Dec. 10, 2015). arXiv:1512.03385 (see pp. 36, 156).

[41] V. Holub and J. Fridrich. *Designing steganographic distortion using directional filters*. In: 2012 IEEE International Workshop on Information Forensics and Security (WIFS). 2012 IEEE International Workshop on Information Forensics and Security (WIFS). IEEE, Dec. 2012 (see p. 20).

[42] V. Holub, J. Fridrich, and T. Denemark. *Universal distortion function for steganography in an arbitrary domain*. In: EURASIP Journal on Information Security (Dec. 2014) (see pp. 20, 31, 108, 144, 154, 168, 169).

[43] R. A. Horn and C. R. Johnson. *Matrix analysis*. Second edition, corrected reprint. Cambridge University Press, 2017. 643 pp. (see p. 126).

[44] S. Ioffe and C. Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. In: arXiv:1502.03167 [cs] (Mar. 2, 2015). arXiv:1502.03167 (see pp. 35, 156).

[45] M. G. Kendall. *The Conditions under which Sheppard's Corrections are Valid*. In: Journal of the Royal Statistical Society (1938) (see p. 104).

[46] C. Kin-Cleaves and A. D. Ker. *Reducing coding loss with irregular syndrome trellis codes*. In: Electronic Imaging (Jan. 13, 2019) (see p. 24).

[47] C. Kin-Cleaves and A. D. Ker. *Simulating Suboptimal Steganographic Embedding*. In: Proceedings of the 2020 ACM Workshop on Information Hiding and Multimedia Security. IH&MMSec '20: ACM Workshop on Information Hiding and Multimedia Security. ACM, June 22, 2020 (see p. 24).

[48] D. P. Kingma and J. Ba. *Adam: A Method for Stochastic Optimization*. In: arXiv:1412.6980 [cs] (Jan. 29, 2017). arXiv:1412.6980 (see p. 45).

[49] J. Kodovsky, J. Fridrich, and V. Holub. *Ensemble Classifiers for Steganalysis of Digital Media*. In: IEEE Transactions on Information Forensics and Security (Apr. 2012) (see p. 41).

[50] J. Kodovský, V. Sedighi, and J. Fridrich. *Study of cover source mismatch in steganalysis and ways to mitigate its impact*. In: IS&T/SPIE Electronic Imaging. Ed. by A. M. Alattar, N. D. Memon, and C. D. Heitzenrater. Feb. 19, 2014 (see pp. 48, 159).

[51] T. Kuroda. *Essential principles of image sensors*. CRC Press, 2015. 174 pp. (see pp. 62, 63, 163).

[52] L. Le Cam. *An approximation theorem for the Poisson binomial distribution*. In: Pacific Journal of Mathematics (Dec. 1, 1960) (see p. 62).

[53] M. Lebrun et al. *Secrets of image denoising cuisine*. In: Acta Numerica (May 2012) (see p. 65).

[54] Y. LeCun et al. *Backpropagation Applied to Handwritten Zip Code Recognition*. In: Neural Computation (Dec. 1989) (see p. 45).

[55] E. L. Lehmann and G. Casella. *Theory of point estimation*. 2nd ed. Springer texts in statistics. Springer, 1998. 589 pp. (see p. 107).

[56] B. Li et al. *A new cost function for spatial image steganography*. In: 2014 IEEE International Conference on Image Processing (ICIP). 2014 IEEE International Conference on Image Processing (ICIP). IEEE, Oct. 2014 (see pp. 20, 154).

[57] Y. Makinen, L. Azzari, and A. Foi. *Collaborative Filtering of Correlated Noise: Exact Transform-Domain Variance for Improved Shrinkage and Patch Matching*. In: IEEE Transactions on Image Processing (2020) (see p. 103).

[58] P. G. Moschopoulos. *The distribution of the sum of independent gamma random variables*. In: Annals of the Institute of Statistical Mathematics (Dec. 1985) (see p. 124).

[59] V. Nair and G. E. Hinton. *Rectified Linear Units Improve Restricted Boltzmann Machines*. In: () (see pp. 35, 156).

