6

STÉGANOGRAPHIE : insertion d'informations dans des contenus multimédia

Patrick BAS[1], Rémi COGRANNE[2] et Marc CHAUMONT[3]
[1]CRIStAL, Lille, CNRS
[2]Univ. Technologique de Troyes, Troyes
[3]LIRMM, Université Montpellier, CNRS

Ce chapitre présente les notions de base en stéganographie. Ces notions utilisent le plus souvent les images numériques comme support mais nombre d'entre elles peuvent être appliquées à d'autres contenus provenant de capteurs comme les sons ou les vidéos. Après un rappel des fondements théoriques liés à la stéganographie, nous présentons les principes fondamentaux de ce domaine pour ensuite détailler des méthodes de bases, utilisant l'image dans son format spatial[1] ou JPEG[2].

La deuxième partie de ce chapitre présente des principes plus avancés en stéganographie, ces notions permettant soit d'augmenter la sécurité du schéma de base, soit de prendre en compte d'autres contextes pratiques (stéganographie d'un groupe d'images, stéganographie d'images couleurs, utilisation d'une image haute résolution lors de l'insertion, ...).

Sécurité multimédia - Partie 1 : authentification et insertion de données cachées, coordonné par William PUECH.

1. c'est à dire où chaque pixel est codé par un niveau de gris ou par 3 canaux couleurs.

2. l'image est ici codée comme un ensemble de coefficients DCT, voir `https://fr.wikipedia.org/wiki/JPEG.b`

6.1. Introduction et fondements théoriques

La stéganographie cherche à modifier le contenu d'un document (appelé document de couverture ou plus communément document « **Cover** ») afin d'insérer un message qui soit **indétectable** en produisant un document stéganographié appelé « **Stégo** ». Le contexte pratique de l'utilisation de la stéganographie, présenté dans (Simmons 1984) comme « le problème du prisonnier », est le suivant : Alice cherche à transmettre une information sensible à Bob dans un canal de communication anodin (envoi d'une photo de vacances attachée à un courrier électronique par exemple). L'adversaire, généralement appelée Ève[3], est capable d'observer le canal de communication entre Alice et Bob, et cherche à détecter l'utilisation de méthodes de stéganographie en utilisant des méthodes de stéganalyse[4].

L'essor de la stéganographie et de la stéganalyse est arrivé après le 11 Septembre 2001, lorsque des journaux américains ont mentionnés que le groupe Al-Qaïda pouvait utiliser des méthodes de stéganographie pour communiquer au sein de son réseau. L'histoire a également montré que la stéganographie a été notamment utilisée pour diverses activités sensibles, le plus souvent malveillantes telles que :

– les réseaux terroristes[5],

– les réseaux pédophiles pour cacher des images dans des images[6],

– les réseaux de « botnets » pour envoyer des commandes aux ordinateurs esclaves sans qu'elles puissent être bloquées par des pare-feux (Pevnỳ *et al*. 2016).

Plus généralement, la stéganographie peut être utilisée comme une solution de communication qui ne lève aucun soupçon.

Dans le domaine académique, la stéganographie et la stéganalyse ont également profité de l'engouement porté au début du 21^{ieme} siècle pour les méthodes permettant de cacher de l'information (le « data-hiding » en anglais). De telles méthodes peuvent également servir à répondre à des problèmes de droits d'auteurs sur les contenus numériques en liant le message inséré au propriétaire de l'image. Depuis maintenant plus de 20 ans, la stéganographie et la stéganalyse sont des disciplines bien établies au sein du champ plus large de la sécurité de l'information.

Dans ce chapitre nous exposons les concepts clés de la stéganographie en prenant comme média les images numériques, cependant un grand nombre des notions présentées peuvent être utilisées pour d'autres media acquis par des capteurs, tels que

3. pour « eavesdropper » en anglais, la personne qui écoute aux portes.

4. ces méthodes sont décrites dans le prochain chapitre.

5. voir `https://www.unodc.org/documents/frontpage/Use_of_Internet_for_Terrorist_Purposes.pdf`

6. voir `http://news.bbc.co.uk/2/hi/science/nature/2082657.stm`

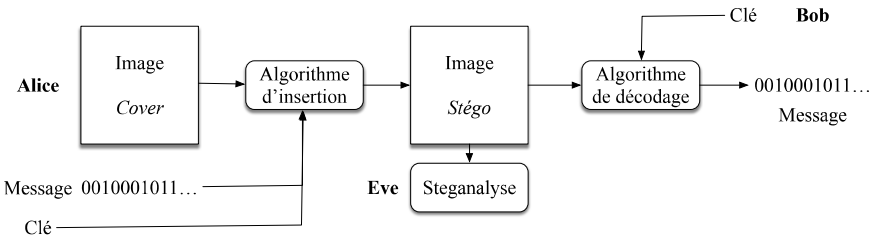


Figure 6.1 – Fonctionnement général de la stéganographie et rôles des différents acteurs (Alice, Bob et Ève).

le son ou les vidéos. Les différentes briques constituantes de la stéganographie sont illustrées sur la figure 6.1.

De manière générale, un algorithme de stéganographie doit faire face à deux contraintes :

– contrainte 1 : l'insertion du message doit être **indétectable**, c'est à dire qu'Ève ne doit pas pouvoir déceler ni l'utilisation de l'algorithme, ni l'insertion d'un message,

– contrainte 2 : l'algorithme doit également **maximiser la capacité d'insertion**, c'est à dire la taille du message inséré.

Dans le reste de ce chapitre nous allons focaliser notre attention sur la stéganographie d'images numériques, compressées au format JPEG ou non.

Dans un premier temps, nous présentons les principes fondamentaux en stéganographie (section 6.2), nous proposons ensuite un panorama synthétique des méthodes de bases en stéganographie (section 6.3), puis nous terminons ce chapitre par une présentation de méthodes avancées (section 6.4).

6.2. Principes fondamentaux

Les contributions théoriques qui permettent de satisfaire les contraintes présentées précédemment font appel à la théorie de l'information, la théorie du codage et l'analyse d'images en général.

6.2.1. *Maximisation de la taille du message inséré*

La capacité d'insertion, c'est à dire la **taille maximale du message** que l'algorithme de stéganographie est susceptible d'insérer dans le contenu Cover se calcule en utilisant le *théorème du codage source* énoncé par Claude Shannon (Shannon 1948). Ce théorème, peut se présenter comme suit :

Soient,

– x_i un échantillon i de l'image Cover[7] et,

– un signal stéganographique associé à un échantillon i et définit par $s_i \in \{K_{\min}, \ldots, K_{\max}\}$, avec $\{p_i(K_{\min}), \ldots, p_i(K_{\max})\}$ les $(K_{\max} - K_{\min} + 1)$ probabilités de modification associées à chacune des valeurs de s_i. Notons ici que pour la plupart des schémas relativement sûrs, l'insertion n'autorise que des modifications binaires (soit $K_{\min} = -1, K_{\max} = 0$ ou $K_{\min} = 0, K_{\max} = +1$) ou bien ternaires ($K_{\min} = -1, K_{\max} = +1$),

– un schéma d'insertion additif, où l'échantillon du contenu Stégo est donné par $y_i = x_i + s_i$,

alors, le *théorème du codage source* stipule que la quantité d'information maximale portée par l'échantillon y_i est donné par l'entropie de la variable aléatoire S_i, soit :

$$H(S_i) = -\sum_{k=K_{\min}}^{K_{\max}} p_i(k) \log_2 p_i(k) \text{ bits.} \qquad [6.1]$$

Le lecteur averti notera que cette entropie ne dépend pas de l'échantillon hôte x_i, cela se justifie par le fait que le contenu hôte est connu par Alice, qui peut donc concevoir un système de codage qui n'a pas à prendre en compte ce signal, même s'il reste inconnu de Bob. Ce résultat est justifié par les résultats sur la prise en compte de l'information adjacente présentés dans (Costa 1983).

D'un point de vue pratique, la prise en compte du calcul de l'entropie est capitale. Prenons un exemple où Alice chercherait à insérer 100 bits dans 1000 pixels d'une image codée en niveau de gris (chaque pixel aura alors une valeur comprise en 0 et 255). Si Alice utilise une méthode rudimentaire de substitution de bits de poids faible (aussi appelée substitution LSB[8], voir section 6.3.1), en moyenne 50 pixels sont ainsi modifiés et la probabilité de modification d'un pixel est donc de $50/1000 = 0.05$.

7. par échantillon nous entendons la valeur d'un pixel d'une image pour l'un des trois canaux couleur si elle est codée sans perte, ou encore la valeur d'un coefficient DCT pour le format JPEG

8. les méthodes de substitution des bits de poids faibles (voir également section 6.3.1) sont des méthodes rudimentaires où pour insérer un message de N bits, les bits de poids faible de N pixels sélectionnés grâce à la clé secrète sont remplacés directement par le message à transmettre.

Le théorème du codage source montre que pour un même nombre moyen de modifications, la taille maximale du message qu'il est possible d'insérer en utilisant une méthode de codage moins naïve que la substitution LSB est égale à :

$$1000\,(-0.05\log_2(0.05) - 0.95\log_2(0.95)) \approx 286\,\text{bits},$$

ainsi en passant de 100 bits à 286 bits, la longueur du message insérée est presque trois fois supérieure à celle du message inséré par substitution LSB !

6.2.2. *Codage du message*

Trouver une méthode de codage utilisable en stéganographie et se rapprochant de l'entropie de Shannon ne fût pas une tâche facile. Les méthodes de codage par code linéaires proposées très tôt par Crandal (Crandall 1998) ont permis de d'augmenter la capacité d'insertion (pour un nombre de modification moyen donné) sans pour autant se rapprocher vraiment de la limite théorique. Comme des méthodes plus évoluées, ces méthodes reposent sur l'idée qu'il existe plusieurs contenus Stego codant le même message, et qu'un codage judicieux consiste à sélectionner le contenu stégo qui se trouve être le plus proche du contenu Cover (voir la figure 6.2).

Prenons l'exemple jouet d'un contenu Cover composé de 3 bits tirés aléatoirement $[x_1, x_2, x_3]$ et son homologue Stégo $[y_1, y_2, y_3]$, et supposons que le système de codage cherche à insérer 2 bits en minimisant le nombre moyen de modifications. La méthode de substitution LSB[9] consiste à sélectionner à l'aide d'une clé secrète 2 bits sur 3 et à les remplacer par le message à insérer. En moyenne, chaque bit aura donc une probabilité de $\frac{1}{2} \times \frac{2}{3} = \frac{1}{3}$ d'être modifié. Si maintenant nous utilisons un codage linéaire, qui consistera par exemple à convenir que le premier bit sera codé[10]

9. voir encore une fois la section 6.3.1

10. $\oplus$ étant l'opérateur XOR.

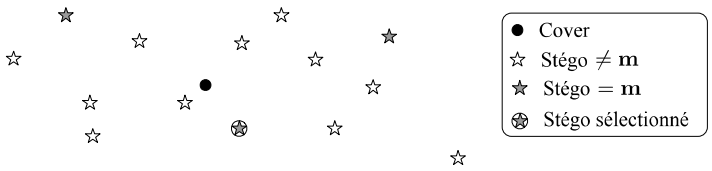


Figure 6.2 – Codage pour la stéganographie : le système de codage génère plusieurs mots de codes associés au même message m afin de sélectionner celui qui se trouve le plus proche du contenu Cover et ainsi minimiser la distorsion entre le contenu Cover et le contenu Stégo.

par $m_1 = y_1 \oplus y_2$ et le second par $m_2 = y_2 \oplus y_3$; x_2 ne sera alors modifié que s'il permet de changer à la fois la valeur de $x_1 \oplus x_2$ et de $x_2 \oplus x_3$, soit avec une probabilité de 0.25. De plus, x_1 (resp. x_3) ne sera modifié que s'il permet de changer uniquement la valeur de $x_1 \oplus x_2$ (resp. $x_2 \oplus x_3$) soit également avec une probabilité de 0.25 pour chacun, et il y aura également une probabilité de 0.25 pour que les 2 bits à insérer soient déjà présent dans le contenu Cover. Au final la probabilité de modification de chaque bit ne sera que de 0.25 contre 0.33 pour le schéma de substitution LSB. Notons encore une fois que le théorème du codage source dit que pour une probabilité de modification de 0.25, la taille maximale du message inséré est de $-0.25\log_2(0.25) - 0.75\log_2(0.75) \approx 0.81$ bit inséré par échantillon modifié au lieu de 0.75 bit par échantillon comme proposé par la méthode décrite.

Cependant en 2010, la référence (Filler *et al.* 2011) a proposé un moyen pratique de se rapprocher de cette borne maximale grâce d'une part à la construction de codes utilisant un système de codage par syndromes, et d'autre part grâce à l'utilisation d'un treillis . Ces codes sont appelés STC pour « Syndrom Trellis Codes », ils combinent le principe du codage par syndrome et un algorithme d'optimisation séquentiel qui permet d'insérer le message voulu tout en minimisant le nombre de modifications nécessaires (l'algorithme de Viterbi qui parcourt le treillis). Notons que ce système de codage peut être utilisé plus généralement avec n'importe quelle distorsion[11] additive entre l'image Cover et l'image Stégo. Nous n'avons pas la place dans ce chapitre de détailler le fonctionnement du STC, mais le lecteur peut retenir que dans le cas d'une distorsion additive, les performances du STC sont très proches de la borne maximale donnée par l'entropie de Shannon.

6.2.3. *Minimisation de la détectabilité*

Comme précisé dans l'introduction, un algorithme de stéganographie ne sera sûr que s'il est indétectable, ce qui arrivera si Ève n'arrive pas à soupçonner la présence d'un message caché en analysant un contenu Stégo.

Il n'est pas réaliste de considérer que la détectabilité est directement proportionnelle au nombre de modifications effectuées sur l'image Cover pour insérer le message. A titre d'exemple, un pixel modifié dans une portion bruitée de l'image (une texture par exemple) contribuera beaucoup moins à augmenter la détectabilité de l'image qu'un pixel modifié dans une zone homogène de l'image, voir au pire des cas une zone constante. En effet, il est facile d'intuiter qu'Ève aura beaucoup plus de facilités à trouver une modification suspecte si celle-ci apparaît dans une zone aux fluctuations faibles (voir nulles lorsque les niveaux des pixels sont saturés) que dans une zone compliquée à modéliser. La mesure de la détectabilité ne peut donc pas être une

11. voir la section 6.2.3 pour une définition plus précise de la distorsion.

simple distance euclidienne entre l'image Cover et l'image Stégo, et il est nécessaire d'analyser au préalable l'image Cover afin de pouvoir associer à chaque échantillon de l'image une détectabilité empirique qui lui est propre.

La formalisation théorique adoptée dans la plupart des cas pour minimiser la détectabilité est l'utilisation de **coûts d'insertion additifs**. Chaque échantillon de l'image Cover x_i est associé à un coût ρ_i^k pour chaque modification k (cela revient à considérer que chaque échantillon s_i du signal stéganographique est tiré indépendamment des autres). Les coûts sont additifs car la modification d'un échantillon suite à l'insertion n'entraîne pas de modification sur les coûts relatifs aux autres échantillons[12]. L'espérance mathématique de la distorsion D entre l'image Cover et l'image stego est donc donnée par :

$$\mathrm{E}[D] = \sum_{i,k} p_i^k \rho_i^k. \qquad [6.2]$$

Par manque de place, nous ne pouvons pas détailler les calculs permettant de faire ressortir la relation entre le coût de modification ρ_i^k et la probabilité de modifier chaque échantillon p_i^k, mais il faut retenir qu'une formulation Lagrangiène[13] permet de minimiser la distorsion donnée dans l'équation [6.2] tout en respectant la contrainte liée à l'entropie, équation [6.1], afin de mettre en évidence une relation explicite entre ces deux caractéristiques. A titre d'exemple, dans le cas d'une insertion binaire, et en considérant que le coût associé à aucune modification est nul, la relation entre le coût d'une modification ρ_i et sa probabilité p_i est la suivante :

$$\rho_i = \lambda \log \frac{1 - 2p_i}{p_i}, \qquad [6.3]$$

où λ est une constante dépendant uniquement de la taille du message à insérer.

Il est important de noter qu'en pratique ces coûts ρ_i^k peuvent être directement utilisés par les STC afin de minimiser une distorsion qui, lorsque le nombre d'échantillons est important, est très proche de D.

Si ce formalisme suit un cadre théorique rigoureux, le calcul des coûts ρ_i^k à partir de l'image Cover en est tout autre ! En effet ce calcul consiste à utiliser des fonctions heuristiques construites à partir de l'intuition qu'un coût doit être important si la valeur de l'échantillon Cover peut être facilement prédite à partir de son voisinage, et qu'au

12. des coûts non-additifs peuvent cependant être considérés via des méthodes de synchronisation, voir section 8.3.

13. voir `https://en.wikipedia.org/wiki/Lagrange_multiplier`

contraire un coût doit être faible si cette même prédiction devient difficile, c'est à dire si l'échantillon se trouve dans une zone bruitée.

La majorité des implémentations en stéganographie cherchent donc dans un premier temps à calculer pour chaque échantillon i le coût ρ_i^k relatif à chaque modification k, puis dans un second temps à utiliser un codage par STC pour insérer le message désiré tout en minimisant la somme des coûts relatifs à l'insertion. La décomposition en ces différentes étapes est illustrée sur la figure 6.3.

6.3. Stéganographie d'images numériques : méthodes de base

Dans cette section nous détaillons différentes implémentations en stéganographie, ces implémentations diffèrent principalement par la fonction utilisée pour calculer les coûts.

6.3.1. *Substitution et correspondance LSB*

La substitution des bits de poids faibles (aussi appelée substitution LSB pour Least Significant Bits) est une méthode extrêmement naïve qui consiste à remplacer les bits de poids faible d'un pixel ou d'un coefficient DCT d'une image par un bit du message à insérer. Le bit de poids faible étant le bit de parité, cette insertion revient donc à appliquer la règle spécifiée sur la figure 6.4(a). Le décodage du message inséré par Bob s'effectue très simplement en lisant les bits de parités des échantillons portant le message[14]. Si cette méthode de stéganographie est très simple, elle a deux inconvénients majeurs :

14. Ces échantillons peuvent être sélectionnés par Alice et Bob en utilisant une même clé secrète

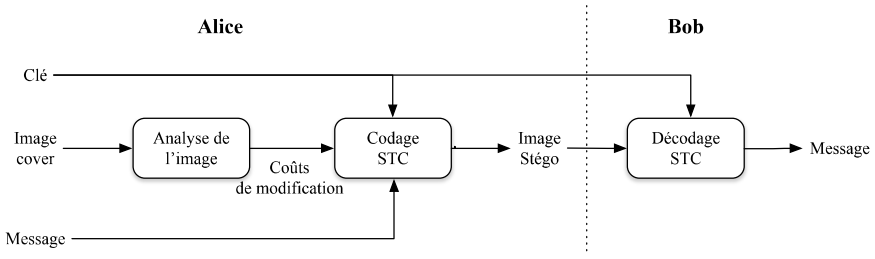
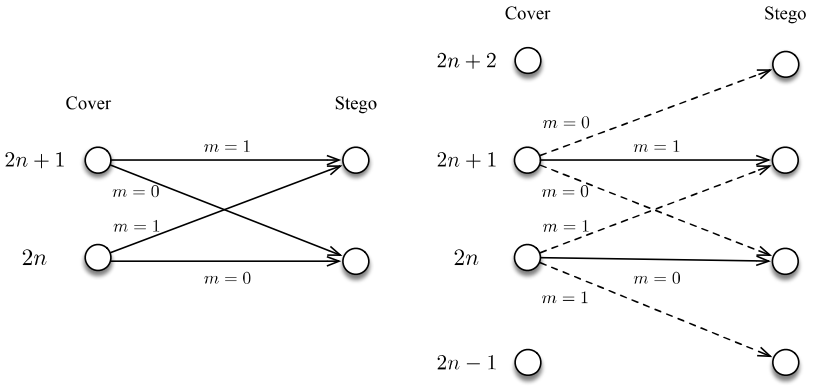


Figure 6.3 – Principe général de l'insertion, du codage et du décodage en stéganographie. La clé secrète peut être utilisée pour chiffrer le message, permuter les échantillons du contenu Cover, et/ou paramétrer le STC.