[60] J. Nakamura, ed. *Image sensors and signal processing for digital still cameras*. Taylor & Francis, 2006. 336 pp. (see pp. 62, 63, 163).

[61] J. Neyman and E. Pearson. *IX. On the problem of the most efficient tests of statistical hypotheses*. In: Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character (Feb. 16, 1933) (see p. 27).

[62] T. Nguyen and S. Oraintara. *The Shiftable Complex Directional Pyramid—Part II: Implementation and Applications*. In: IEEE Transactions on Signal Processing (Oct. 2008) (see p. 54).

[63] H. Nyquist. *Thermal Agitation of Electric Charge in Conductors*. In: Physical Review (July 1, 1928) (see p. 63).

[64] *Parameter values for the HDTV standards for production and international programme exchange*. June 2015 (see p. 76).

[65] T. Pevný, P. Bas, and J. Fridrich. *Steganalysis by Subtractive Pixel Adjacency Matrix*. In: (2010) (see pp. 35, 155).

[66] T. Pevný, T. Filler, and P. Bas. *Using High-Dimensional Image Models to Perform Highly Undetectable Steganography*. In: Information Hiding. Ed. by R. Böhme, P. W. L. Fong, and R. Safavi-Naini. Series Title: Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010 (see p. 20).

[67] A. Polesel, G. Ramponi, and V. Mathews. *Image enhancement via adaptive unsharp masking*. In: IEEE Transactions on Image Processing (Mar. 2000) (see p. 53).

[68] Y. Qian et al. *Deep learning for steganalysis via convolutional neural networks*. In: IS&T/SPIE Electronic Imaging. Ed. by A. M. Alattar, N. D. Memon, and C. D. Heitzenrater. Mar. 4, 2015 (see pp. 35, 156).

[69] V. Sedighi, R. Cogranne, and J. Fridrich. *Content-Adaptive Steganography by Minimizing Statistical Detectability*. In: IEEE Transactions on Information Forensics and Security (Feb. 2016) (see pp. 24, 33, 51, 101, 103, 106, 111, 154, 166).

[70] V. Sedighi, J. Fridrich, and R. Cogranne. *Content-adaptive pentary steganography using the multivariate generalized Gaussian cover model*. In: IS&T/SPIE Electronic Imaging. Ed. by A. M. Alattar, N. D. Memon, and C. D. Heitzenrater. Mar. 4, 2015 (see pp. 106, 107).

[71] V. Sedighi, J. Fridrich, and R. Cogranne. *Toss that BOSSbase, Alice!* In: Electronic Imaging (Feb. 14, 2016) (see pp. 24, 48).

[72] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: from theory to algorithms*. Cambridge University Press, 2014. 397 pp. (see pp. 16, 38, 39).

[73] G. J. Simmons. *The Prisoners' Problem and the Subliminal Channel*. In: Advances in Cryptology. Ed. by D. Chaum. Springer US, 1984 (see p. 11).

[74] X. Song et al. *Steganalysis of Adaptive JPEG Steganography Using 2D Gabor Filters*. In: Proceedings of the 3rd ACM Workshop on Information Hiding and Multimedia Security. IH&MMSec '15: ACM Information Hiding and Multimedia Security Workshop. ACM, June 17, 2015 (see p. 37).

[75] T. Taburet et al. *Natural Steganography in JPEG Domain With a Linear Development Pipeline*. In: IEEE Transactions on Information Forensics and Security (2021) (see pp. 19, 81, 95, 117, 129, 155).

[76] M. Tan and Q. V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. In: arXiv:1905.11946 [cs, stat] (Sept. 11, 2020). arXiv:1905.11946 (see pp. 36, 50, 108, 156, 168).

[77] W. Tang et al. *CNN-Based Adversarial Embedding for Image Steganography*. In: IEEE Transactions on Information Forensics and Security (Aug. 2019) (see pp. 142, 154).