1) Comme indiqué dans la section 6.2.1, ce codage par substitution est clairement sous-optimal.

2) La sélection naïve des échantillons portant le message rend également cette méthode très détectable, il existe même des méthodes de stéganalyse permettant d'estimer facilement la taille du message inséré par la substitution LSB, comme par exemple présenté dans la référence (S.Dumitrescu *et al*. 2003) ou dans la section 7.3.2 du chapitre suivant. Une alternative à cette stratégie d'insertion consiste à rajouter un aléa en effectuant une opération de type $+1$ ou -1 de façon équiprobable pour changer la valeur du bit de poids faible : c'est le principe de la correspondance LSB qui est illustré dans la figure 6.4(b) et dont la stéganalyse est étudiée dans la section 7.3.3 du chapitre suivant.



(a) Substitution LSB. (b) Correspondance LSB.

Figure 6.4 – Principes de la substitution LSB et de la correspondance LSB, les flèches en pointillé représente une modification effectué avec une probabilité de 0.5, $n \in \mathbb{N}^+$.

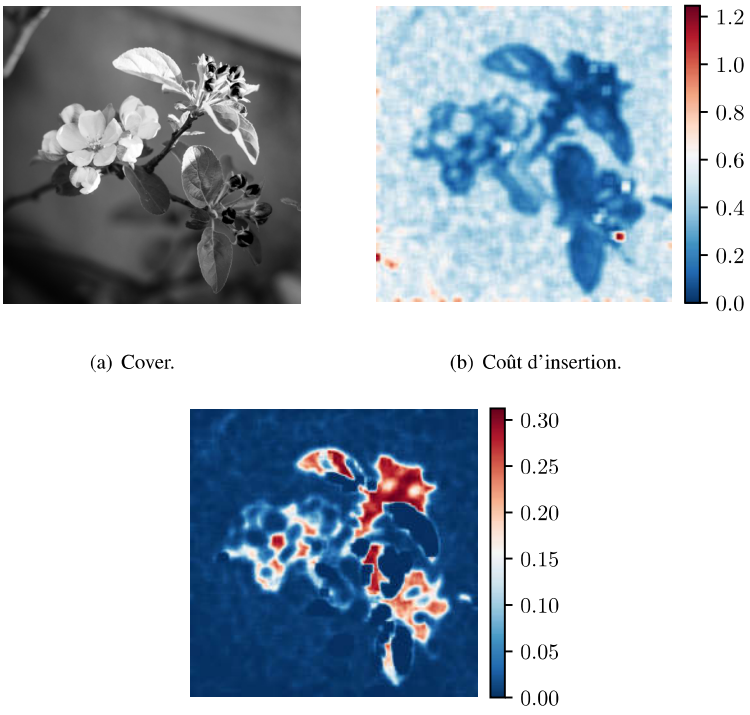
6.3.2. *Méthodes d'insertion adaptatives*

Nous présentons maintenant des méthodes de stéganographie **adaptatives**, c'est à dire des méthodes qui analysent au préalable l'image Cover afin d'associer à chaque échantillon un coût de détection. Ces méthodes utilisent toutes l'intuition qu'un échantillon (pixel ou coefficient DCT) qui se trouve dans une zone complexe ou bruitée de l'image sera associé à un coût plus faible (car l'échantillon sera difficilement prédictible) qu'un coefficient se trouvant dans une zone simple ou peu bruité de l'image.

Les deux premières méthodes présentées opèrent dans le domaine spatial, la troisième dans le domaine JPEG.

6.3.2.1. *Exemple de coûts dans le domaine des pixels (schéma HILL)*

Le schéma HILL est remarquable de part sa simplicité et son efficacité. Les auteurs qui ont proposé cet algorithme (Li *et al*. 2014) sont partis du principe que, pour une image codée dans le domaine spatial, les pixels appartenant à des zones texturées ou bruitées devaient être associés à un coût de modification faible alors que les pixels appartenant à des zones homogènes devaient être associés à un coût important. Pour ce schéma, la caractérisation du bruit s'effectue simplement par l'utilisation d'un filtre passe-haut $\mathbf{H}$ (filtre dérivateur), et de deux filtres passe-bas $\mathbf{L}_1$ et $\mathbf{L}_2$ qui sont également employés afin de prendre en compte les variations autour du voisinage du pixel



(a) Cover.

(b) Coût d'insertion.

(c) Probabilités d'insertion (0,3 bit par pixel).

Figure 6.5 – a) Image Cover, b) Carte de coût associée pour l'algorithme HILL. Les textures de l'image sont associées aux coûts les plus faibles. c) Carte de probabilité de modification, la probabilité maximal étant égale à $1/3$ pour une insertion ternaire, et le taux d'insertion maximal à $\log_2(3) \simeq 1.6$ bit par pixel.

considéré. Le calcul de la carte des coûts ρ pour une image $\mathbf{I}$, codée en niveau de gris, est donc le résultat de trois convolutions 2D :

$$\rho = \frac{1}{|\mathbf{I} * \mathbf{H}| * \mathbf{L}_1} * \mathbf{L}_2, \qquad [6.4]$$

avec

$$\mathbf{H} = \begin{bmatrix} -1 & 2 & -1 \\ 2 & -4 & 2 \\ -1 & 2 & -1 \end{bmatrix},$$

et $\mathbf{L}_1$ et $\mathbf{L}_2$ deux filtres passe-bas moyenneurs de taille respective 3×3 et 15×15.

La figure 6.5 montre pour une image en niveau de gris donnée, l'image de coûts associée. Nous pouvons constater que les coûts localisés dans les zones texturées sont beaucoup plus faibles que les coûts localisés dans les zones homogènes.

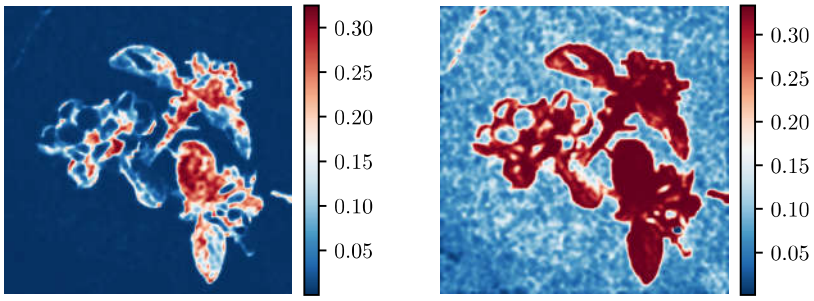
6.3.2.2. *Coût prenant en compte la détectabilité (schéma MiPod)*

MiPod est une méthode beaucoup moins heuristique qui consiste à associer à chaque coût une détectabilité statistique (Sedighi *et al.* 2016). La détectabilité statistique d'une modification peut être associée à une quantité, appelée déflexion, qui est liée à la probabilité que l'image ou le pixel considéré appartient à la classe Cover ou la classe Stégo. Plus la déflexion sera importante plus la preuve apportée envers l'hypothèse Stégo aura du poids. En posant comme hypothèse que chaque pixel avant quantification peut être considéré comme une variable aléatoire continue suivant une loi Gaussienne de variance σ_i^2, et que la probabilité d'effectuer une opération ± 1 lors de l'insertion est égale à p_i, le coût est alors donné par :

$$\rho_i = \frac{p_i}{\sigma_i^4}, \qquad [6.5]$$

le carré de la déflexion étant égal à p_i^2/σ_i^4 pour chaque échantillon.

Il est important de noter que le coût ne dépend pas uniquement des propriétés de l'image telle que la variance σ_i^2 (directement estimée à partir de la variance locale de chaque pixel de l'image), mais qu'il dépend également de la probabilité de modification p_i, l'équation [6.3] n'est donc pas directement applicable mais la résolution des deux contraintes présentées dans la section 6.1 reste cependant possible. Nous pouvons également remarquer que le coût décroit lorsque σ_i^2 augmente (le bruit associé à chaque pixel devient de plus en plus important) ou lorsque p_i diminue (la probabilité de modifier le pixel décroit). La figure 6.6 illustre les probabilités de modification de chaque pixel pour deux taux d'insertion différents.



(a) 0.3 bit par pixel. (b) 1.0 bit par pixel.

Figure 6.6 – Carte de probabilités de modification associées pour l'algorithme MiPod pour deux taux d'insertion différents.

6.3.2.3. *Exemple de coûts dans le domaine JPEG (schéma UERD)*

Le calcul des coûts proposé par le schéma UERD (Guo *et al.* 2015) s'effectue pour des images Cover compressées en JPEG, il est donc nécessaire d'associer un coût à chaque coefficient DCT quantifié c_i de l'image (q_i étant le pas de quantification associé). Les auteurs de ce schéma proposent de faire en sorte d'une part que les coefficients appartenant à des blocs texturés[15] (resp. homogènes) soient associés à un coût faible (resp. fort). La mesure de la « texture » se fait ici simplement en calculant l'énergie $D_{\mathcal{B}}$ du bloc JPEG 8×8 $\mathcal{B}$ contenant c_i. Cette énergie est définie par $D_{\mathcal{B}} = \sum_{i \in \mathcal{B}} q_i |c_i|$ et nous pouvons noter que le coût construit est proportionnel au pas de quantification q_i afin de traduire le fait qu'une opération du type $(+1)$ sur un coefficient quantifié entraîne une différence de q_i sur la transformée DCT de l'image. Hormis pour les coefficients DC qui ont un coût légèrement différent, le coût ρ_i associé à chaque coefficient c_i est ainsi donné par :

$$\rho_i = \frac{q_i}{D_{\mathcal{B}} + 0.25\left(D_{\mathcal{B}}^{\leftarrow} + D_{\mathcal{B}}^{\rightarrow} + D_{\mathcal{B}}^{\downarrow} + D_{\mathcal{B}}^{\uparrow} + D_{\mathcal{B}}^{\swarrow} + D_{\mathcal{B}}^{\searrow} + D_{\mathcal{B}}^{\nwarrow} + D_{\mathcal{B}}^{\nearrow}\right)}, \quad [6.6]$$

le deuxième membre du dénominateur permettant de prendre en compte les énergies des 8 blocs connexes au bloc considéré.

15. c'est à dire contenant beaucoup de coefficients DCT non-nuls

6.4. Principes avancés en stéganographie

Après avoir présenté les principes de bases de la stéganographie (codage et calcul de coûts), nous présentons maintenant différentes directions qui permettent d'améliorer encore la sécurité d'un schéma d'insertion, ou bien de lui permettre de s'adapter à des contextes différents. Nous tenons à préciser au lecteur que ces avancées souvent récentes restent associés à des domaines où la recherche amènera probablement encore d'autres évolutions dans le futur. Ces principes avancés sont rappelés ci-dessous :

– la **synchronisation des modifications** qui permet de corréler les modifications voisines de l'image,

– la **stéganographie d'images couleurs**, qui généralise les méthodes utilisées sur les niveaux de gris aux images à 3 composantes,

– la **stéganographie groupée**, qui permet de répartir le message au sein de plusieurs images,

– l'**utilisation de l'information adjacente** qui prend en compte les erreurs liées à la quantification pour améliorer la sécurité du schéma,

– la **stéganographie mimant un modèle statistique**, qui renforce la sécurité en rendant l'insertion fidèle à un modèle statistique de l'image,

– la **stéganographie adverse** qui utilise une insertion cherchant directement à contourner le stéganalyste,

– enfin la conclusion souligne la nécessité de développer des méthodes de **stéganographie robustes** au transcodage.

6.4.1. *Synchronisation des modifications*

Afin de diminuer la détectabilité d'un schéma d'insertion, une stratégie possible est de synchroniser les modifications. Si par exemple dans le domaine spatial, un pixel a été modifié par une opération de type $(+1)$, il pourra être intéressant de privilégier une insertion de type $(+1)$ plutôt que $(+0)$ ou (-1) sur un pixel voisin, et ce afin de favoriser l'apparition d'un signal stégo dont les variations sont similaires à celles de l'image. Il est donc nécessaire pour cela de rendre le calcul du coût d'un coefficient dépendant des modifications déjà effectuées dans son voisinage. C'est le principe de la synchronisation des modifications.

Une manière générale d'effectuer cette synchronisation est de décomposer la grille naturelle d'échantillonnage des coefficients codant l'image[16] en un ensemble de grilles disjointes[17] telles que pour une partition donné, chaque coefficient peut être

16. par exemple la grille naturelle des pixels de l'image dans le domaine spatial ou une matrice de coefficients DCT dans le domaine JPEG.

17. appelé partitions en français et lattices en anglais.

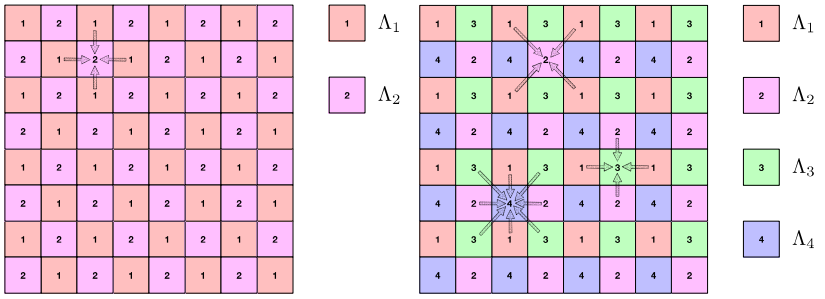
modifié de façon indépendante des autres, c'est à dire en utilisant un coût additif. Le principe est illustré sur la figure 6.7 où 4 partitions $\{\Lambda_1, \Lambda_2, \Lambda_3, \Lambda_4\}$ sont utilisées.

Il est important de distinguer deux grandes classes de méthodes de synchronisation :

1) les méthodes heuristiques, qui augmentent ou diminuent les coûts liés à l'insertion. La grande majorité des méthodes appartient à cette classe. A titre d'exemple si dans le domaine spatial un +1 a été effectué sur un pixel, le coût relatif à la modification $(+1)$ sur un pixel connexe sera diminué, au contraire un coût relatif à une modification (-1) sera augmenté.

2) les méthodes reposants sur des modèles probabilistes du signal stéganographique. Ici les coûts sont directement dérivés à partir des probabilités conditionnelles précisées plus haut, en appliquant ensuite les relations entre les coûts et les probabilités de modification.

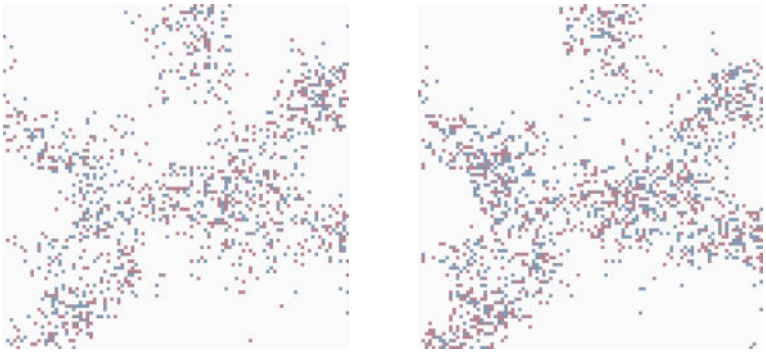
Afin d'illustrer une implémentation appartenant à la première classe de méthode, nous présentons brièvement la méthode proposée par (Li *et al.* 2015) via le schéma de synchronisation « CMD » (Clustering Modification Directions). Ce schéma peut par exemple être utilisé avec le coût proposé par HILL dans la section 6.3.2.1. Quatre



(a) Insertion sur 2 treillis (dépendance en 4 connexité).

(b) Insertion sur 4 treillis (dépendance en 8 connexité).

Figure 6.7 – Principe de l'insertion synchronisée en utilisant 2 ou 4 partitions : les coûts ou les probabilités d'insertions sont dans un premier temps calculés sur la partition Λ_1 où les modifications sont supposées indépendantes. L'insertion s'effectue ensuite sur Λ_1, puis les coûts ou les probabilités d'insertions sont calculés sur la partition Λ_2 en prenant en compte les modifications déjà effectuées sur Λ_1. Lorsque 4 partitions sont utilisées, l'algorithme itère ensuite jusqu'à la partition Λ_4 où les coûts sont calculés à partir des modifications effectuées sur Λ_1, Λ_2 et Λ_3. Les flèches représentent les relations de connexités utilisées à la fois pour prendre en compte les dépendances et mettre à jour les coûts.



(a) HILL. (b) CMD.

Figure 6.8 – Visualisation de modifications $(-1,+1)$ apportées à l'image sans synchronisation en utilisant : a) La méthode HILL, b) La stratégie CMD. Nous pouvons remarquer que les modifications de même signe sont davantage connexes lorsque la synchronisation est utilisée. Notons également que même si la sécurité du schéma est renforcée, le nombre de modifications augmente pourtant de 20% en moyenne après synchronisation.

partitions sont utilisés (voir figure 6.7(b)). Sans restreindre la généralité de ce qui précède, nous considérons ici une modification de type $(+1)$, soit μ_{ij} la moyenne des modifications déjà effectuées dans le voisinage du pixel (i,j) sur les précédentes partitions, le nouveau coût ρ'_{ij} est donné par :

$$\rho'_{ij}\left(+1\right) = \frac{1}{9}\rho_{ij}\left(+1\right), \text{ si } \mu_{ij} > 0, \qquad [6.7]$$

$$\rho'_{ij}\left(+1\right) = \rho_{ij}\left(+1\right), \text{ si } \mu_{ij} \leq 0. \qquad [6.8]$$

Notons que le poids $1/9$ est arbitraire et qu'il est utilisé pour maximiser les performances du schéma. L'insertion s'arrête une fois que les 4 partitions ont été parcourues. La figure 6.8 montre l'effet de la synchronisation sur les modifications $(-1/+1)$ apportées à l'image et montre comment cette stratégie permet de faciliter des modifications connexes. Le gain lié à la synchronisation en terme de sécurité pratique est appréciable puisque la probabilité d'erreur de détection pour le stéganalyste peut augmenter de 5% par rapport au schéma HILL pour un taux d'insertion de 0.4 bit par pixel.

6.4.2. *Stéganographie groupée*

La stéganographie groupée, présentée par Ker (Ker 2007), considère un scénario où Alice dispose, non pas d'une seule image pour insérer son message, mais d'un groupe d'images. Contrairement à un scénario de stéganographie classique où Alice ne transmet à Bob qu'une seule image, Alice devra ici répartir de message dans chacune des images composant le groupe. L'adversaire, Ève, pour elle concevoir une méthode d'analyse utilisant l'ensemble des images envoyées par Alice (ce qui donne lieu à la stéganalyse par accumulation, ou stéganalyse groupée, voir section 7.6.6 de l'ouvrage).

L'essentiel du problème lié à la stéganographie groupée réside donc dans la façon dont Alice va répartir son message dans chaque image. D'un point de vue pratique, ce scénario se justifie dans au moins deux situations : d'une part lorsque Alice transmet en une seule fois plusieurs images à Bob, d'autre part lorsque Alice communique de façon séquentielle des images à Bob. Notons que dans le premier cas la taille du message transmis sera connue alors que dans le deuxième cas il est possible qu'Alice doive adapter la taille totale du message à transmettre à ses besoins de communication. Sans rentrer dans les détails en terme de stéganalyse (ce n'est pas l'objectif de ce chapitre), il convient cependant de remarquer que plus Ève accumulera de contenus transmis par Alice à Bob, plus elle sera en mesure d'effectuer une analyse précise pour décider si Alice transmet ou non des contenus Stego à Bob.

Les principales méthodes de répartition du message étudiées dans la littérature (Pevnỳ and Nikolaev 2015 ; Cogranne *et al*. 2017 ; Zakaria *et al*. 2019) sont décrites ci-dessous :

– la transmission du message dans le moins d'images possible, les autres images du groupe appartenant ainsi à la classe Cover,

– la répartition du message de manière uniforme au sein du groupe, cette uniformité peut se traduire par :
  - une taille de message identique pour chaque image,
  - un taux d'insertion identique pour chaque image[18],

– mais aussi des stratégies de répartition prenant en compte le contenu des images
  - le coût de modification (voir équation [6.2]) est constant pour chaque image,
  - la distorsion statistique (voir équation [6.5]) est constante pour chaque image,
  - une minimisation globale des coûts appliquée sur l'ensemble des images. Notons que cette stratégie n'est possible que si Alice possède au préalable l'ensemble des images Cover pour les transformer en image Stégo, alors que ce n'est pas le cas pour les deux stratégies précédentes.

18. dans le cas d'images au format JPEG, le taux d'insertion s'exprimant souvent en bit pas coefficients AC non nuls, cela n'est pas équivalent à une taille identique.

Si les conclusions liées à la stratégie la plus sûre ne sont pas encore assez mûres pour être définitives (elles dépendent de la stratégie de stéganalyse de Ève), quelques résultats intéressants se dégagent :

– dans (Cogranne *et al*. 2017), la minimisation globale des coûts permet d'obtenir la stratégie la plus sûre pour un stéganalyste omniscient qui connaîtrait la stratégie de répartition

– lorsque cette stratégie doit être estimée (Zakaria *et al*. 2019), une distorsion statistique constante permettrait de maximiser la sécurité pratique.

Il est à noter que dans aucune des références citées, ni le fait de transmettre le message dans le moins d'images possibles, ni la stratégie consistant à répartir uniformément le message dans le groupe, ne semble être une alternative optimale.

6.4.3. *Stéganographie d'images couleurs*

Les méthodes de la littérature proposant dans la majorité des cas d'insérer le message dans une image en niveau de gris, un raccourci trop rapide consisterait donc à développer une méthode de stéganographie pour des images couleur[19] en insérant la même quantité d'information dans chaque canal de façon indépendante. Ce raisonnement est erroné pour plusieurs raisons :

– premièrement les composantes couleurs ne sont pas indépendantes, cette dépendance est importante pour les canaux Rouge/Vert/Bleu et beaucoup moins importante pour les canaux Luminance/Chrominance-rouge/Chrominance-bleue[20] utilisés par le codage JPEG.

– deuxièmement le stéganalyste dispose de beaucoup plus d'échantillons pour effectuer son analyse (au maximum 3 fois plus de données), et les résultats de (Ker *et al*. 2008) montrent qu'à détectabilité égale, la taille du message inséré doit théoriquement évoluer avec la racine carré du nombre d'échantillons pour un codage naïf.

– troisièmement le contenu de chacun des canaux peut être très différent d'un canal à l'autre : s'il est souvent très similaire pour des images RVB, pour les images JPEG codées en Luminance/Chrominance-rouge/Chrominance-bleue, le contenu perceptif est beaucoup plus important sur la composante de luminance que sur les deux composantes de chrominance. Il conviendra donc d'insérer moins d'information sur ces deux composantes.

Une manière un peu plus rigoureuse d'aborder le problème de la stéganographie couleur est donc de l'appréhender comme un problème de stéganographie groupée

19. *i.e.* codées sur trois composantes

20. voir `https://fr.wikipedia.org/wiki/Chrominance`

(voir section 6.4.2) où Alice doit répartir le message au sein de chacun des trois canaux. Une telle approche a été développée par (Cogranne *et al*. 2020) dans le cas des images JPEG. Les différentes stratégies évoquées dans la section 6.4.2 sont appliquées et les auteurs observent que la stratégie par minimisation de la distorsion statistique est la plus souvent la plus efficace. Une alternative, proposée par (Taburet *et al*. 2018) consiste à fixer directement les proportions du message du message à répartir dans les trois composantes. La sécurité pratique est maximisée lorsque 80% du message est inséré dans la composante de luminance lorsqu'aucun sous-échantillonnage n'est appliqué sur les canaux de chrominance[21].

Enfin, lorsque les images sont codées dans le domaine spatial, les travaux de (Wang *et al*. 2019) ont montré l'importance de pouvoir synchroniser les modifications à la fois entre les pixels voisins et les composantes couleurs. Ce schéma propose d'effectuer l'insertion dans un premier temps sur canal vert de l'image en utilisant l'algorithme CMD (voir section 6.4.2) pour ensuite mettre à jour les coûts sur les canaux rouge et bleu en fonction des modifications faites sur le canal vert. Le canal vert est sélectionné en premier car c'est celui qui est le plus corrélé avec les deux autres canaux. Les règles de mise à jours sont identiques aux règles de CMD présentées dans les équations [6.7] et [6.8] mais le terme $\mu_{i,j}$ prend maintenant en compte la modification effectuée dans le canal vert. Par rapport à l'algorithme HILL naïf, le gain en terme de sécurité pratique est substantiel (le taux d'erreur augmente de plus de 10%) lorsque les taux d'insertion sont de l'ordre de 0.4 bit par pixel.

6.4.4. *Utilisation de l'information adjacente*

Les méthodes d'insertion par information adjacente utilisent une mesure fiable d'indétectabilité basée sur l'erreur de quantification produite lors d'une transformation de l'image comme une compression JPEG, une transformation géométrique ou colorimétrique suivie d'une re-compression. Le terme « information adjacente (side information » en anglais) relève de « l'image pré-Cover », c'est à dire de l'image avant quelle ne soit transformée en image Cover, et qui est codée soit dans le domaine continu, soit dans un domaine haute-résolution (par exemple 16 bits pour l'image pré-Cover contre 8 bits pour l'image Cover).

Dans le cas d'une insertion binaire, de façon intuitive il est assez facile de se rendre compte que si le coefficient avant quantification se trouve presque à égale distance de deux cellules de quantification (cas (a) de la figure 6.9), alors ce coefficient pourra être modifié indifféremment vers l'une ou l'autre des cellules et dans ce cas-ci la modification sera indétectable. Si au contraire le coefficient avant quantification se

21. voir `https://en.wikipedia.org/wiki/Chroma_subsampling` pour plus d'information sur le sous-échantillonnage couleur.

trouve être très proche d'une cellule de quantification (cas (c) de la figure 6.9), alors modifier ce coefficient vers une cellule moins proche sera plus détectable.

Nous pouvons citer par ordre chronologique plusieurs implémentations qui utilisent la notion d'information adjacente.

Dans le cas de la double compression JPEG, l'algorithme PQ (pour « *Perturbed Quantization* » (Fridrich *et al.* 2004) et (Fridrich *et al.* 2005)) ont proposé d'utiliser les wet-paper codes pour ne modifier que les coefficients DCT tels que $(\frac{1}{2} - e) = 0$, où e représente l'erreur de quantification associée au pas de quantification q (voir figure 6.9). Filler *et al.* revisitent cette assertion en précisant que le coût doit être égal à $\rho_{SI} = (\frac{1}{2} - e)$ pour les coefficients non nuls qui eux ne changent pas (Filler *et al.* 2011). Cette stratégie est appelée S_1.

L'article pionnier présentant les STCs présenté par (Filler *et al.* 2011) propose également trois autres stratégies appelées S_2, S_3 et S_4. Dans chacun des cas, les coefficients nuls ne sont pas modifiés. Notons également qu'ici les coûts ne dépendent que du contenu pré-Cover (*i.e.* avant quantification), contrairement aux schémas adaptatifs qui ont été présentés précédemment.

– Pour S_2, $\rho_{SI} = (\frac{1}{2} - e)q$.

– Pour S_3, $\rho_{SI} = 1$ si la valeur quantifiée et égale à -1 ou 1 et $\rho = 2\left(\frac{1}{2} - e\right)$ sinon.

– Pour S_4, $\rho_{SI} = q$, si la valeur quantifiée et égale à -1 ou 1 et $\rho = 2\left(\frac{1}{2} - e\right)q$ sinon.

Parmi les quatre stratégies S_1, S_2, S_3 et S_4, les auteurs de (Filler *et al.* 2011) montrent que c'est la stratégie S_4 qui se trouve être la moins détectable.

L'article présenté par Denemark et Fridrich (Denemark and Fridrich 2015) passe en revue l'utilisation de l'information adjacente pour les algorithmes d'insertion adaptatifs modernes S-Uniward (Holub *et al.* 2014), HILL et J-Uniward (Holub *et al.*

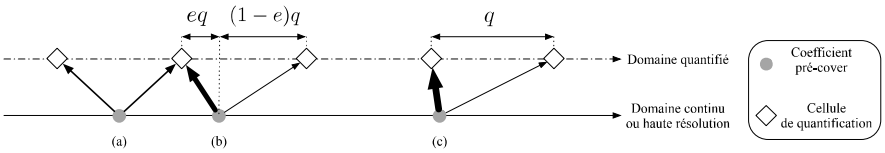


Figure 6.9 – Principe de pondération par l'erreur de quantification : la situation (a), où la valeur non quantifiée est à égale distance de deux valeurs quantifiées, sera d'avantage propice à la modification de la valeur vers la zone quantifiée voisine que la situation (c), où la valeur non-quantifiée est très proche de la valeur quantifiée. La situation (b) est intermédiaire.

2014). Pour chacun des algorithmes, une insertion binaire et une insertion ternaire sont envisagées. Pour l'insertion binaire, la relation entre le coût adaptatif ρ_{Ad} et le nouveau coût est :

$$\rho_{SI} = 2\left(\frac{1}{2} - e\right)\rho_{Ad}. \qquad [6.9]$$

Nous pouvons remarquer que le pas de quantification q a été ici remplacé par le coût adaptatif ρ_{Ad}, ce qui revient à considérer que cette implémentation cherche à minimiser la différence entre la somme des coûts engendrés par la conversion du contenu pré-Cover au contenu Cover. Notons également que le facteur 2 ne change rien au processus d'optimisation qui consiste à minimiser la somme des coûts.

Pour une insertion ternaire, les auteurs du même article proposent deux modulations différentes. Sans perte de généralité, si la valeur non-quantifiée est plus proche de la valeur inférieure que de la valeur supérieure (ce qui est le cas pour l'exemple (b) de la figure 6.9), alors l'opération (-1) ne prend pas en compte l'information adjacente (c'est un choix arbitraire, peut-être motivé par l'expérience) :

$$\rho_{SI}^{(-1)} = \rho_{Ad}, \qquad [6.10]$$

alors que l'opération $(+1)$ la prend en compte :

$$\rho_{SI}^{(+1)} = 2\left(\frac{1}{2} - e\right)\rho_{Ad}. \qquad [6.11]$$

Il est à noter que l'utilisation de l'information adjacente permet le plus souvent d'accroître considérablement les performances du schéma d'insertion en terme de sécurité. A titre d'exemple, la version SI de J-Uniward pour un facteur de qualité de 75% permet d'augmenter le taux d'erreur en stéganalyse de plus de 20% pour un taux d'insertion se situant autour de 0.4 bit par coefficient AC non nul.

6.4.5. *Stéganographie mimant un modèle statistique*

Une autre stratégie, relativement intuitive, pour cacher de l'information est celle du camouflage (contrairement aux méthodes qui utilisent un coût mesurant une distorsion statistique), ici l'idée est de directement mimer les propriétés statistiques de l'image Cover. A titre d'exemple illustratif, si le contenu Cover peut être complètement caractérisé par une distribution donnée $f()$, il « suffit » de générer des contenus stégo suivant également la distribution $f()$.

Une telle opération n'est cependant concevable que lorsqu'il est possible de caractériser sans aucune ambiguïté $f()$. Si ce n'est pas le cas, l'insertion devient forcement détectable. Nous présentons dans cette section deux exemples qui se basent sur ce principe de mime, l'un relativement ancien et l'autre plus récent.

6.4.5.1. *Distribution des coefficients DCT*

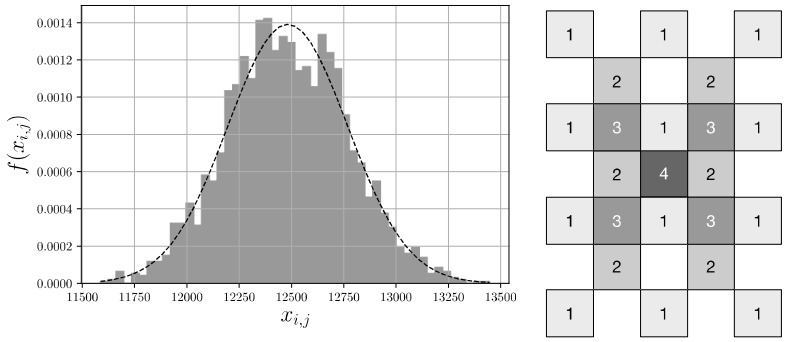
La référence (Sallee 2003) propose de mimer la distribution des coefficients DCT d'une image JPEG. La distribution $f()$ choisie est une distribution de Cauchy généralisée de type $f(x) = \frac{p-1}{2s}(|x/s|+1)^{-p}$ où p et s sont 2 paramètres à estimer. L'auteur prend comme hypothèse que les coefficients sont statistiquement indépendants les uns des autres. Le « mime » s'effectue en faisant en sorte que la distribution des coefficients après mise à zéro des bits de poids faible reste inchangée. L'insertion du codage est basée sur un décodeur arithmétique qui permet d'insérer un message en respectant au mieux les probabilités d'apparition de chacune des valeurs des coefficients. La seule contrainte liée à ce type de codage est que l'histogramme des coefficients avec leurs bits de poids faibles à zéro doit rester inchangé afin de pouvoir construire le décodeur, ce qui est possible pour cet implémentation, mais peut être difficile dans d'autres conditions.

Cette méthode fut à la fois pionnière tout en proposant des concepts avancés en stéganographie (mime d'un modèle, codage avancé) et elle est restée longtemps une référence en stéganographie. Elle est cependant devenue relativement détectable par les méthodes de stéganalyse qui captent les dépendances entre les coefficients DCT puisque celles-ci ne sont pas préservées.

6.4.5.2. *Bruit photonique*

Beaucoup plus récemment (Bas 2016 ; Taburet *et al.* 2020) ont cherché à mimer le modèle statistique du bruit photonique soit au niveau des pixels de l'image soit au niveau des coefficients DCT. Ici, l'insertion stéganographique (appelée « Stéganographie Naturelle ») ne reproduit pas le modèle statistique de l'image Cover, mais mime le modèle statistique d'une image Cover qui aurait été acquise avec une sensibilité ISO supérieure à celle de l'image Cover.

La distribution statistique de ce bruit photonique dépend de la chaîne de traitement qui permet de passer de l'image RAW enregistrée par le capteur à l'image Cover (codée dans le domaine spatial ou dans le domaine JPEG). Dans le domaine RAW, le modèle statistique est simple puisque le bruit photonique appliqué à chaque photo-site est indépendamment distribué suivant une loi Normale $\mathcal{N}(0, a\mu + b)$ où μ représente la valeur sans bruit du photo-site, et (a, b) sont des constantes liées uniquement au capteur et au paramètre ISO_1 (voir également figure 6.10(a)). Dans le domaine RAW, le signal stéganographique ajouté aux photo-sites est également distribué par une loi Normale $\mathcal{N}(0, a'x+b')$ où x est la valeur du photo-site observée et (a', b') sont choisis pour mimer une sensibilité $\mathrm{ISO}_2 > \mathrm{ISO}_1$. En faisant l'hypothèse que $x \simeq \mu$, l'image semble avoir été bruitée par un bruit photonique de distribution $\mathcal{N}(0, (a+a')\mu+b+b')$.



(a) Histogramme normalisé du bruit photonique (Leica Monochrome, ISO 4000) dans le domaine RAW et distribution Normale associée (pointillés).

(b) Blocs JPEG intervenants pour une insertion effectuée dans le bloc central.

Figure 6.10 – La Stéganographie Naturelle cherche à mimer la distribution Normale du bruit photonique (a), cette distribution dans le domaine JPEG devient multivariée, elle doit utiliser une décomposition en treillis pour permettre la prise en compte des corrélations entre les coefficients DCT (b).

Une image développée et transmise par Alice étant rarement dans le domaine RAW, il convient ensuite de modéliser la distribution du bruit photonique dans le domaine spatial ou dans le domaine JPEG. Dans les deux cas, lorsque le développement est linéaire, la distribution du signal stéganographique est une Gaussienne multivariée qui prend en compte les corrélations entre les pixels voisins ou les coefficients DCT voisins. D'un point de vue pratique, il convient donc de synchroniser les modifications (voir section 8.3) sur un nombre de treillis qui peut être très important (4×64, voir également figure 6.10(b)). De plus, la conversion entre la loi de probabilité mimant le bruit photonique et les coûts utilisés lors de l'insertion s'effectue en utilisant l'équation [6.3].

Lorsqu'il est possible de modéliser correctement la chaîne de développement de l'image, la Stéganographie Naturelle permet d'insérer une quantité d'information importante[22] tout en garantissant également une bonne sécurité pratique. Ces bonnes performances s'explique d'une part par la méthodologie utilisée (mimer un modèle statistique), d'autre part par l'utilisation de schémas de synchronisations (voir section 8.3) et de l'utilisation de l'image RAW qui exploite l'erreur d'arrondie (voir section 6.4.4).

22. d'autant plus importante que l'écart entre les deux paramètres ISO l'est.

6.4.6. *Stéganographie adverse*

Les méthodes de stéganographie adverses prennent automatiquement en compte un adversaire, la stéganalyste Ève pour générer les coûts d'insertion. Nous pouvons envisager deux familles de méthodes adverses décrites ci-dessous : les méthodes par contournement et les méthodes utilisant des générateurs adverses.

6.4.6.1. *Stéganographie par contournement*

Les méthodes de stéganographie adverses sont conçues de telle manière à ce qu'Alice cherche directement à contourner l'adversaire, en l'occurrence ici la stéganalyste Ève. Ces schémas peuvent devenir itératifs : Alice utilise dans un premier temps un schéma d'insertion $\mathrm{Ins}_1()$ et Ève construit une méthode de stéganalyse $\mathrm{An}_1()$ qui cherche à détecter $\mathrm{Ins}_1()$. Dans un second temps Alice cherche ensuite à concevoir un schéma d'insertion $\mathrm{Ins}_2()$ qui va contourner $\mathrm{An}_1()$. En fonction du scénario de sécurité considéré, c'est à dire en fonction des connaissances qu'Alice a sur la stratégie de Ève et réciproquement, ce jeu entre Alice et Ève peut ensuite continuer à évoluer. Ève peut construire par exemple une méthode $\mathrm{An}_2()$ cherchant à détecter $\mathrm{Ins}_2()$ ou $\{\mathrm{Ins}_1(), \mathrm{Ins}_2()\}$, Alice peut ensuite chercher à contourner ce nouveau classifieur via un schéma $\mathrm{Ins}_3()$, et ainsi de suite.

Le premier schéma d'insertion adverse a été proposé par (Kouider *et al.* 2013), ici les coûts utilisés par $\mathrm{Ins}_2()$ sont construits à partir d'un ensemble de classifieurs linéaires (voir section 7.4.2) appris pour détecter $\mathrm{Ins}_1()$. Ainsi, si une modification de type +1 est faite sur un coefficient x_i, le coût ρ_i sera tel que :

$$\rho_i = \sum_k [f_k(x_i + 1) - f_k(x_i)], \qquad [6.12]$$

$f_k()$ étant la fonction qui retourne la valeur avant seuillage pour le k^{ieme} classifieur.

De cette manière les coûts faibles sont associés à des modifications non détectables.

Plus récemment (Tang *et al.* 2019) ont proposé un schéma adverse ou l'adversaire Ève a entraîné un réseau de neurones profond $\mathrm{An}_1()$. Ces réseaux ont deux avantages, ils sont devenus la référence en stéganalyse (voir section 7.5 du chapitre suivant) et la fonction $\mathrm{An}_1()$ est une fonction qui se différencie simplement et rapidement. De

façon générale, les coûts de certains coefficients DCT de l'image sont modifiés de la façon suivante pour une insertion $(+1)$:

$$\rho'_i(+1) = \begin{cases} \frac{1}{\alpha}\rho_i(+1) \text{ si } \frac{\partial \mathrm{An}_1}{\partial x_i}(\mathbf{x}) < 0, \\ \rho_i(+1) \quad \text{si } \frac{\partial \mathrm{An}_1}{\partial x_i}(\mathbf{x}) = 0, \\ \alpha\rho_i(+1) \text{ si } \frac{\partial \mathrm{An}_1}{\partial x_i}(\mathbf{x}) > 0, \end{cases} \quad [6.13]$$

où α vaut habituellement 2.