[78] T. H. Thai, F. Retraint, and R. Cogranne. *Statistical detection of data hidden in least significant bits of clipped images*. In: Signal Processing (May 2014) (see p. 92).

[79] Thanh Hai Thai, R. Cogranne, and F. Retraint. *Camera Model Identification Based on the Heteroscedastic Noise Model*. In: IEEE Transactions on Image Processing (Jan. 2014) (see pp. 66, 68).

[80] H. L. Van Trees, K. L. Bell, and Z. Tian. *Detection estimation and modulation theory*. Second edition. John Wiley & Sons, Inc, 2013 (see pp. 124, 135).

[81] R. F. Voss. *Linearity of 1 f Noise Mechanisms*. In: Physical Review Letters (Apr. 3, 1978) (see p. 64).

[82] B. Widrow and I. Kollar. *Quantization noise: roundoff error in digital computation, signal processing, control, and communications*. OCLC: ocn183916250. Cambridge University Press, 2008. 751 pp. (see pp. 104, 155).

[83] G. Xu. *Deep Convolutional Neural Network to Detect J-UNIWARD*. In: Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security. IH&MMSec '17: ACM Information Hiding and Multimedia Security Workshop. ACM, June 20, 2017 (see pp. 35, 156).

[84] G. Xu, H.-Z. Wu, and Y.-Q. Shi. *Structural Design of Convolutional Neural Networks for Steganalysis*. In: IEEE Signal Processing Letters (May 2016) (see pp. 35, 156).

[85] Y. Yousfi, J. Butora, and J. Fridrich. *CNN Steganalyzers Leverage Local Embedding Artifacts*. In: 2021 IEEE International Workshop on Information Forensics and Security (WIFS). 2021 IEEE International Workshop on Information Forensics and Security (WIFS). IEEE, Dec. 7, 2021 (see p. 150).

[86] Y. Yousfi et al. *Improving EfficientNet for JPEG Steganalysis*. In: Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security. IH&MMSec '21: ACM Workshop on Information Hiding and Multimedia Security. ACM, June 17, 2021 (see pp. 36, 157).

-

English abstract Steganography is the discipline concerned with techniques designed to embed hidden data into an innocuous cover media. In the case of this manuscript, the cover media of choice are JPEG images. Steganography schemes based on a statistical model of natural images possess a clear advantage against schemes based on heuristics. Indeed, they provide a direct link between theoretical detectability and empirical performance. However, this advantage is dependent on the accuracy of the underlying cover and the stego model. Until the work presented in this manuscript, the available models were not accurate enough for statistical steganography schemes to attain competitive performances in the JPEG domain or to provide security guarantees for natural images. In this manuscript, we propose two main contributions to solve this problem. First, we derive a model of noise in the developed domain which takes into account the camera sensor, ISO setting and the full processing pipeline. This leads to a multivariate Gaussian model of the noise which models intra and inter-block dependencies in JPEG images. Secondly, we design a series of steganographic algorithms leveraging this noise model. They minimize or bound the power of the most powerful detector to provide security guarantees when meeting the model assumptions. In particular, we show that the optimal covariance of the stego signal is proportional to the covariance of the cover noise. Finally, these algorithms are shown to attain state-of-the-art performance, greatly outperforming the standard algorithms in side-informed JPEG steganography.
Keywords: Cryptography, Statistical hypothesis testing,Signal processing, Parameter estimation.