Ainsi les coûts diminuent si la modification permet de déplacer l'image de la région Cover vers la région Stégo (dérivée négative pour une opération $(+1)$) et augmentent dans le cas contraire. Cette nouvelle méthode d'insertion $\mathrm{Ins}_2()$, appelée ADV-EMB (pour « adversarial embedding ») permet le plus souvent de contourner le classifieur $\mathrm{An}_1()$ mais ne propose pas de solution à Alice si Ève propose à nouveau d'entraîner un nouveau détecteur $\mathrm{An}_2()$ ciblé sur $\mathrm{Ins}_2()$ car à l'étape suivante Alice devra faire face à deux adversaires, $\mathrm{An}_1()$ et $\mathrm{An}_2()$.

Pour répondre à ce problème Bernard *et al.*ont proposé une stratégie, consistant à l'itération k, à sélectionner l'image Stégo adverse générée lors des k premières itérations (chaque image contournant le dernier classifieur entraîné par Ève) qui contourne au *mieux* (c'est à dire celle qui est considérée le plus comme une image Cover) le *meilleur* des k classifieurs (voir également la figure 6.11 qui illustre la stratégie) (Bernard *et al*. 2019). En théorie du jeu, cette stratégie s'appelle une stratégie $\min\max$. Une telle stratégie permet à chaque itération d'augmenter la sécurité pratique du schéma d'insertion. Le gain peut être significatif, par exemple la version adverse pour le schéma UERD (voir section 6.3.2.3) permet d'augmenter après 8 itérations le taux d'erreur en détection de plus de 10% pour un taux d'insertion de 0.4 bits par coefficient AC non nul.

6.4.6.2. *Stéganographie par générateur adverse*

Si la stéganographie par contournement attaque un schéma d'insertion spécifique en cherchant à le contourner, il existe des méthodes adverses qui apprennent automatiquement à générer des insertions cherchant à être indétectables. Ces méthodes, basées sur le concept de réseaux générateurs adverses[23], utilisent deux réseaux de neurones, l'un jouant le rôle du stéganographe (Alice), l'autre du stéganalyste (Ève). L'optimisation des contraintes respectives d'Alice et de Ève se fait conjointement. Le principe de base de ce type de schéma présenté la première fois par (Tang *et al*. 2017) et amélioré

23. aussi appelés *Generative Adversarial Networks* où GANs en anglais), voir (Goodfellow *et al*. 2014)).

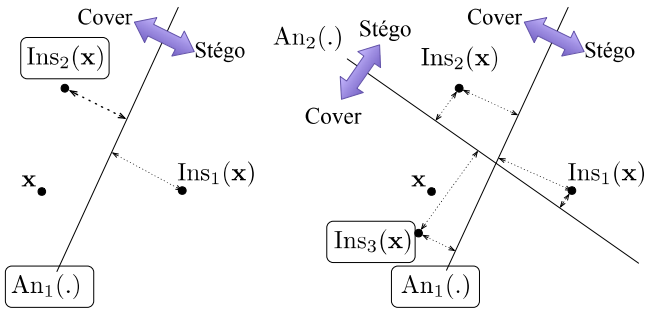
par (Yang *et al*. 2019), est illustré sur la figure 6.12, où deux réseaux de neurones sont présentés :

1) un réseau générateur de probabilités d'insertion, qui analyse l'image afin de prédire pour chaque pixel (i), une probabilité de modification p_i. Le coût associé ρ_i est ensuite calculé en utilisant l'équation [6.3]. Ce réseau est associé à une fonction de perteliée à la taille du message à insérer via l'équation [6.1]. Ce générateur analyse l'image Cover en se basant sur un réseau initialement dédié à la segmentation d'images.

2) un réseau adverse, jouant le rôle du stéganalyste, utilisant un réseau de stéganalyse comme détaillé dans la section 7.5 du chapitre suivant, et qui est associé à une fonction de perte liée à la détectabilité de l'insertion utilisée.

L'apprentissage des deux réseaux s'effectue de façon alternée :

1) la fonction de perte du générateur est une somme pondérée de la perte associée à la taille du message inséré et de la perte associée à la détectabilité. L'objectif est ici d'apprendre un générateur capable d'insérer un message de taille désirée tout en minimisant la détectabilité.



(a) $k = 1$. (b) $k = 2$.

Figure 6.11 – Deux premières itérations de la stratégie proposée par (Bernard *et al.* 2019) sur un exemple jouet où les adversaires sont des classifieurs linéaires et où la distance signée à la frontière de décision représente la valeur de sortie des fonctions $\mathrm{An}_k()$. a) Première itération, pour un contenu Cover x le contenu Stégo originel, $\mathrm{Ins}_1(\mathrm{x})$ est logiquement remplacé par sa version adverse $\mathrm{Ins}_2(\mathrm{x})$ (encadré) ; à cette étape Ève ne dispose que d'un seul classifieur $\mathrm{An}_1()$. b) Ève construit un nouveau classifieur $\mathrm{An}_2()$ qui est à nouveau contourné par Alice en créant le contenu $\mathrm{Ins}_3(\mathrm{x})$. Au final, le contenu sélectionné par Alice sera le contenu $\mathrm{Ins}_3(\mathrm{x})$, il s'agit en effet du contenu le plus difficile à classer par le meilleur classifieur, ici $\mathrm{An}_1()$, puisque celui-ci maximise sa sortie pour le meilleur contenu adverse.

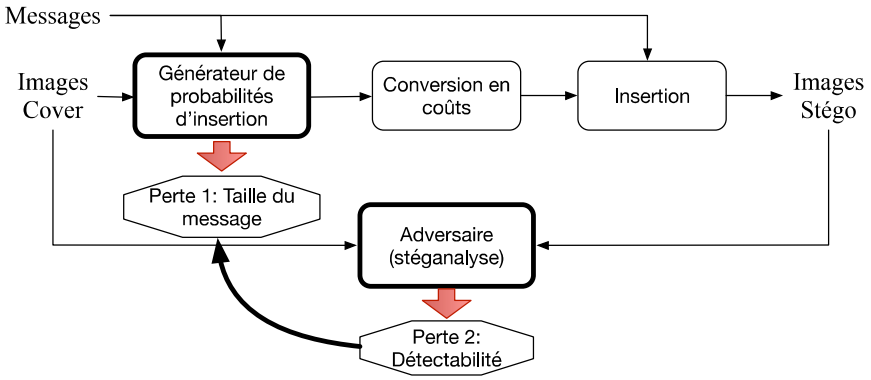


Figure 6.12 – Principe de la stéganographie basée sur des générateurs adverses.

2) L'adversaire lui cherche uniquement à maximiser la détectabilité. Comme classiquement en stéganalyse, sa fonction de perte utilise un ensemble d'image Cover et d'images Stego pour l'entraînement.

Afin que l'apprentissage de ces deux réseaux soit possible, il est nécessaire que l'ensemble des opérations effectuées soient différentiables : la fonction générant une image Stégo doit pouvoir se différentier par rapport à ses paramètres. La fonction d'insertion est donc modifiée de façon à prendre en compte cette contrainte.

En terme de performances, le schéma d'insertion obtenu par Yang *et al.*, après convergence du système permet d'avoir des performances en terme de détectabilité qui sont légèrement supérieures au schéma HILL (voir section 6.3.2.1), ce qui est remarquable compte tenu du caractère totalement automatique de cette approche (Yang *et al*. 2019).

6.5. Conclusions et perspectives

Ce chapitre se conclut en précisant que le domaine de la stéganographie, qui a vu ses premières formalisations à la fin du XXe siècle, reste en constante évolution. Ces évolutions sont liées au fait que le domaine de la stéganalyse progresse aussi, mais également que de nouvelles pistes (voir la section 6.4) ont vu le jour.

Si d'autres pistes de recherche ont été présentées par Ker *et al.* (insertion non additive, lois de mise à l'échelle, aspects liés à la sécurité), il convient également de préciser que les méthodes de stéganographie futures devront de plus en plus prendre

en compte le transcodage[24] possible des contenus Stégo (Ker *et al*. 2013). En effet cette opération est de plus en plus utilisée par les réseaux sociaux ou bien les plateformes de visio-conférence pour avoir un débit de transmission qui puisse s'adapter à la qualité de service. La stéganographie devra alors prendre en compte des contraintes liées au tatouage de documents (voir (Bas *et al*. 2016)), afin de permettre une transmission indétectable mais également robuste à ce traitement devenu usuel.

6.6. Bibliographie

Bas, P. (2016), Steganography via Cover-Source Switching. IEEE Workshop on Information Forensics and Security (WIFS).
URL: *https ://hal.archives-ouvertes.fr/hal-01360024*

Bas, P., Furon, T., Cayre, F., Doërr, G., Mathon, B. (2016), *Watermarking Security : Fundamentals, Secure Designs and Attacks*, Springer.

Bernard, S., Pevny, T., Bas, P., Klein, J. (2019), Exploiting Adversarial Embeddings for Better Steganography, *in* IH-MMSEC, ACM, Paris (France).

Cogranne, R., Giboulot, Q., Bas, P. (2020), Steganography by Minimizing Statistical Detectability : The cases of JPEG and Color Images., *in* ACM Information Hiding and MultiMedia Security (IH&MMSec), Denver, CO, United States.

Cogranne, R., Sedighi, V., Fridrich, J. (2017), Practical strategies for content-adaptive batch steganography and pooled steganalysis, *in* 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 2122–2126.

Costa, M. (1983), Writing on dirty paper, *IEEE Trans. on Information Theory*, 29(3), 439–441.

Crandall, R. (1998), Some notes on steganography, *Posted on steganography mailing list*, pp. 1–6.

Denemark, T., Fridrich, J. (2015), Side-informed steganography with additive distortion, *in* Information Forensics and Security (WIFS), 2015 IEEE International Workshop on, IEEE, pp. 1–6.

Filler, T., Judas, J., Fridrich, J. (2011), Minimizing additive distortion in steganography using syndrome-trellis codes, *Information Forensics and Security, IEEE Transactions on*, 6(3), 920–935.

Fridrich, J., Goljan, M., Soukal, D. (2004), Perturbed quantization steganography with wet paper codes, *in* Proceedings of the 2004 workshop on Multimedia and security, ACM, pp. 4–15.

24. le transcodage consiste à passer d'un format à autre, il implique le plus souvent une recompression avec perte.

Fridrich, J., Goljan, M., Soukal, D. (2005), Perturbed quantization steganography, *Multimedia Systems*, 11(2), 98–107.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y. (2014), Generative adversarial nets, *in* Advances in Neural Information Processing Systems, pp. 2672–2680.

Guo, L., Ni, J., Su, W., Tang, C., Shi, Y.-Q. (2015), Using statistical image model for jpeg steganography : Uniform embedding revisited, *IEEE Transactions on Information Forensics and Security*, 10(12), 2669–2680.

Holub, V., Fridrich, J., Denemark, T. (2014), Universal distortion function for steganography in an arbitrary domain, *EURASIP Journal on Information Security*, 2014(1), 1–13.

Ker, A. (2007), Batch steganography and pooled steganalysis, *in* Information Hiding, Springer, pp. 265–281.

Ker, A. D., Bas, P., Böhme, R., Cogranne, R., Craver, S., Filler, T., Fridrich, J., Pevnỳ, T. (2013), Moving steganography and steganalysis from the laboratory into the real world, *in* Proceedings of the first ACM workshop on Information hiding and multimedia security, ACM, pp. 45–58.

Ker, A. D., Pevnỳ, T., Kodovskỳ, J., Fridrich, J. (2008), The square root law of steganographic capacity, *in* Proceedings of the 10th ACM workshop on Multimedia and security, ACM, pp. 107–116.

Kouider, S., Chaumont, M., Puech, W. (2013), Adaptive steganography by oracle (aso), *in* Multimedia and Expo (ICME), 2013 IEEE International Conference on, IEEE, pp. 1–6.

Li, B., Wang, M., Huang, J., Li, X. (2014), A new cost function for spatial image steganography, *in* Image Processing (ICIP), 2014 IEEE International Conference on, IEEE, pp. 4206–4210.

Li, B., Wang, M., Li, X., Tan, S., Huang, J. (2015), A strategy of clustering modification directions in spatial image steganography, *Information Forensics and Security, IEEE Transactions on*, 10(9), 1905–1917.

Pevnỳ, T., Kopp, M., Křoustek, J., Ker, A. D. (2016), Malicons : Detecting payload in favicons, *Electronic Imaging*, 2016(8), 1–9.

Pevnỳ, T., Nikolaev, I. (2015), Optimizing pooling function for pooled steganalysis, *in* 2015 IEEE International Workshop on Information Forensics and Security (WIFS), IEEE, pp. 1–6.

Sallee, P. (2003), Model-based steganography, *in* International Workshop on Digital Watermarking (IWDW), LNCS, vol. 2.

S.Dumitrescu, X.Wu, Z.Wang (2003), Detection of LSB steganography via sample pair analysis, *in* IEEE transactions on Signal Processing, pp. 1995–2007.

Sedighi, V., Cogranne, R., Fridrich, J. (2016), Content-adaptive steganography by minimizing statistical detectability, *Information Forensics and Security, IEEE Transactions on*, 11(2), 221–234.

Shannon, C. (1948), A mathematical theory of communication, *Bell System Tech. J.*, 27, 379–423, 623–656.

Simmons, G. (1984), The prisoners' problem and the subliminal channel, *in* Proc. of CRYPTO '83, Plenum Press, pp. 51–67.

Taburet, T., Bas, P., Sawaya, W., Fridrich, J. (2020), 'Natural steganography in jpeg domain with a linear development pipeline'.

Taburet, T., Filstroff, L., Bas, P., Sawaya, W. (2018), An Empirical Study of Steganography and Steganalysis of Color Images in the JPEG Domain, *in* IWDW, International Workshop on Digital Forensics and Watermarking, Jeju, South Korea.
URL: *https ://hal.archives-ouvertes.fr/hal-01904482*

Tang, W., Li, B., Tan, S., Barni, M., Huang, J. (2019), Cnn-based adversarial embedding for image steganography, *IEEE Transactions on Information Forensics and Security*, .

Tang, W., Tan, S., Li, B., Huang, J. (2017), Automatic steganographic distortion learning using a generative adversarial network, *IEEE Signal Processing Letters*, 24(10), 1547–1551.

Wang, Y., Zhang, W., Li, W., Yu, X., Yu, N. (2019), Non-additive cost functions for color image steganography based on inter-channel correlations and differences, *IEEE Transactions on Information Forensics and Security*, .

Yang, J., Ruan, D., Huang, J., Kang, X., Shi, Y.-Q. (2019), An embedding cost learning framework using gan, *IEEE Transactions on Information Forensics and Security*, 15, 839–851.

Zakaria, A., Chaumont, M., Subsol, G. (2019), Pooled steganalysis in jpeg : how to deal with the spreading strategy ?, *arXiv preprint arXiv :1906.11525*, .

7

STÉGANALYSE : détection d'information cachée dans des contenus multimédia

Rémi COGRANNE[1], Marc CHAUMONT[2], Patrick BAS[3]
[1]Université de Technologie de Troyes
[3]LIRMM, Univ Montpellier, CNRS, Univ Nimes
[2]CRIStAL, Lille, CNRS

Dans le chapitre 6 précédent, nous avons présenté la stéganographie, c'est-à-dire les méthodes permettant de dissimuler des informations dans des images numériques. Nous avons notamment insisté sur le fait que les méthodes de stéganographie se construisent dans un contexte où un adversaire cherche à identifier parmi une liste d'images celles servant à véhiculer des informations dissimulées (les images Stégo). Dans ce chapitre, nous allons voir comment procéder à une analyse d'une image numérique en vue d'obtenir des informations sur les données qui peuvent y avoir été cachées.

7.1. Introduction, enjeux et contraintes

Avant de rentrer dans le vif du sujet, nous rappellerons brièvement que le but de la stéganographie est de dissimuler des informations secrètes dans des médias numériques afin que ces derniers demeurent visuellement et statistiquement, « aussi proche

Sécurité multimédia - Partie 1 : authentification et insertion de données cachées, coordonné par William PUECH.

que possible » que des médias originaux. L'exemple du problème de prisonnier[1] (Simmons 1984), permet d'illustrer ce contexte. Le but de la stéganalyse peut sembler, à première vue, assez simple : il s'agit « seulement » de détecter les médias contenant des informations cachées afin d'empêcher leur transmission. En pratique, le sujet est beaucoup plus vaste que cela. En réalité, comme très souvent dans le domaine de la sécurité, le but de la stéganalyse dépend largement du *scénario* considéré et notamment des connaissances *a priori* dont dispose Ève[2].
Nous allons voir dans ce chapitre que la stéganalyse a été très largement étudiée comme outil d'évaluation des méthodes de stéganographie. Cette approche pose un contexte très particulier dans lequel Ève est supposée être *omnisciente* dans le sens où elle peut accéder à (presque) toutes les informations relatives aux données dissimulées. Dans ce scénario nous considérerons généralement qu'Ève ignore uniquement (1) quel est le message dissimulé, bien que sa distribution statistique soit connue, (2) quelle est la clé d'insertion utilisée et, (3) si un message est effectivement inséré. Ce scénario est très pratique pour le stéganographe Alice, car d'une part cela lui permet de se focaliser sur le problème d'intérêt sans tenir compte des « difficultés techniques » annexes. Par ailleurs, dans ce scénario Alice envisage le pire cas en terme de stéganalyse, elle peut donc être certaine que son évaluation est pessimiste et que, en pratique, un adversaire plus réaliste pourra difficilement concevoir une stéganalyse aussi précise car il ne connaît souvent pas la taille du message inséré ou encore l'algorithme utilisé. Nous verrons dans la dernière partie de ce chapitre 7.6, que les conditions expérimentales sont également très éloignées des conditions opérationnelles dans lesquelles la stéganalyse pourrait être réalisée.

7.1.1. *Les différents objectifs de la stéganalyse*

De nombreux travaux ont été proposés pour mettre en œuvre différents types de stéganalyse, nous allons brièvement les décrire dans la suite de cette section.

– D'une part, différentes méthodes de détection peuvent être envisagées selon le média Cover[3] : les images, les sons et les vidéos étant des médias différents et compressés de façon différente, ces derniers devraient être traités différemment pour y déceler des informations cachées. Plus largement la stéganographie dans les textes ou dans les réseaux informatiques sont de nature si différente qu'il apparaît difficile de les analyser des façons analogues.

– D'autres travaux, plus proches de ce qui se fait en cryptanalyse, se focalisent sur la recherche de la clé d'insertion à partir d'images contenant des informations cachées. Ces travaux requièrent souvent des connaissances importantes de la part de la stéganalyste Ève.

---

1. voir également la section 6.1.
2. voir la section 6.1 pour la définition des rôles d'Alice, Bob et Ève
3. voir la section 6.1 pour la définition des termes *Cover et Stégo.*

– Dans la stéganalyse *quantitative* le but est d'estimer la taille d'un possible message dissimulé. Nous comprendrons aisément qu'estimer la taille du message peut être rapproché d'un problème de détection binaire. Une première vision, assez naïve, consiste à estimer la taille du message caché et de décider que le média est Stégo lorsque cette estimation dépasse un seuil donné.

– Enfin nous pouvons mentionner la stéganalyse active dans laquelle Ève modifie légèrement le média transmis afin de préserver son aspect visuel tout en rendant l'extraction du message caché impossible.

Par manque de place, il est difficile d'aborder dans ce seul chapitre l'ensemble des problèmes qui constituent la stéganalyse. Aussi, bien que ces questions sont intéressantes, nous nous en tiendrons uniquement au cadre classique qui s'intéresse à détecter des informations cachées dans les images numériques.

Comme nous l'avons précédemment expliqué, la stéganalyse s'est largement développée pour permettre l'évaluation des méthodes de stéganographie. Aussi, la plupart des travaux dans ce domaine invoquent le principe de Principe de Kerckhoffs (Auguste 1883), reformulé par Claude. E. Shannon comme le fait que « l'adversaire connaît le système utilisé ». En vertu de ce principe, la stéganalyse repose généralement sur le fait que Ève ignore uniquement ce qui concerne le message inséré et la clé d'insertion utilisée. Nous reviendrons plus longuement dans la section 7.3 en détail sur ce scénario de stéganalyse *ciblée*, mais cela suppose qu'Ève, connaît le type de média, généralement une image, l'algorithme d'insertion, la taille du message et ses propriétés statistiques.

7.1.2. *Les différentes méthodes pour faire de la stéganalyse*

Les méthodes de détection d'informations cachées les plus efficaces sont indubitablement celles basées sur des signatures. Dans ce cas il est possible de détecter la stéganographie indirectement en identifiant une propriété particulière qui se retrouve systématiquement lorsque l'insertion d'un message a été effectuée avec un certain outil et uniquement dans ce cas. Il s'agit donc de détecter une spécificité liée à l'utilisation d'un outil précis plus que la présence d'une information cachée elle-même. Cette approche peut être rapprochée de ce qui se fait pour la sécurité informatique (détection d'attaques réseaux, de virus, etc. ...) et est abordée dans la section 7.2 au travers de quelques exemples.

Une seconde famille de méthodes de stéganalyse, plus générale, basée sur la modélisation et la détection statistique est abordée dans la section 7.3. Ce type de méthodes de détection est très complexe à mettre en œuvre en pratique, car elle nécessite de disposer d'un modèle statistique très précis d'une image Cover et d'une image Stégo afin de pouvoir évaluer, *statistiquement* si une image donnée provient plus vraisemblablement d'un de ces deux modèles. Dans cette section nous définissons précisément le

cadre de la stéganalyse *ciblée* qui est nécessaire pour la construction de tels modèles statistiques.

La section 7.4 traite de l'utilisation de méthodes d'apprentissage statistiques. Les méthodes de stéganalyse les plus efficaces reposent sur cette approche et c'est donc naturellement celles qui ont été le plus largement étudiées. La stéganalyse peut être ici subdivisée en deux problèmes distincts : le premier concerne l'extraction de *caractéristiques* pertinentes et le second est d'apprendre automatiquement à classer, sur la base d'un vaste lot d'exemples, les *caractéristiques* provenant d'images Cover et Stégo.

Depuis peu, nous assistons à un développement spectaculaire des méthodes d'apprentissage profond, ou *Deep Learning*, qui, au contraire, procèdent en une seule phase qui regroupe caractérisation et classification. L'utilisation de ces méthodes pour la stéganalyse est abordée dans la section 7.5.

Ce chapitre se clôture, en section 7.6 par une brève liste non exhaustive des sujets qui nous semblent les plus intéressants et qui demeurent largement ouverts.

7.2. Détection par signature incompatible

Dans le domaine de la sécurité informatique, la détection par signature se définit généralement comme une méthode de détection basée sur la recherche de motifs spécifiques, de traces caractéristiques indiquant l'utilisation d'un logiciel ou algorithme particulier. Ici l'aspect logiciel est important, car, une signature ne dépend pas de la méthode d'insertion considérée, mais est bien spécifique à une implémentation donnée. Comme nous le verrons dans cette section, cette signature peut apparaître dans les méta-données du média, ou bien dans les propriétés du signal qui constitue le média.

Un exemple simple est illustré par l'algorithme de stéganographie F5, proposé par A. Westfeld en 2001 (Westfeld 2001). Il est associé à un démonstrateur afin d'en permettre une utilisation au sein de la communauté scientifique. A. Westfeld ne souhaitait pas programmer entièrement la partie compression JPEG et a utilisé un encodeur libre, ... mais ce dernier insérait systématiquement dans l'entête du fichier le commentaire suivant « JPEG ENCODER COPYRIGHT 1998, JAMES R. WEEKS AND BIOELECTROMECH ». Cet encodeur étant en réalité très peu utilisé, il a été possible de détecter les images contenant des informations cachées avec F5 non pas en détectant directement les données dissimulées, mais en détectant ce commentaire très particulier. Ici il s'agit en quelque sorte de détecter une « erreur d'implémentation » et, généralement, un logiciel de stéganographie ne doit pas laisser de « signatures » qui permettent de l'identifier. En ce sens une signature est toujours une faille liée à une implémentation bien précise.

Un second exemple de détection par signature, en utilisant cette fois-ci des propriétés mathématiques, a été présenté dans (Goljan and Fridrich 2015) et concerne les images couleurs. Rappelons que le capteur d'un appareil photographique est par nature insensible à la couleur. Pour révéler la couleur, un micro filtre rouge, vert ou bleu, est placé devant chaque photo-site du capteur qui enregistre alors l'information couleur correspondant à la couleur du filtre (Sharma and Bala 2017). Il faut ensuite « reconstituer les deux couleurs manquantes » ce qui se fait à partir des pixels voisins. Il a été observé dans (Goljan and Fridrich 2015) que la stéganographie peut aller à l'encontre des règles élémentaires de reconstruction de la valeur des couleurs manquantes. Par exemple dans le cadre d'une interpolation linéaire, si la composante verte d'un pixel estimée à partir de ses voisins devient, à cause de la stéganographie, plus importante que tous ses pixels voisins, cette valeur deviendra incompatible avec son voisinage et révélera la stéganographie !
Les auteurs de (Goljan and Fridrich 2015) ont alors proposé d'utiliser 7 caractéristiques qui compte le nombre de pixels dont la valeur n'est pas en accord avec des règles de reconstruction des couleurs à partir des pixels voisins ; ces valeurs sont toujours supposées être nulles pour une image naturelle et permettent très efficacement de détecter par signature la stéganographie dans les images couleurs.

Un autre exemple de signature incompatible est celui d'images Cover codées dans le domaine spatial, mais qui ont été préalablement compressées au format JPEG. Cette stratégie peut sembler intéressante au premier abord car une image non compressée peut théoriquement accueillir une plus grande quantité d'information cachée que sa version compressée. Cependant, si une image codée dans le domaine spatial a été d'abord compressée en utilisant le standard JPEG, les blocs de 8×8 pixels peuvent être décomposés comme une somme des composantes de la base de représentation *DCT* pondérée par des coefficients entiers. Plus exactement notons $\mathbf{X}$ un bloc de 8×8 pixels ce dernier peut s'écrire :

$$\mathbf{X} = \text{round}\left(\sum_{k=0}^{7}\sum_{l=0}^{7} c_{k,l} q_{k,l} \mathbf{D}_{k,l}\right), \qquad [7.1]$$

avec $q_{k,l}$ les éléments de la matrice de quantification, $c_{k,l}$ les coefficients (inconnus) des composantes de la base DCT notés $\mathbf{D}_{k,l}$ et round$(\cdot)$ la fonction d'arrondie.
Si certains pixels de ce bloc $\mathbf{X}$ sont modifiés après compression JPEG, il devient impossible de trouver des coefficients $c_{k,l}$ entiers permettant de représenter le bloc $\mathbf{X}$. Le stéganalyste, Alice, peut alors déduire que cette image semble avoir été compressée au format JPEG, mais que certains pixels vont à l'encontre de ce qui aurait dû être obtenu lors de la décompression. Encore une fois, cette incompatibilité mathématique peut être utilisée pour détecter une image Stégo.

Plus récemment, une signature assez similaire, basée également sur ce qui a été appelé la « compatibilité avec la compression JPEG » a été proposée dans (Butora and

Fridrich 2020 ; Cogranne 2020*a*). Il s'agit dans ce cas de détecter la stéganographie dans les images compressées au format JPEG en utilisant le fait que modifier les coefficients DCT peut amener à produire des valeur de pixels qui ne sont pas possible pour une image naturelle. Cette méthode de détection utilise une stratégie de stéganalyse statistique et est décrite plus en détail en fin de section 7.3.

Notons enfin que si les méthodes de détection par signature sont en général très fiables dans le sens où les erreurs sont peu fréquentes, chaque logiciel ou algorithme doit être néanmoins minutieusement analysée afin de trouver des traces caractéristiques qui relèvent son utilisation. Ce type d'analyse est donc très chronophage et tout en étant peu généralisable. Les stratégies de stéganalyse qui suivent se veulent plus génériques.

7.3. Détection par méthodes statistiques

Les méthodes de détection qui sont présentées dans la suite de ce chapitre sont souvent moins fiables que les méthodes de détection par signature, mais présentent l'avantage d'être beaucoup plus générales dans le sens où elles visent à détecter les modifications liées à la dissimulation d'information dans le contenu même d'un média.

Nous introduisons dans un premier temps les méthodes statistiques simples. Considérons une image $\mathbf{X}$ de taille $M \times N$, décrite comme une matrice de pixels codés sur 8 bits $x_{m,n} \in \{0; \ldots; 255\}$, la stéganalyse consiste alors à choisir entre les deux hypothèses suivantes :

1) $\mathcal{H}_0$: les pixels $x_{m,n}$ sont issus d'une image Cover,

2) $\mathcal{H}_0$: les pixels $x_{m,n}$ sont issus d'une image Stégo.

La principale difficulté est alors de définir précisément ce qui caractérise statistiquement une image Cover et ce qui la différencie d'une image Stégo.

7.3.1. *Test statistique du χ^2*

Historiquement, la première approche pour aborder ce problème a été proposée dans (Westfeld and Pfitzmann 1999). Ne pouvant décrire statistiquement ce qu'est une image naturelle, il a été proposé de modéliser les pixels après utilisation de la stéganographie. Pour expliquer le fonctionnement de ce test, il nous faut rappeler comment procède l'insertion d'un message par substitution LSB[4]). Pour insérer dans le pixel

4. voir également la section 6.3.1 du chapitre précédant.

$x_{m,n}$ le i-ième bit du message $m_i \in \{0;1\}$ la stéganographie par substitution des bits de poids faibles modifie le pixel de la façon suivante :

$$z_{m,n} = x_{m,n} - \text{LSB}(x_{m,n}) + m_i, \qquad [7.2]$$

avec $z_{m,n}$ le pixel de l'image Stégo et $\text{LSB}(x_{m,n})$ le bit de poids faible de $x_{m,n}$. Il est généralement admit en stéganalyse que le message inséré $\mathbf{m} = (m_1, \ldots, m_I)$ est une suite de bits tous indépendants et identiquement distribués (i.i.d) suivant une loi uniforme : $\mathbb{P}[m_i = 0] = \mathbb{P}[m_i = 1] = 1/2$.
La conséquence de l'insertion du bit m_i dans le pixel $x_{m,n}$ est que le pixel Stégo peut se modéliser par la distribution de probabilité suivante :

$$\mathbb{P}[z_{m,n} = x_{m,n}] = \frac{1}{2} = \mathbb{P}[z_{m,n} = \bar{x}_{m,n}], \qquad [7.3]$$

avec $\bar{x} = x + (-1)^x$ la valeur entière x dont le bit de poids faible a été inversé.
En supposant, en outre, que la totalité des pixels est utilisée pour dissimuler un message secret (très) volumineux, la distribution des pixels Stégo peut être modélisée de la façon suivante :

$$\forall k \in \{0; \ldots; 255\}, \forall (m,n), \mathbb{P}[z_{m,n} = k] = \mathbb{P}\left[z_{m,n} = \bar{k}\right]. \qquad [7.4]$$

Il devient alors possible de tester statistiquement si une image est Stégo en mesurant la différence entre la distribution théorique de l'équation [7.4], et celle observée sur une image analysée. C'est justement l'objet du test du χ^2 qui mesure la différence entre une distribution théorique et celles des observations de la façon suivante :

$$\chi^2 = \sum_{k=0}^{255} \frac{(N_k - N_k^\star)^2}{N_k^\star} \quad \text{avec} \quad N_k^\star = \frac{N_k + N_{\bar{k}}}{2} \qquad [7.5]$$

où N_k représente le nombre de pixels dont la valeur est k et $N_k^\star$ représente le nombre attendu de pixels avec la valeur k. L'équation [7.5] représente, en effet, une mesure de la différence entre distribution théorique et empirique au travers du terme : $(N_k - N_k^\star)^2$. La figure 7.1 illustre les modèles de distribution de l'équation [7.4] et de l'équation [7.5] pour une image Stégo ; on constate en effet un nombre de pixels k et $\bar{k}$ qui se trouvent "égalisés" sur l'image Stégo.

Lorsque la valeur du χ^2 est supérieure à un seuil donné τ, l'image est jugée comme étant Cover, ou statistiquement trop différente de la distribution théorique pour être jugée comme Stégo [7.4]. Comment faut-il alors fixer le seuil τ ? Les auteurs dans (Westfeld and Pfitzmann 1999) proposent de choisir ce seuil afin d'assurer

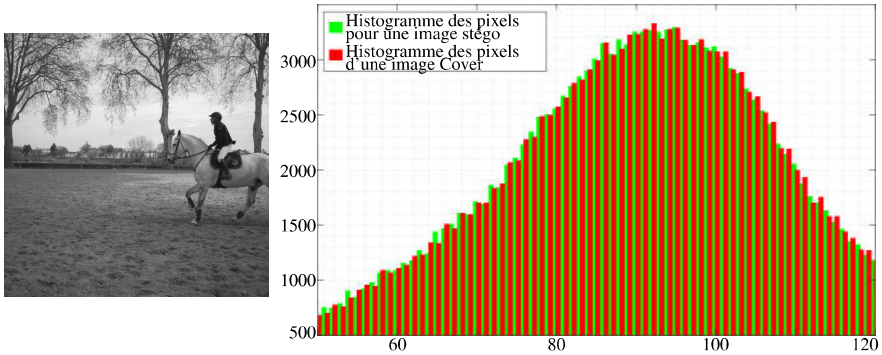


Figure 7.1 – Illustration de l'impact de la stéganographie par LSBR et sa détection par le test du χ^2.

que, théoriquement, une image Stégo soit jugée comme naturelle avec une probabilité[5] p_0. Pour cela les auteurs utilisent la distribution du χ^2 qui permet de calculer cette probabilité avec la relation suivante :

$$p_{\chi^2}(\tau) = \mathbb{P}\left[\chi^2 > \tau\right] = 1 - \int_0^{\tau} \frac{t^{(\nu-2)/2} e^{-t/2}}{2^{\nu-2}\Gamma(\nu)} dt, \qquad [7.6]$$

avec ici $\nu = 128 - 1$ le « nombre de degrés de liberté » défini par le nombre de « paires de valeurs » qui peuvent être permutées entre elles.

La figure 7.2 illustre cette façon de fixer le seuil de décision, en fonction d'une probabilité p_0 préalablement fixée de « faux-négatif ». Elle permet également de comparer la distribution théorique de la statistique χ^2 de l'équation [7.5] avec la distribution théorique du χ^2. La différence entre les observations et le modèle théorique est principalement dû au fait que la très grande majorité des images ne possèdent pas des pixels avec tous les valeurs entre 0 et 255 et ont donc un nombre de degré de liberté inférieur à $\nu = 128 - 1$.

La force du test dû χ^2 est de proposer un test statistique sans avoir à résoudre le problème (très délicat) de modéliser la distribution statistique des pixels d'une image Cover. En effet le test propose essentiellement de vérifier si les pixels d'une image analysée répondent au modèle d'une image Stégo ; dans le cas contraire l'image est jugée, par défaut, comme étant Cover. Un autre intérêt de ce test est de tenter de fixer le seuil de détection τ sur la base d'une probabilité de détection manquée $p_{\chi^2}(\tau)$, définie dans l'équation [7.6].

5. probabilité dite de « détection manquée » ou de « faux-négatif ».

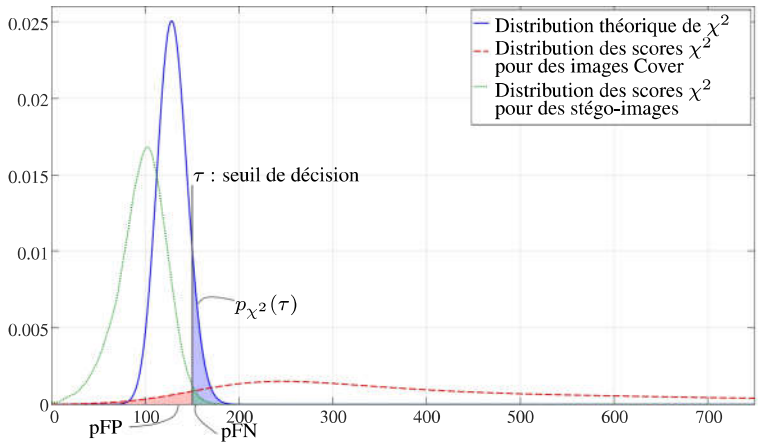


Figure 7.2 – Illustration des distributions de probabilités (empiriques et théoriques) pour le résultats du test dû χ^2 et probabilité d'erreur qui en résultent.

Malheureusement, ce test n'est pas très performant, notamment, car il ne modélise pas la distribution statistique d'une image Cover mais uniquement celle de l'image Stégo. De façon générale en détection statistique, lorsque seule une des deux hypothèses peut être caractérisée, nous procédons à un test du type « goodness-of-fit », adéquation des observations à ce modèle et cela est généralement moins fiable que quand il est possible de caractériser exactement les deux hypothèses en concurrence, comme proposé par le test du rapport de vraisemblance dans la section suivante.

7.3.2. *Test du rapport de vraisemblance*

Afin de bien comprendre l'approche présentée dans cette section, il est nécessaire de formaliser la stéganalyse et la détection statistique. Un test statistique est une fonction δ qui, à partir d'un ensemble d'observations $\mathbf{X}$ retourne une décision binaire : $\delta : \mathbf{X} \rightarrow \{0, 1\}$ de sorte que l'hypothèse $\mathcal{H}_0$ est acceptée si $\delta(\mathbf{X}) = 0$. Rappelons brièvement qu'un test n'est jamais parfait et que deux types d'erreurs sont possibles : faux-positif et faux-négatif (ou fausse-alarme et non-détection). En général, l'hypothèse $\mathcal{H}_0$ est dite « nulle » et correspond au cas « normal » alors que l'hypothèse alternative $\mathcal{H}_1$ correspond à la situation problématique qu'il est souhaité détecter. Le faux-positif correspond donc au cas où le test décide d'accepter l'hypothèse $\mathcal{H}_1$ alors qu'en vérité les observations sont issues de l'hypothèse $\mathcal{H}_0$.

Dans le cas qui nous intéresse, l'image analysée est une image Cover qui est classée comme Stégo. À l'inverse, un faux-négatif correspond au cas où le test accepte l'hypothèse $\mathcal{H}_0$ alors que les observations sont réellement issues de l'hypothèse alternative $\mathcal{H}_1$; pour la stéganalyse, c'est le cas où la gardienne Ève manque la détection d'une image Stégo.

Résultat \ Vérité	Hypothèse 0 : (image Cover)	Hypothèse 1 : (stégo-image)
Accepter l'hypothèse 0	Décision correcte	Faux-négatif (détection manquée) pFN : $1-\varsigma$)
Accepter l'hypothèse 1	Faux-positif (pFP : α=$\mathbb{P}\left[\delta(\mathbf{X})=\mathcal{H}_1|\mathcal{H}_0\right]$)	Décision correcte (ς=$\mathbb{P}\left[\delta(\mathbf{X})=\mathcal{H}_1|\mathcal{H}_1\right]$)

Figure 7.3 – Les différentes possibilités de bonnes et de mauvaises détection.

Afin de montrer la faiblesse du test du χ^2 nous devons disposer d'un modèle statistique des images Cover ; ce dernier est construit en supposant les pixels sont statistiquement indépendants et tous issus d'une distribution Gaussienne :

$$x_{m,n} \sim \mathcal{N}\left(\mu_{m,n}, \sigma^2_{m,n}\right), \qquad [7.7]$$

où $\mu_{m,n}$ représente l'espérance mathématique du pixel (*i.e.* sa « moyenne » ou sa valeur théorique) et $\sigma^2_{m,n}$ représente la variance du bruit. C'est un modèle couramment utilisé en traitement des images qui suppose qu'une image peut être décomposée sous la forme d'un « contenu » théorique et d'un bruit lié aux imperfections de l'appareil photographique. Pour être plus précis, les pixels sont représentés par des valeurs entières et, par souci de simplicité, nous supposons que cela n'a pas d'impact sur la distribution de probabilité des pixels :

$$\mathbf{p}_0 = \mathbb{P}\left[x_{m,n}{=}k\right] = \left(p_0(k)\right)_{k\in\mathcal{Z}} \propto \left(\frac{1}{\sigma_{m,n}\sqrt{2\pi}} \exp\left(\frac{(k-\mu_{m,n})^2}{2\sigma^2_{m,n}}\right)\right)_{k\in\mathbb{Z}}. \qquad [7.8]$$

Dans un premier temps, nous étudions le cas où Alice utilise la stéganographie par substitution LSB[6]. Chaque pixel peut être modifié avec la même probabilité de $\frac{\beta}{2} = \frac{I}{2MN}$ qui correspond au ratio du nombre de bits du message inséré (I) par pixel (MN), la distribution d'un pixel Stégo devient alors :

$$p_1(k) = \mathbb{P}\left[x_{m,n} = k|\mathcal{H}_1\right] = \left(1-\frac{\beta}{2}\right)p_0(k) + \frac{\beta}{2}p_0(\bar{k}). \qquad [7.9]$$

Comment utiliser ces modèles statistiques pour décider si une image inspectée $\mathbf{X}$ provient plutôt du modèle de l'hypothèse $\mathcal{H}_0$[7], ou de l'hypothèse $\mathcal{H}_1$[8] ?