-

Abstract français La stéganographie est la discipline traitant des techniques visant à dissimuler de l'information dans un média de couverture jugé inoffensif. Dans le cadre de ce manuscrit, les médias de couvertures choisis sont des images JPEG. Les schémas stéganographiques basés sur un modèle statistique d'images naturelles présentent un avantage certain par rapport aux schémas basés sur des heuristiques. En effet, ils fournissent un lien direct entre détectabilité théorique et performances empiriques. Cependant, cet avantage dépend de la précision des modèles sous-jacents. Cette précision était insuffisante dans les travaux précédants ce manuscrit. Nous proposons deux contributions principales pour résoudre ce problème. Premièrement, nous dérivons un modèle de bruit prenant en compte le capteur, le réglage ISO et la chaîne de traitement. Cela conduit à un modèle gaussien multivarié du bruit modélisant les dépendances intra et inter-blocs dans les images JPEG. Ensuite, nous concevons une série de schémas stéganographiques exploitant ce modèle de bruit. Ils minimisent ou bornent la puissance du détecteur le plus puissant pour fournir des garanties de sécurité lorque les hypothèses du modèle sont respectées. En particulier, nous montrons que la covariance optimale du signal stégo est proportionnelle à la covariance du bruit de l'image cover. Enfin, nous montrons que ces algorithmes atteignent des performances à l'état de l'art, dépassant largement les algorithmes standard de la stéganographie JPEG.

Mots clés : Cryptographie, Tests d'hypothèses (statistique), Traitement du signal, Estimation de paramètres.

Quentin GIBOULOT

Doctorat : Optimisation et Sûreté des Systèmes

Année 2022

Stéganographie statistique fondée sur un modèle de bruit utilisant la chaine de développement d'image

La stéganographie est la discipline traitant des techniques visant à dissimuler de l'information dans un média de couverture jugé inoffensif. Dans le cadre de ce manuscrit, les médias de couvertures choisis sont des images JPEG. Les schémas stéganographiques basés sur un modèle statistique d'images naturelles présentent un avantage certain par rapport aux schémas basés sur des heuristiques. En effet, ils fournissent un lien direct entre détectabilité théorique et performances empiriques. Cependant, cet avantage dépend de la précision des modèles sous-jacents. Cette précision était insuffisante dans les travaux précédants ce manuscrit. Nous proposons deux contributions principales pour résoudre ce problème. Premièrement, nous dérivons un modèle de bruit prenant en compte le capteur, le réglage ISO et la chaîne de traitement. Cela conduit à un modèle gaussien multivarié du bruit modélisant les dépendances intra et inter-blocs dans les images JPEG. Ensuite, nous concevons une série de schémas stéganographiques exploitant ce modèle de bruit. Ils minimisent ou bornent la puissance du détecteur le plus puissant pour fournir des garanties de sécurité lorsque les hypothèses du modèle sont respectées. En particulier, nous montrons que la covariance optimale du signal stégo est proportionnelle à la covariance du bruit de l'image cover. Enfin, nous montrons que ces algorithmes atteignent des performances à l'état de l'art, dépassant largement les algorithmes standard de la stéganographie JPEG.

Mots clés : cryptographie – tests d'hypothèses (statistique) – traitement du signal – estimation de paramètres.

Statistical Steganography based on a Sensor Noise Model using the Processing Pipeline

Steganography is the discipline concerned with techniques designed to embed hidden data into an innocuous cover media. In the case of this manuscript, the cover media of choice are JPEG images. Steganography schemes based on a statistical model of natural images possess a clear advantage against schemes based on heuristics. Indeed, they provide a direct link between theoretical detectability and empirical performance. However, this advantage is dependent on the accuracy of the underlying cover and the stego model. Until the work presented in this manuscript, the available models were not accurate enough for statistical steganography schemes to attain competitive performances in the JPEG domain or to provide security guarantees for natural images. In this manuscript, we propose two main contributions to solve this problem. First, we derive a model of noise in the developed domain which takes into account the camera sensor, ISO setting and the full processing pipeline. This leads to a multivariate Gaussian model of the noise which models intra and inter-block dependencies in JPEG images. Secondly, we design a series of steganographic algorithms leveraging this noise model. They minimize or bound the power of the most powerful detector to provide security guarantees when meeting the model assumptions. In particular, we show that the optimal covariance of the stego signal is proportional to the covariance of the cover noise. Finally, these algorithms are shown to attain state-of-the-art performance, greatly outperforming the standard algorithms in side-informed JPEG steganography.

Keywords: cryptography – statistical hypothesis testing – signal processing – parameter estimation.

Thèse réalisée en partenariat entre :

Ecole Doctorale "Sciences pour l'Ingénieur"