6. voir également la section 6.3.1.
7. définie dans l'équation [7.8].
8. définie dans l'équation [7.9].

Il existe plusieurs réponses qui ont en commun un élément central, le rapport de vraisemblance (RV) qui s'exprime comme :

$$\Lambda(\mathbf{X}) = \frac{p_1(\mathbf{X})}{p_0(\mathbf{X})} = \prod_{m,n} \frac{p_1(x_{m,n})}{p_0(x_{m,n})}, \qquad [7.10]$$

avec p_0 et p_1 sont respectivement les modèles statistiques de distribution pour des images Cover et Stégo, définis respectivement par les équations [7.8] et [7.9]. La seconde partie de l'égalité de l'équation [7.10] découle directement du modèle d'indépendance statistique entre les pixels qui, même si cela n'est pas totalement exact, est très largement utilisé, car il simplifie beaucoup les choses.

Clairement, le rapport de vraisemblance est d'autant plus grand que la probabilité d'observer les données à analyser est plus grande sous $\mathcal{H}_1$ que sous $\mathcal{H}_0$; à l'inverse, si la probabilité d'observer ces données est beaucoup plus grande sous $\mathcal{H}_0$ que sous $\mathcal{H}_1$, le rapport de vraisemblance [7.10] sera faible. Basé sur cette observation, le test du rapport de vraisemblance consiste simplement à seuiller ce rapport de vraisemblance : $\delta(\mathbf{X}) = \mathcal{H}_0$ si $\Lambda(\mathbf{X}) < \tau$ et $\delta(\mathbf{X}) = \mathcal{H}_1$ si $\Lambda(\mathbf{X}) \geq \tau$.

L'utilisation du test du rapport de vraisemblance se justifie théoriquement car le Lemme de Neyman-Pearson stipule qu'il permet d'atteindre la plus grande puissance[9] ς pour une probabilité de faux-positif α fixée à $\alpha = \mathbb{P}\left[\Lambda(\mathbf{X}) > \tau\right]$. Cette dernière relation permet également de fixer le seuil afin de respecter un taux de faux-positif préalablement établi.

En utilisant le modèle des images Cover de l'équation [7.8] et le modèle des images Stégo de l'équation [7.9], le rapport de vraisemblance (RV) s'écrit :

$$\Lambda(\mathbf{X}) = \prod_{m,n} \left[(1-\beta) + \beta \frac{p_0(x_{m,n}) + p_0(\bar{x}_{m,n})}{2p_0(x_{m,n})} \right]. \qquad [7.11]$$

Par souci de simplicité, le logarithme du RV est généralement utilisé, cela afin de remplacer le produit, dans l'équation [7.11], par une somme. Par ailleurs, en utilisant la définition de $p_0(x_{m,n})$ [7.8], un calcul (un peu fastidieux) permet de simplifier la relation précédente comme suit (Cogranne 2011) :

$$\log\big(\Lambda(\mathbf{X})\big) = \sum_{m,n} \beta \log\big(\Lambda(x_{m,n})\big) = \sum_{m,n} \beta \frac{(x_{m,n} - \mu_{m,n})(x_{m,n} - \bar{x}_{m,n})}{2\sigma^2_{m,n}}. \qquad [7.12]$$

9. voir définition dans la figure 7.3.

Dans cette approche, le plus intéressant n'est pas tant de disposer d'une relation simple permettant de calculer le rapport de vraisemblance (bien que quelques simplifications aient dû être faites), mais davantage de pouvoir caractériser la distribution statistique de ce dernier et donc, *in fine*, de maîtriser les probabilités de fausses-alarmes et de non-détection. En particulier, il est possible de montrer que pour une image Cover, l'espérance mathématique et la variance du terme $\log\big(\Lambda(\mathbf{X})\big)$ sont données par :

$$\mathbb{E}\left[\log\big(\Lambda(\mathbf{X})\big)|\mathcal{H}_0\right] = 0 \quad \text{et} \quad \mathbb{V}ar\left[\log\big(\Lambda(\mathbf{X})\big)|\mathcal{H}_0\right] = \frac{\beta^2}{4\sigma_{m,n}^2}. \tag{7.13}$$

Il est ensuite possible de normaliser le RV comme suit :

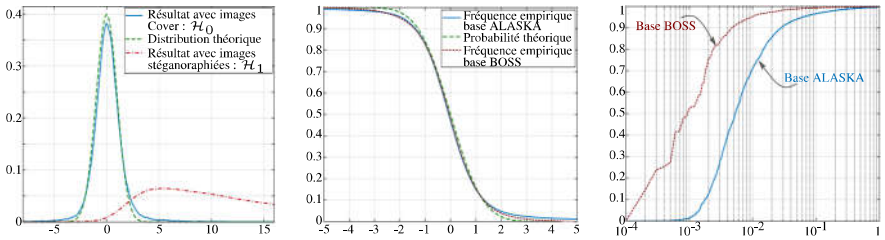
$$\log\left(\Lambda(\mathbf{X})\right) = \frac{1}{\varrho}\sum_{m,n}\log\left(\Lambda(x_{m,n})\right) \quad \text{avec} \quad \varrho = \left(\sum_{m,n}\frac{\beta^2}{4\sigma_{m,n}^2}\right)^{1/2}, \tag{7.14}$$

de sorte que, en vertu du théorème de la limite centrale (Lehmann and Romano 2005, Théorème 11.2.5) il est possible de calculer la distribution statistique du log-RV pour une image donnée :

$$\begin{cases} \text{sous } \mathcal{H}_0 : & \log\left(\Lambda(\mathbf{X})\right) \sim \mathcal{N}\left(0,1\right), \\ \text{sous } \mathcal{H}_1 : & \log\left(\Lambda(\mathbf{X})\right) \sim \mathcal{N}\left(\varrho,1\right). \end{cases} \tag{7.15}$$

La distribution Gaussienne étant assez simple à « manipuler » nous pouvons aisément calculer le seuil de décision τ afin d'assurer une probabilité de faux-positif α_0 fixée : $\tau(\alpha_0) = \Phi^{-1}\left(1-\alpha_0\right)$ de sorte que $\mathbb{P}\left[\log\left(\Lambda(\mathbf{X})\right) > \tau(\alpha_0)|\mathcal{H}_0\right] = \alpha_0$. De même la puissance du test ς peut se calculer comme la probabilité $\mathbb{P}\left[\log\left(\Lambda(\mathbf{X})\right) > \tau|\mathcal{H}_0\right] = 1 - \Phi\left(\tau - \varrho\right)$.

De nombreuses remarques sont nécessaires pour comprendre ces différents résultats. D'une part, notons qu'il est ici très « spécifique » que l'espérance mathématique du RV pour une image Stégo soit égale à sa variance. C'est pour cette raison que le facteur de normalisation ϱ dans l'équation [7.14] correspond également à l'espérance dans l'équation [7.15].
Notons également que, tous les calculs précédents nécessitent de connaître l'espérance mathématique $\mu_{m,n}$ et la variance $\sigma_{m,n}^2$ de chacun des pixels, or, en pratique, ces variables sont inconnues. Il est donc proposé de les remplacer par des estimations, ces estimations sont assez délicates à obtenir de façon « précise », et c'est là que l'application de cette approche devient beaucoup plus compliquée et notamment que la garantie des probabilités d'erreur devient très difficile. Quelques résultats sont présentés dans les figures 7.4 en utilisant la base d'images ALASKA (Cogranne *et al*. 2019).



(a) Distribution du log-RV pour des images Cover et Stégo.

(b) Proba. de faux-positif (PFA) α en fonction du seuil de décision ; théorie et résultats empiriques.

(c) Courbe COR représentant la puissance ς en fonction de la PFA α.

Figure 7.4 – Résultats de l'application du test du RVG [7.14]

La figure 7.4(a) montre que la distribution du log-RV obtenue pour des images Cover est plutôt cohérente avec la théorie. Pour les images Stégo, cela dépend du facteur ϱ qui varie pour chaque image créant cet « étalement ». La figure 7.4(b) compare la probabilité de faux-positif en fonction du seuil de décision τ en théorie et en pratique en utilisant deux bases d'images différentes. Nous pouvons également constater que cette propriété est valide pour des probabilités « assez élevées ». Cependant notons que dans le cas où nous souhaiterions obtenir de très faibles taux de faux-positif, les estimations ne sont pas assez précises pour offrir des garanties pertinentes. Enfin, la figure 7.4(c) montre les performances obtenues au travers d'une courbe COR (Caractéristiques Opérationelles de Réception) qui présente la probabilité de faux-positif $\alpha_0(\tau)$ en fonction de la puissance de détection $\varsigma(\tau)$. Nous pouvons observer que pour un taux d'insertion $\beta \approx 0.09$ (24 kilobits insérés dans des images de 512x512 pixels) les performances sont plutôt très bonnes sur les deux bases utilisées pour des probabilités d'erreur importantes.

7.3.3. *Détection de correspondance de LSB*

Fondamentalement, les détecteurs présentés dans la section 7.3.2 montrent que la stéganographie par substitution des bits de poids faibles (LSB) introduit clairement un biais en incrémentant les valeurs paires et en décrémentant les valeurs impaires. Cela explique pourquoi cette méthode d'insertion est clairement à éviter au profit de la correspondance (LSBM ou LSB$\pm$1), qui modifie les bits de poids faibles selon la régle d'insertion définie dans la section 6.3.1 du chapitre précédant.

Après avoir présenté avec détail l'application d'un test statistique pour la stéganalyse, nous pouvons aborder la détection de LSB$\pm$1 qui a été largement moins étudiée dans la littérature au travers de tests « simples ». Par ailleurs, nous considérerons également le cas d'un schéma de stéganographie adaptatif, c'est-à-dire que la probabilité

d'utiliser le pixel $x_{m,n}$ est potentiellement différente pour chaque pixel[10] et sera donc notée $\beta_{m,n}$. Le lecteur notera qu'il est très facile de remplacer le « taux d'insertion moyen » β par un taux d'insertion pour chaque pixel $\beta_{m,n}$ sans aucune modification supplémentaire dans les équations [7.9] et [7.11]-[7.14].

Nous ne détaillerons pas tous les détails des calculs qui peuvent être trouvés dans, (Cogranne 2011 ; Sedighi, Cogranne and Fridrich 2016), mais le calcul du log-RV pour la correspondance de bit de poids faibles amène, après quelques simplifications, à :

$$\Lambda^{\pm}(\mathbf{X}) = \sum_{m,n} \beta_{m,n} \Lambda^{\pm}(x_{m,n}) = \sum_{m,n} \beta_{m,n} \left(\frac{(x_{m,n} - \mu_{m,n})^2 - 1/12}{\sigma^4_{m,n}} - \frac{1}{\sigma^2_{m,n}} \right) . \quad [7.16]$$

Là encore, le plus intéressant est de calculer les probabilités d'erreurs de ce test et, en utilisant comme précédemment le théorème de la limite centrale, cela nécessite de connaître les deux premiers moments qui sont donnés par :

$$\begin{aligned} &\mathbb{E}\left[\Lambda^{\pm}(x_{m,n})|\mathcal{H}_0\right] = 0 \ , \ \mathbb{E}\left[\Lambda^{\pm}(x_{m,n})|\mathcal{H}_1\right] = \tfrac{2\beta^2_{m,n}}{\sigma^4_{m,n}}, \\ &\mathbb{V}ar\left[\Lambda^{\pm}(x_{m,n})\right] = \tfrac{2\beta^2_{m,n}}{\sigma^4_{m,n}}. \end{aligned} \quad [7.17]$$

La comparaison entre le RV pour la détection substitution de LSB, donné dans l'équation [7.12], et le RV pour la détection de LSB$\pm$1, donné dans l'équation [7.16], permet de constater que la détection de LSBR (substitution) repose essentiellement sur un écart à l'espérance, au travers du terme $(x_{m,n}\mu_{m,n})$; à l'inverse la détection de LSB$\pm$1 repose essentiellement sur un écart entre la variance théorique et celle observée au travers du terme $(x_{m,n} - \mu_{m,n})^2$. Or, si estimer l'espérance mathématique des pixels est largement étudié[11], l'estimation précise de la variance des pixels est beaucoup plus difficile et a été nettement moins étudiée. En outre, nous souhaitons détecter un écart à une variance qui n'est pas connue et doit être estimée, et cela indépendamment de la présence d'informations cachées. Tout ceci explique pourquoi les méthodes de stéganalyse statistique s'attaquant aux méthodes d'insertion adaptatives ne sont pas aussi performantes que la stéganalyse sur la substitution LSB vue dans la section précédente.

Si l'application de la détection statistique explique pourquoi la stéganalyse de LSB$\pm$1 est plus difficile, il faut noter que l'étude des performances montre également que pour un pixel donné, la « détectabilité » dépend essentiellement du « rapport insertion-sur-bruit » définit par $\beta^2_{m,n}/\sigma^4_{m,n}$ dans les relations [7.17].

---

10. et cela via l'utilisation d'un coût d'insertion voir les sections 6.2.3 et 6.3.2 du chapitre précédant.

11. c'est un problème de « débruitage » qui a un intérêt majeur pour l'amélioration des images.

Une application intéressante de la théorie des tests d'hypothèses a été d'utiliser ce résultat concernant la « détectabilité statistique » de chacun des pixels afin de concevoir un algorithme d'insertion (Sedighi, Cogranne and Fridrich 2016) qui, au lieu de minimiser une distorsion heuristique, minimise la détectabilité théorique. Bien que cela nécessite d'estimer l'espérance mathématique et la variance de chaque pixel (qui reste un problème ouvert), cela a montré son efficacité. Plus de détails sur cette applications en stéganographie de la théorie des test d'hypothèses sont données dans la section 6.3.2.2 du chapitre précédant.

Pour conclure cette section sur la stéganalyse statistique, mentionnons qu'il a été récemment proposé une méthode de stéganalyse statistique sur les images JPEG qui brille par son efficacité. Cette méthode, présentée par (Butora and Fridrich 2020 ; Cogranne 2020*a*) exploite le fait que les pixels sont quantifiés avant utilisation de la transformée en cosinus discrète (DCT). Pour une image compressée, il est donc possible de la décompresser[12] et de mesurer la variance du bruit de quantification par :

$$\frac{1}{MN}\left\|\mathbf{X}-\text{round}\big(\mathbf{X}\big)\right\|_2^2=\frac{1}{MN}\sum_{m,n}\big(x_{m,n}-\text{round}\big(x_{m,n}\big)\big)^2\,, \qquad [7.18]$$

avec M et N le nombre de lignes et de colonnes de l'image $\mathbf{X}$, round$\big(\,\cdot\,\big)$ la fonction d'arrondie à la plus proche valeur entière.
Si des informations ont été cachées dans les coefficients DCT, l'erreur d'arrondie dans le domaine spatial la statistique définie dans l'équation [7.18] aura tendance a augmenter. Ce test est en fait très efficace (il a été légèrement amélioré dans (Cogranne 2020*b*)) avec une détection de quelques centaines de bits « quasi-parfaite »), car il repose sur un modèle très précis et qui ne dépend pas de paramètres à estimer mais uniquement du bruit de quantification qui, lui, est le même quel que soit l'image analysée. Cependant, il s'agit d'un cas très spécifique qui peut être assimilé à de la détection par signature incompatible (voir section 7.2).

7.4. Détection par apprentissage supervisé

Nous allons à présent décrire des approches de stéganalyse qui sont radicalement différentes de ce qui a été précédemment décrit. Ces méthodes ne reposent donc ni sur la présence d'une signature incompatible, ni la connaissance à priori d'un modèle statistique de l'image Cover et/ou Stégo, mais reposent sur des méthodes d'apprentissage statistique supervisées dont le fondement peut être décomposé en deux phases. Dans un premier temps il s'agit d'extraire des "caractéristiques" à partir des objets de

12. c'est-à-dire de recalculer la valeur des pixels à partir des coefficients DCT quantifiés.

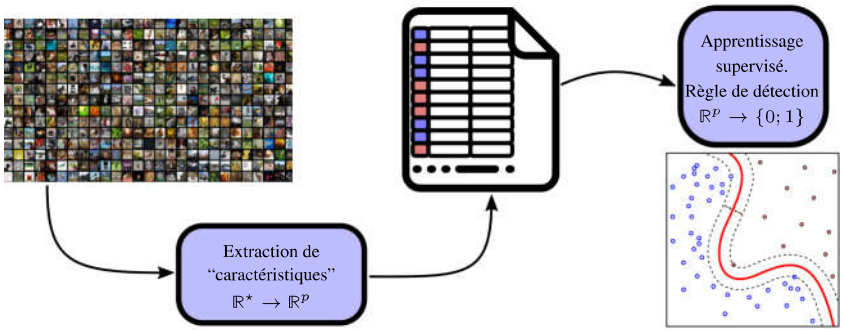


Figure 7.5 – Illustration du principe "*l'apprentissage supervisé*" qui vise à déterminer une règle de détection à partir d'une base de données labélisée (image de droite sous licence "CC BY-SA 4.0", réalisée par Zirguezi).

la base considérée. Ces caractéristiques doivent être révélatrices de la présence d'informations cachées (ou de ce qu'il souhaitable de détecter en général) et permettent de réduire un objet complexe et variable comme une image, de taille quelconque, à un vecteur de p valeurs réelles. Sur la base de ces caractéristiques, une méthode d'apprentissage est utilisée pour déterminer une règle de décision qui fournira un résultat binaire qui sera le résultat de stéganalyse. Cet apprentissage est supervisé dans le sens où chaque image est associée durant cette phase d'apprentissage à un label qui indique si l'image est Cover ou Stégo. Déterminer une règle de décision correspond en fait à la résolution d'un problème d'optimisation qui cherche à associer à chaque vecteur de caractéristiques une valeur aussi proche que possible du label à prédire.

Nous allons brièvement décrire dans cette section comment ces deux phases (1) d'extraction de caractéristiques et (2) d'apprentissage supervisé sont mises en œuvre usuellement en stéganographie.

7.4.1. *Extraction de caractéristiques dans le domaine spatial*

7.4.1.1. *Caractéristiques SPAM*

Il serait évidemment trop long de décrire en détail l'évolution de la stéganalyse pour aboutir aux techniques actuelles, mais notons globalement qu'une pierre angulaire des méthodes actuelles trouve son origine dans (Pevný *et al.* 2010) qui exploite des différences entre pixels adjacents que nous noterons $\mathbf{D}^{\rightarrow}$:

$$d^{\rightarrow}_{m,n} = x_{m,n} - x_{m,n+1}, \qquad [7.19]$$

avec ici $\rightarrow$ qui représente la direction horizontale dans laquelle les différences sont calculées, parmi les 8 possibles et utilisées $\{\leftarrow, \rightarrow, \uparrow, \downarrow, \nwarrow, \searrow, \nearrow, \swarrow\}$.

Sur la base de ces différences, il est proposé d'estimer la fréquence avec laquelle des différences successives se trouvent présentes dans une image. Représenter une fréquence empirique correspond à un dénombrement généralement appelé, par abus de langage, "histogramme" en stéganalyse. Ces dénombrements sont arrangés en vecteur :

$$f^{\rightarrow}_{k,l} = \sum_{m,n} \mathbb{1}\left[d^{\rightarrow}_{m,n} == k, d^{\rightarrow}_{m,n+1} == l\right] \ , \ (k,l) \in \{-T, \ldots, T\}\,, \qquad [7.20]$$

avec $\mathbb{1}[\cdot]$ la fonction indicatrice $\mathbb{1}[e] = 1$ si l'événement e est vraie et $\mathbb{1}[c] = 0$ dans le cas contraire et T le seuil de différence maximale considérée.

Compte tenu que l'on dénombre les valeurs de différences adjacentes conjointement sur des positions voisines, ces vecteurs sont appelés "co-occurrence" en stéganalyse. Ce concept peut être généralisé en utilisant plus que deux valeurs disjointes au prix d'une augmentation du nombre de valeurs de co-occurence possibles. En effet, en utilisant des co-occurrences de c différences adjacentes comprises entre $-T$ et T, cela donne $(2T+1)^c$ valeurs distinctes de co-occurrences possibles. Enfin, une dernière étape proposée dans (Pevný *et al.* 2010) vise à limiter le nombre de caractéristiques en regroupant les valeurs calculées pour des directions opposées, par exemple $\leftarrow$ et $\rightarrow$ ou encore $\nwarrow$ et $\searrow$. Les auteurs proposent de regrouper directions "diagonales" et les directions horizontale et verticale :

$$f_{k,l} = \frac{1}{4}\left(f^{\rightarrow}_{k,l} + f^{\leftarrow}_{k,l} + f^{\uparrow}_{k,l} + f^{\downarrow}_{k,l}\right). \qquad [7.21]$$

Cette étape, appelée "symétrisation", a pour but de réduire le nombre de caractéristiques.

Les caractéristiques originales SPAM "subtractive pixel adjacency matrix" (Pevný *et al.* 2010), utilisent des triplés de 3 différences adjacentes, $c = 3$, $f_{k,l,m}$ (dites de second ordre) et 3 valeurs distinctes pour chacune $T = 3$. Cela revient à dénombrer un nombre distinct de " co-occurrence" de $(3 \times 2 + 1)^3 = 7^3 = 343$. En ajoutant le fait que les co-occurrences diagonales sont distinguées des co-occurrences horizontales et verticales durant la phase de symétrisation, cela donne un total de 686 caractéristiques.
Clairement il s'agit là d'un nombre de caractéristiques (également appelées dimensions) plutôt élevées par rapport aux problèmes généralement étudiés dans le domaine de l'apprentissage statistique.

7.4.1.2. *Caractéristiques SRM*

Cependant, il a été empiriquement constaté que, pour le cas particulier de la stéganalyse qui constitue essentiellement une détection d'un signal très faible au sein du

média Cover), généraliser une telle approche permet d'améliorer la performance de détection. Ainsi, l'approche que nous avons brièvement décrite a été largement développée par la suite dans une méthodologie dont le principe de fonctionnement est illustré dans la figure 7.6. Cette figure illustre le fonctionnement des "modèles spatiaux riches" proposés dans (Fridrich and Kodovskỳ 2012). Il s'agit essentiellement d'un enrichissement de la méthode proposée dans (Pevný *et al.* 2010). Ces modèles sont en outre assez standards et illustrent eux aussi les méthodes actuelles de stéganalyse via la décomposition en 4 grandes étapes qui sont (1) le calcul de résidus, (2) la quantification et le seuillage, (3) le dénombrement de co-occurrence, et enfin (4) la réduction de redondances par symétrisation.

Le calcul de résidus se fait par généralement par utilisation d'un filtre linéaire. À partir d'une image $\mathbf{X}$ dont les pixels sont donnés par $x_{m,n}$, nous appliquons la même relation entre pixels adjacents sur l'ensemble de l'image :

$$r^{(n)}_{m,n} = \sum_{i,j} x_{m+i,n+j} k^{(n)}_{i,j}. \qquad [7.22]$$

Cette somme pondérée de pixels voisins est une opération de convolution $\mathbf{R}^{(n)} = \mathbf{X} \star \mathbf{K}^{(n)}$ dont le noyau $\mathbf{K}^{(n)}$ spécifie la valeur de ces facteurs de pondération. Nous parlons généralement de filtre passe-bas lorsque $\sum_{i,j} k_{i,j} = 1$, il s'agit typiquement d'un "lissage" visant à diminuer le "bruit". En stéganalyse au contraire on parlera de "résidus" pour caractériser des filtrages passe-haut caractérisés par $\sum_{i,j} k_{i,j} = 0$, $k_{0,0} = -1$. En d'autres termes il s'agit "d'estimer" la valeur d'un pixel $x_{m,n}$ à partir de ces voisins $\widehat{x}^{(n)}_{m,n} = \sum_{(i,j)\neq(0,0)} x_{m+i,n+j} k^{(n)}_{i,j}$ puis de calculer la différence $r^{(n)}_{m,n} = \widehat{x}^{(n)}_{m,n} - x_{m,n}$ qui correspond à "l'erreur d'estimation" due au filtre $\mathbf{K}^{(n)}$.

Afin de pouvoir capturer toutes les traces possibles dues à la stéganographie, le nombre de filtres est important ; ainsi 78 résidus sont utilisés pour construire les caractéristiques SRM présentées dans (Fridrich and Kodovskỳ 2012).

La seconde étape est la quantification et le seuillage. De façon générale, les facteurs de pondération utilisés dans les noyaux $\mathbf{K}^{(n)}$ ne sont pas des entiers et les résidus $\mathbf{R}^{(n)}$ doivent être regroupés en valeurs "proches" afin de pouvoir calculer des "histogrammes". Le principe général est alors de diviser les résidus $\mathbf{R}^{(n)}$ par un pas de quantification q et d'arrondir ensuite les résultats. Le pas de quantification q détermine alors la "granularité" de l'histogramme. Un pas très grand fournit des histogrammes grossiers : par exemple un pas de $q = 10$ conduit à un arrondi à la dizaine la plus proche au contraire, un pas de $q = 1$ conduit à un arrondi à l'entier le plus proche. Le seuillage consiste simplement à limiter l'intervalle des valeurs possibles (malgré la quantification) afin de limiter la dimension des "co-occurences" qui en découle. Il est simplement proposé de ne compter que les valeurs des résidus en dessous d'un certain seuil T et d'ignorer les valeurs au-delà.

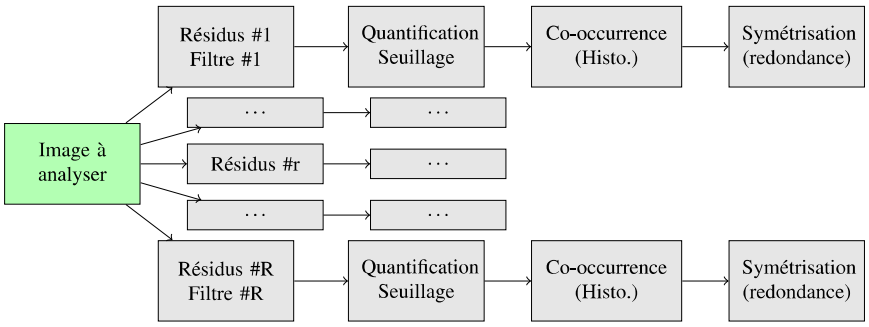


Figure 7.6 – Illustration du principe de l'extraction des caractéristiques SRM "*Spatial Rich Model*".

Cette seconde étape peut finalement se représenter par :

$$\widetilde{r}_{m,n}^{(n)} = \text{Seuil}_T \left(\text{Round} \left(\frac{r_{m,n}^{(n)}}{q} \right) \right), \qquad [7.23]$$

avec Seuil_T l'opération de seuillage avec le seuil T et Round l'opération d'arrondie à l'entier le plus proche. Dans les modèles riches spatiaux (Fridrich and Kodovskỳ 2012), 3 pas de quantification distincts sont utilisés ($q = \{1, 2, 3\}$), mais le seuil est toujours fixé à $T = 3$ cela afin de pouvoir représenter des valeurs de résidus plus ou moins grandes suivant le pas de quantification.

La troisième étape vise à représenter les "résidus" au travers d'histogrammes à plusieurs dimensions ou de co-occurrences. Cela permet principalement de mieux représenter les propriétés statistiques globales des résidus, indépendamment de leurs positions dans une image, mais également de réduire le nombre de données et enfin d'avoir une représentation identique quelle que soit la taille de l'image analysée. Le calcul de co-occurrence pour un type de résidus donnés $\mathbf{R}^{(n)}$ se fait en procédant, comme expliqué dans l'équation [7.20], en dénombrant le nombre valeur adjacente dont les valeurs sont toutes égales à un certain motif recherché. Il s'agit donc d'une généralisation de l'équation [7.20] à un tuplet de c résidus adjacents. Il est également notable qu'en utilisant c résidus adjacents, on peut également multiplier les directions qui peuvent être bien plus générales que simplement les 8 directions considérées dans le cas de paires. Le calcul des co-occurrences se limite généralement aux directions verticales et horizontales car l'ensemble des parcours possibles deviendrait beaucoup trop important avec des co-occurrences de dimension $c = 4$ telles qu'utilisées dans (Fridrich and Kodovskỳ 2012).

Enfin la dernière étape consiste à fusionner, au travers généralement d'un calcul de moyenne, les co-occurrences semblables, typiquement verticales vers la gauche ou

vers la droite. Le principe sous-jacent étant qu'il n'y a pas de raison objective de supposer que les traces de la stéganographie soient représentées différemment dans une direction plutôt qu'une autre. La symétrisation est une étape plus délicate à définir de façon générale, car cela dépend largement d'une part des filtres utilisés pour calculer les résidus, d'autre part des co-occurrences utilisées. Généralement on agglomérera les différentes directions (co-occurrences verticales et horizontales) ainsi que les filtres qui peuvent se déduire les uns des autres par symétrie. Cette dernière étape permet de réduire largement le nombre de caractéristiques utilisées. En effet, considérons le cas classique des caractéristiques issues des modèles spatiaux riches (Fridrich and Kodovskỳ 2012). Nous dénombrons 78 filtres distincts avec 3 pas de quantifications, un seuillage avec le seuil $T = 2$ et des co-occurrences calculées horizontalement et verticalement. Une image caractérisée par les SRM compte final (après symétrisation) $34,761$ caractéristiques.

7.4.1.3. *Extraction de caractéristiques dans le domaine JPEG*

En ce qui concerne les images compressées au format JPEG, l'extraction de caractéristiques pose un premier problème : faut-il utiliser directement les coefficients DCT, qui peuvent être modifiés, ou bien est-il préférable de "décompresser" l'image afin d'analyser les pixels ?

Les deux approches ont été étudiées et, aujourd'hui, la détection dans les images décompressées présente des performances plus intéressantes. Cela est notamment dû au fait que la modélisation et l'analyse de pixels contigus sont bien plus simples que l'analyse des coefficients DCT qui sont le résultat de filtres différents et présentent donc des propriétés qui ne permettent pas une analyse aisément.

Un compromis entre ces deux domaines d'analyse a été proposée par (Holub and Fridrich 2015), nous allons brièvement le décrire car il est performant et a donné lieu à de nombreux travaux ultérieurs, notamment (Song *et al*. 2015). Cette approche appelée DCTR (pour "DCT Resiuals") analyse une image en commençant par la décompresser, puis en appliquant aux pixels une transformation DCT similaire à celle utilisée dans la compression JPEG. Plus exactement, nous avons brièvement expliqué que la compression JPEG applique une série de 64 (8×8) filtre orthogonaux à des blocs disjoints de 64 (8×8) pixels. Le principe de DCTR repose sur l'utilisation de ces 64 filtres sur des blocs non-disjoints. Plus exactement ces filtres sont appliqués par convolution avec l'image décompressée ce qui revient à calculer des coefficients DCT en plus de ceux utilisés dans la compression JPEG sur des portions de blocs 8×8 adjacents. Le résultat de cette opération est donc 64 images distinctes de même taille que l'image d'origine, chacune correspondant à l'application d'un des filtres de transformation DCT. Cette étape est équivalent au calcul de 64 résidus.

Ces images sont ensuite quantifiées, modifiées en valeurs absolues (les valeurs négatives sont reportées en valeurs positives) puis seuillées. La quantification est d'autant plus grande que le facteur de qualité est petit (de façon similaire aux matrices

de quantification JPEG) et le seuillage est fixé à 4. Pour chacune de ces 64 "sous-images" plusieurs histogrammes sont calculés en fonction de la position des résidus. L'idée est clairement de donner une "référence" comparable aux coefficients DCT qui sont utilisés dans la compression JPEG (et donc possiblement modifié par la stéganographie) avec ceux qui ne sont pas utilisés mais qui correspondent à des images qui sont statistiquement très similaires puisque décalées seulement de quelques pixels.

Dans le cas des DCTR aucune symétrisation n'est utilisée, cependant certaines positions de "résidus DCT" sont rassemblées et nous nous retrouvons au final avec 25 histogrammes par filtres DCT et $4+1$ valeurs dans l'histogramme seuillé, soit au total $64 \times 25 \times 5 = 8000$ caractéristiques.

7.4.2. *Classification des caractéristiques*

Une fois les caractéristiques extraites, il reste une phase de classification à mettre en œuvre. Plus exactement, il faut extraire ces caractéristiques pour un grand nombre d'images Cover, pour ensuite cacher des informations dans ces images à l'aide d'une méthode de stéganographie pour générer des images Stégo et enfin extraire à nouveau des caractéristiques à partir de ces dernières. L'apprentissage d'une règle de classification dans un tel cadre fait partie de l'apprentissage statistique supervisé : nous avons ainsi deux bases de caractéristiques, la première pour des images sans informations Cover et la seconde pour des images Stégo. Il existe de nombreuses méthodes d'apprentissage statistique supervisées ; en général cela correspond mathématiquement à un problème d'optimisation afin de rechercher les paramètres d'une fonction permettant de maximiser la performance en détection. Puisqu'en stéganalyse les principaux classifieurs adaptés sont linéaires, nous nous focaliserons donc sur ce type de méthodes.

Considérons deux bases de L caractéristiques calculées sur I images que nous notons sous forme matricielle $\mathbf{C} = (\mathbf{c}^{(1)}, \ldots, \mathbf{c}^{(I)})$ pour les images Cover (de classe -1) et $\mathbf{S} = (\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(I)})$ pour les images Stégo (de classe 1). Un classifieur linéaire repose sur une projection des caractéristiques sur un vecteur de discrimination $\mathbf{w}^\top \mathbf{c}^{(i)} = \sum_{l=1}^{L} \mathbf{w}_l \mathbf{c}_l^{(i)}$. Cette opération est une somme pondérée des caractéristiques. Le but est donc de trouver un vecteur des coefficients de pondération $\mathbf{w}$ qui permet après projection de discriminer les données Cover et Stégo. Un critère simple pour cela consiste à chercher les coefficients de pondération $\mathbf{w}$ de sorte que les vecteurs $\mathbf{c}_i$ sont proches de la valeur -1 alors que les vecteurs $\mathbf{s}_i$ sont proches de la valeur 1.

Si l'on crée des grands vecteurs $\mathbf{y}_0$ et $\mathbf{y}_1$, le premier contenant I fois la valeur -1 et le second contient I fois la valeur 1, il s'agit alors de trouver les facteurs de pondération $\mathbf{w}$ minimisant la différence entre les valeurs $\mathbf{w}^\top \mathbf{C}$ et $\mathbf{y}_0$ (et réciproquement entre $\mathbf{w}^\top \times \mathbf{S}$ et $\mathbf{y}_1$), soit $\left\|\mathbf{w}^\top \mathbf{C} - \mathbf{y}_0\right\|_2^2 + \left\|\mathbf{w}^\top \mathbf{S} - \mathbf{y}_1\right\|_2^2$.

Cette méthode revient en fait à un problème de minimisation au sens des moindres carrés. Cette approche est particulièrement intéressante, car une solution analytique est donnée par :

$$\mathbf{w} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y}, \qquad [7.24]$$

ou $\mathbf{X}$ est une matrice regroupant les caractéristiques $\mathbf{X} = (\mathbf{C}, \mathbf{S})$ et, de façon similaire, le vecteur $\mathbf{y}$ contient toutes les valeurs de labels $\mathbf{y} = (\mathbf{y}_0; \mathbf{y}_1)$. Pour que cette méthode fonctionne correctement, il est généralement utile d'ajouter un facteur de régularisation, qui permet de trouver un compromis entre une solution "simple" et une solution adaptée aux données utilisées pour l'apprentissage. C'est cette méthode qui a été proposée dans (Cogranne *et al*. 2015) et qui permet d'obtenir des performances très proches de l'état de l'art en seulement quelques secondes (pour 10000 images et 40000 caractéristiques).

Une approche alternative intéressante reposant sur un autre classifieur linéaire, celui de Fisher, très similaire au précédent, mais qui vise à maximiser le critère de séparabilité :

$$\frac{\mathbf{w}^\top \left(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0\right)}{\mathbf{w}^\top \left(\boldsymbol{\Sigma}_0 + \boldsymbol{\Sigma}_1\right) \mathbf{w}}, \qquad [7.25]$$

avec $\boldsymbol{\mu}_i$ la moyenne des caractéristiques pour les images de la classe i et $\boldsymbol{\Sigma}_i$ la matrice de covariance des caractéristiques de la classe i. Cette méthode présente également l'avantage d'avoir une solution directement calculable :

$$\mathbf{w} = \left(\boldsymbol{\Sigma}_0 + \boldsymbol{\Sigma}_1\right)^{-1} \left(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0\right). \qquad [7.26]$$

Le principal intérêt de la méthode proposée dans (Kodovskỳ *et al*. 2012) réside dans l'utilisation d'une multitude de classifieurs. Pour cela de nombreux classifieurs sont entraînés sur des "sous-ensembles" d'images et/ou sur des "sous-ensembles" de caractéristiques. L'intérêt repose principalement sur le fait que cet entraînement spécifique et cette procédure de multiplication des classifieurs permet de construire, au final, une classifieur "robuste", non linéaire, et performant. Cette méthode a été améliorée dans (Cogranne and Fridrich 2015), mais, dans les deux cas, les résultats sont légèrement meilleurs que ceux obtenus avec un classifieur linéaire simple pour une complexité calculatoire beaucoup plus importante (de l'ordre de 20 à 100 fois supérieur).

7.5. Détection par réseaux de neurones profonds

Nous présentons maintenant les dernières évolutions des méthodes de stéganalyse basées sur l'apprentissage, à savoir l'utinisation de réseaux de neurones profonds. Les réseaux de neurones sont étudiés depuis les années cinquante. Initialement, ils ont été proposés pour modéliser le comportement du cerveau. En informatique, en particulier en intelligence artificielle, ils ont été utilisés pendant 30 ans à des fins d'apprentissage. Au début des années 2020 (Hinton and Salakhutdinov 2006), les réseaux de neurones profonds étaient considérés comme ayant un temps d'apprentissage trop long et comme étant moins efficaces que les classifieurs comme les SVMs ou les forêts aléatoires.

Grâce aux avancées récentes dans le domaine des réseaux de neurones (Bengio *et al*. 2013), grâce à la puissance de calcul fournie par les cartes graphiques (GPUs), et enfin grâce à la profusion de données, les approches d'apprentissage en profondeur ont été proposées comme une extension naturelle des réseaux neuronaux. Depuis 2012, ces réseaux profonds ont marqué profondément les domaines du traitement du signal et de l'intelligence artificielle car leurs performances ont permis de surpasser les méthodes les plus performantes de l'époque, mais aussi de solutionner des problèmes que les scientifiques n'arrivaient pas à résoudre jusqu'à maintenant (LeCun *et al*. 2015).

En stéganalyse, pendant les dix dernières années, la détection d'un message caché dans une image a été majoritairement réalisée par le calcul d'un modèle riche (SRM[13]) (Fridrich and Kodovskỳ 2012) suivi d'une classification par un classificateur par ensemble (CE) (Kodovskỳ *et al*. 2012). En 2015, la première étude utilisant un réseau de neurones convolutionnel (CNN) a obtenu des premiers résultats de stéganalyse par "deep-learning" approchant les résultats des approches en deux étapes (EC+SRM[14]) (Qian *et al*. 2015). Dès lors, sur la période 2015 - 2019, de nombreuses publications ont montré qu'il est possible d'obtenir de meilleures performances que cela soit pour la stéganalyse d'images spatiale (non compressées), la stéganalyse d'image compressées en JPEG, mais également la stéganalyse informée, la stéganalyse quantitative, et la stéganalyse couleur.

Dans la section 7.5.1 nous présentons de manière générique la structure d'un réseau de neurones profond. La lecture de cette section peut être complétée par des lecture sur l'apprentissage artificiel, et notamment sur la définition du *perceptron*, la *descente de gradient stochastique*, la *retro-propagation* et l'extension au cas *multi-classes*.

13. voir section 7.4.1.2.

14. EC+SRM désignera de manière générale les approches en deux étapes reposant sur le calcul d'un modèle riche (SRM) puis l'utilisation d'un ensemble classifieur (EC).

7.5.1. *Les briques de bases d'un réseau de neurones profond*

Dans les sections suivantes, nous rappelons les concepts majeurs d'un réseau de neurones convolutif (Convolutional Neural Network) ; nous rappelons les briques élémentaires d'un réseau en nous basant sur le réseau Yedroudj-Net qui a été publié en 2018 (Yedroudj, Comby and Chaumont 2018) et qui reprend les idées présentes dans Alex-Net (Krizhevsky *et al*. 2012), ainsi que celles présentes dans des réseaux développés pour la stéganalyse dont le tout premier réseau de Qian *et al* ., et les réseaux de Xu-Net (Xu *et al*. 2016) et Ye-Net (Ye *et al*. 2017).

7.5.1.1. *Vue globale d'un réseau de neurones convolutif*

Avant de décrire la structure d'un réseau de neurones ainsi que ses briques élémentaire, il est utile de rappeler qu'un réseau de neurones appartient à la famille de l'apprentissage automatique. Dans le cas de l'apprentissage supervisé qui est le cas qui nous concerne, il est nécessaire de disposer d'une base de données d'images, avec pour chaque image son label, c'est-à-dire sa classe.

Les réseaux d'apprentissage profond sont de grands réseaux neuronaux qui peuvent directement prendre des données brutes en entrée. En traitement d'image, le réseau est directement alimenté par les pixels formant l'image. Un réseau d'apprentissage en profondeur apprend ainsi, de manière jointe, à la fois à extraire les caractéristiques intrinsèques de l'image (généralement appelées *carte de caractéristiques* ou bien *espaces latents*), et à dessiner la frontière de séparation les différentes classes (*plans séparateurs*).

Le protocole d'apprentissage est similaire aux méthodes classiques d'apprentissage automatique. Chaque image est donnée en entrée du réseau. Chaque valeur du pixel est transmise à un ou plusieurs neurones. Le réseau est constitué d'un nombre donné de *blocs*. Un bloc est constitué de neurones qui prennent des valeurs réelles en entrée, effectuent des calculs, puis transmettent au bloc suivant les valeurs réelles calculées. Un réseau de neurones peut donc être représenté par un graphe orienté où chaque nœud représente une unité de calcul. L'apprentissage est alors réalisé en fournissant au réseau des exemples composés d'une image et de son label, et le réseau modifie les paramètres de ces unités de calcul en apprenant grâce aux algorithmes de « rétro-propagation » et de « descente de gradient stochastique ».

Les réseaux de neurones convolutifs utilisés pour la stéganalyse sont majoritairement construits en trois parties, appelés *modules* : le module de pre-processing, le module de convolution, et le module de classification. À titre d'illustration, la figure 7.7 schématise le réseau proposé par Yedroudj *et al.* en 2018 (Yedroudj, Comby and Chaumont 2018). Ce réseau traite les images en niveau de gris de taille 256×256 pixels.

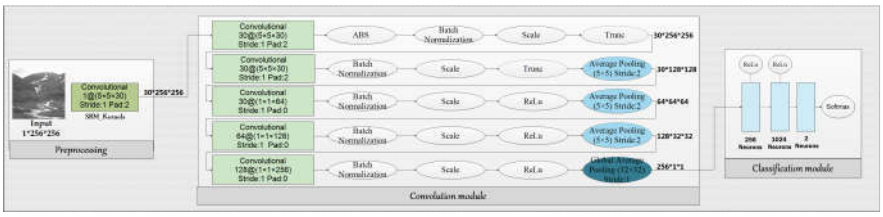


Figure 7.7 – Le réseau Yedroudj-Net (Yedroudj, Comby and Chaumont 2018).

7.5.2. *Le module de préprocessing :*

Nous pouvons voir sur la figure 7.7 que dans le module de pré-processing l'image est filtré par 30 filtres passe-haut. L'utilisation d'un ou plusieurs filtres passe-haut comme pré-traitement est présente dans la majorité des réseaux utilisés pour la stéganalyse durant la période 2015-2019. À titre d'exemple, le noyau « *square S5a* » (Fridrich and Kodovskỳ 2012) est donné par :

$$F^{(0)} = \frac{1}{12}\begin{pmatrix} -1 & 2 & -2 & 2 & -1 \\ 2 & -6 & 8 & -6 & 2 \\ -2 & 8 & -12 & 8 & -2 \\ 2 & -6 & 8 & -6 & 2 \\ -1 & 2 & -2 & 2 & -1 \end{pmatrix}. \qquad [7.27]$$

Cette étape préliminaire de filtrage permet au réseau de converger plus rapidement et est probablement nécessaire pour obtenir de bonnes performances lorsque la base d'apprentissage est trop petite (Yedroudj, Chaumont and Comby 2018) (seulement 4 000 paires cover/stego de taille 256×256 pixels). Les images filtrées sont ensuite transmises au premier bloc de convolution du réseau. Le réseau *Yedroudj-Net* possède 5 blocs de convolution (Yedroudj, Comby and Chaumont 2018), comme les réseaux de Qian *et al.* (Qian *et al*. 2015) et de Xu *et al.* (Xu *et al*. 2016).

Notons que le récent réseau SRNet (Boroumand *et al*. 2019) n'utilise pas de pré-filtres fixes, mais apprend les valeurs des filtres. Cela nécessite donc d'avoir une base de données beaucoup plus importante (plus de 15 000 paires cover/stego de 256×256 pixels), et éventuellement d'utiliser un apprentissage préliminaire pour partir d'une bonne initialisation. Il y a d'ailleurs un débat dans la communauté pour savoir si nous devons utiliser des valeurs de filtres fixées, ou initialiser les valeurs des filtres avec des valeurs pré-sélectionnées et ensuite affiner ces valeurs par apprentissage, ou encore avoir une initialisation aléatoire et laisser le réseau apprendre les valeurs des filtres. Au moment où nous écrivons ce chapitre, nous pouvons penser que le meilleur choix est lié à l'architecture du réseau, la taille de la base de données d'apprentissage utilisée,

et à la possibilité d'utiliser un apprentissage existant pour initialiser l'apprentissage d'intérêt.

7.5.2.1. *Le module de convolution :*

Afin d'éviter toute ambiguïté sur les termes, nous éviterons d'utiliser le terme *couche* ; le *opérations* sera alors préféré pour une fonction élémentaire (convolution, activation, ...) et le terme de *bloc* pour un ensemble de ces opérations qui peuvent se succéder. Un *bloc* est composé d'unités de calculs qui prennent des valeurs réelles en entrée, effectuent des calculs, puis renvoient des valeurs réelles, qui sont transmises au bloc suivant. Concrètement, un *bloc* prend un ensemble de *carte de caractéristique*[15] en entrée et retourne un ensemble de *carte de caractéristique* en sortie. À l'intérieur d'un bloc, nous retrouvons un certain nombre d'opérations dont les quatre suivantes : la *convolution*, l'*activation*, la *fusion*[16], et enfin la *normalisation*.

Notons que le concept de neurone, tel que défini dans la littérature avant l'émergence des réseaux convolutifs, est toujours présent, mais il a disparu dans les codes source que structure de donnée. Dans les modules de convolution, il faut imaginer un neurone comme une unité de calcul qui, pour une position dans la *carte de caractéristique* prise par le noyau de convolution lors de l'opération de convolution, effectue la somme pondéré entre le noyau et le groupe de pixels considérés. Le concept de neurone correspond au produit scalaire entre les données d'entrée que sont les pixels et des données propres au neurone[17] suivi de l'application d'une fonction de $\mathbb{R}$ dans $\mathbb{R}$ appelée la fonction d'activation. Ensuite, par extension, nous pouvons considérer que la *fusion* et la normalisation sont des opérations propres aux neurones.

En ne comptant pas le bloc de pré-traitement, le réseau Yedroudj-Net (Yedroudj, Comby and Chaumont 2018) a un module de convolution composé de 5 blocs de convolution, comme les réseaux de Qian *et al.* (Qian *et al.* 2015) et de Xu *et al.* (Xu *et al.* 2016). Le réseau Ye-Net (Ye *et al.* 2017) possède un module de convolution composé de 8 blocs de convolution. Le réseau SRNet (Boroumand *et al.* 2019) contient un module de convolution composé de 11 blocs de convolution.

Sur la figure 7.7 représentant le réseau Yedroudj-Net, le premier bloc du module de convolution génère 30 *carte de caractéristique*, chacune de taille 256×256 pixels. Notons que cela signifie qu'il y a 30 filtres et donc 30 convolutions qui sont apprises sur le groupe d'images données en entrée (les 30 images filtrées) de taille 256×256 pixels. Dans chacun des 5 blocs, nous retrouvons une opération de convolution, une opération de normalisation ("Batch" + "Scale"), une opération d'activation (ABS, trunc, ReLU), et une opération de fusion ("average" ou "global pooling").

15. qui peut être vu comme un ensemble d'images.
16. appelée *pooling* en anglais
17. c'est-à-dire le noyau de convolution.

7.5.2.2. *Le module de classification :*

Le dernier bloc du module de convolution (voir section 7.5.2.1) est connecté au *module de classification* qui est généralement un réseau de neurones *entièrement connecté* composé de un à trois blocs. Ce *module de classification* est souvent un réseau de neurones traditionnel où chaque neurone est complètement connecté à la *couche* de neurones suivante et à la *couche* de neurones précédente.

Après ce réseau de neurones complètement connecté, nous trouvons souvent une fonction "softmax" qui permet de normaliser les deux sorties fournies par le réseau de sorte que les valeurs émises appartiennent à $[0, 1]$ et que leur somme vaut 1. La fonction softmax retourne donc un score d'appartenance à une classe, c'est-à-dire un score par neurone de sortie. Par abus de langage ces scores sont appelés probabilités. Nous conserverons cette dénomination. Dans le scénario habituel de stéganalyse binaire, le réseau délivre donc deux valeurs en sortie : l'une donnant la probabilité de classement dans la première classe (par exemple la classe Cover), et l'autre donnant la probabilité de classement dans la seconde classe (par exemple la classe Stégo). La décision de classification est alors obtenue en renvoyant la classe avec la plus forte probabilité. Le réseau Yedroudj-Net (voir figure 7.7) fournit effectivement deux valeurs en sortie.

Notons que juste avant ce *module de classification*, nous pouvons trouver une opération de *fusion* particulière, telle qu'un *global average pooling*, un *Spatial Pyramid Pooling (SPP)* (He *et al.* 2014), ou un *extracteur de moments statistiques* (Fuji-Tsang and Fridrich 2018). Une telle opération de fusion renvoie un vecteur de taille fixe, c'est-à-dire une carte de caractéristiques de dimension fixe, et cela, quelle que soit la dimension de l'image en entrée du réseau. Le bloc venant après cette opération de fusion est alors toujours connecté à un vecteur de taille fixe, il a donc un nombre fixe de paramètres d'entrée. Il est ainsi possible de présenter au réseau des images de toutes tailles sans avoir à modifier la topologie du réseau. Cette propriété est disponible dans le réseau Yedroudj-Net (Yedroudj, Comby and Chaumont 2018), le réseau Zhu-Net (Zhang *et al.* 2020), ou le réseau (Fuji-Tsang and Fridrich 2018).

Notons que (Fuji-Tsang and Fridrich 2018) est le seul papier (jusqu'à 2019 et le concours de stéganalyse ALASKA (Cogranne *et al.* 2019 ; Yousfi *et al.* 2019)), qui se soit sérieusement penché sur la viabilité d'un réseau invariant à la dimension des images. Le problème reste cependant ouvert. La solution proposée dans (Fuji-Tsang and Fridrich 2018) est une variante du concept d'"average pooling". Jusqu'en 2019, le faible nombre d'études n'est pas suffisant pour déterminer quelle est la topologie du réseau à retenir, ni de savoir comment construire la base de données d'apprentissage, ou dans quelle mesure le nombre de bits insérés influe sur l'apprentissage.

Ici, s'achève la rapide présentation d'un réseau de neurones convolutif vue à travers les publications majeures en stéganographie/stéganalyse [18].

7.5.2.3. *Utilisation de la carte de probabilité de modification (SCA)*

La carte de probabilité de modification, lorsqu'elle est connue par Ève, peut permettre d'améliorer les performances en stéganalyse de façon significatives. Fin 2018, deux approches (SCA, « Selection Channel-Aware ») combinent cette connaissance avec les réseau de neurones profonds : le SCA-Ye-Net (qui est la version SCA de Ye-Net) (Ye *et al*. 2017), et le SCA-SRNet (qui est la version SCA du SRNet) (Boroumand *et al*. 2019). L'idée est d'utiliser un réseau utilisé pour la stéganalyse non informée et d'injecter non seulement l'image à stéganalyser, mais aussi la carte de probabilité de modification. Nous supposons donc qu'Eve connaît, ou peut obtenir une bonne estimation (Sedighi and Fridrich 2015) de cette carte, c'est-à-dire qu'Eve a accès aux informations du canal de sélection (SCA).

Cette carte est donnée au bloc de pré-traitement SCA-Ye-Net (Ye *et al*. 2017), et de manière équivalente au premier bloc de convolution pour SCA-SRNet (Boroumand *et al*. 2019), mais les valeurs du noyau sont remplacées par leur valeur absolue. Après la convolution, chaque carte de caractéristiques est additionnée point à point à la carte de probabilité de modification filtrée correspondante. Notons que la fonction d'activation de cette première convolution (bloc de pré-traitement pour SCA-Ye-Net ou premier bloc pour SCA-SRNet) est (si ce n'est pas déjà le cas) remplacée par une activation "ReLU". Dans SCA-Ye-Net, la fonction d'activation de troncature est, en effet, remplacée par un ReLU. Ceci permet de propager “virtuellement” dans tout le réseau les informations relatives à l'image, et les informations relatives à la carte de probabilité.

Notons que cette procédure de transformation d'un réseau classique en réseau SCA s'inspire de la propagation de la carte de probabilité de modification proposée par (Denemark *et al*. 2016). Ces deux articles constituent une amélioration par rapport au précédent maxSRM Rich Models (Denemark *et al*. 2014). Dans maxSRM, au lieu d'accumuler le nombre d'occurrences dans la matrice de co-occurrence, nous utilisons une accumulation du maximum d'une probabilité locale. Dans (Denemark *et al*. 2016), l'idée est de transformer la carte de probabilité de modification de la même manière que le filtrage de l'image, puis de mettre à jour la matrice de co-occurrence en utilisant la version modifiée de la carte de probabilité de modification plutôt que la carte de probabilité de modification initiale.

---

18. pour un peu plus de détails concernant la description sur la convolution, l'activation, le fusion, la normalisation, l'efficacité et la complexité en temps et en mémoire, ou bien le lien entre le deep-learning et les approches passées, ou même les approches d'insertion par deep-learning, le lecteur peut consulter (Chaumont 2020), disponible en ligne ici `https://arxiv.org/abs/1904.01444`.

7.5.2.4. *La stéganalyse JPEG*

Le meilleur CNN pour la stéganalyse JPEG à la fin de 2018 était SRNet (Boroumand *et al.* 2019). A cette époque, c'est le seul réseau pour la stéganalyse JPEG à disposer d'une version SCA (Side Channel Aware). Il est intéressant d'énumérer et de discuter brièvement des CNNs précédents utilisés pour la stéganalyse JPEG. Le premier réseau, publié en février 2017, était le réseau de Zeng *et al.* Il a été évalué avec un million d'images (Zeng *et al.* 2017) (Zeng *et al.* 2018). Puis, en juin 2017, à IH&MMSec'2017, deux réseaux ont été proposés : PNet (Chen *et al.* 2017) et Xu-Net-Jpeg (Xu 2017). Enfin, SRNet (Boroumand *et al.* 2019) a été mis en ligne en septembre 2018.

Dans le réseau de Zeng *et al.* (Zeng *et al.* 2017) (Zeng *et al.* 2018), le bloc de pré-traitement prend en entrée une image « dé-quantifiée » (valeurs réelles), puis la convolue avec 25 bases DCT, puis quantifie et tronque les 25 images filtrées. Ce bloc de pré-traitement, utilise des noyaux de filtres fabriqués à la main (bases DCT), les valeurs des noyaux sont fixes, et ces filtres sont inspirés des modèles riches DCTR (Holub and Fridrich 2015). Il y a trois quantifications différentes et le bloc de pré-traitement produit 3×25 images résiduelles. Le CNN est donc constitué de 3 sous-réseaux qui produisent chacun un vecteur caractéristique de dimension 512. Les sous-réseaux sont inspirés de Xu-Net (Xu *et al.* 2016). Les trois vecteurs de caractéristiques, retournés par les trois sous-réseaux, sont ensuite donnés à une structure entièrement connectée, et le réseau se termine par une couche softmax.

À l'instar de ce qui a été fait pour la stéganalyse spatiale, ce réseau utilise un bloc de pré-traitement inspiré des SRM (Holub and Fridrich 2015). Notons que les modèles riches les plus efficaces, aujourd'hui, sont les « *Gabor Filter Rich Models* » (GFR) (Song *et al.* 2015). Notons également que ce réseau tire parti de la notion d'ensemble, qui provient des trois sous-réseaux. Le réseau de Zeng *et al.* est moins efficace que Xu-Net-Jpeg (Xu 2017), mais donne une première approche intéressante guidée par les Rich Models.

L'idée principale du PNet (et aussi du VNet qui est moins efficace, mais qui prend moins de mémoire) (Chen *et al.* 2017) est d'imiter les caractéristique sensibles à la phase, tels que les DCTR (Holub and Fridrich 2015) ou les GFR (Song *et al.* 2015), et d'obtenir une image en entrée décomposée en 64 cartes de caractéristiques représentant les 64 phases d'une image JPEG. Le bloc de pré-traitement prend en entrée une image dé-quantifiée (valeurs réelles), procède à des convolutions avec quatre filtres, le "SQUARE5$\times$5" provenant du Spatial Rich Models (Fridrich and Kodovskỳ 2012), un filtre passe-haut "point" (appelé "catalyst kernel") qui complète le "SQUARE5$\times$5", et deux filtres directionnels de Gabor (angles 0 et $\pi/2$).

Juste après le deuxième bloc de convolution, un module de "division de phase" (*PhaseSplit Module*) divise l'image résiduelle en 64 cartes de caractéristiques (une

carte = une phase), de la même façon que dans les modèles riches. Certaines procédures intéressantes ont été utilisées telles que : (1) l'enchaînement de convolutions à valeurs figées dans le bloc de pré-traitement, avec une seconde convolution dont les poids sont appris, (2) une mise à jour intelligente des paramètres de la normalisation par lots, (3) l'utilisation de "Filter Group Option" qui construit virtuellement des sous-réseaux, (4) le bagging sur 5 validations croisées, (5) l'utilisation des 5 dernières évaluations afin de donner l'erreur moyenne d'un réseau, (6) le brassage de la base de données au début de chaque époque afin d'avoir un meilleur comportement de la normalisation par lots et aider à la généralisation, et (7) éventuellement l'utilisation d'un ensemble de réseaux de même type avec une fusion des sorties par vote majoritaire. Avec un tel savoir-faire, PNet a battu les approches classiques d'apprentissage machine en deux étapes dans les scénarios non-SCA et SCA (Ensemble Classifier + GFR).

Le Xu-Net-Jpeg (Xu 2017) est d'autant plus attractif que l'approche est légèrement meilleure que PNet, et ne nécessite pas une forte inspiration de domaine comme pour PNet. Le Xu-Net-Jpeg est fortement inspiré de ResNet (He *et al.* 2016), un réseau bien établi de la communauté d'apprentissage automatique. ResNet permet l'utilisation de réseaux plus profonds grâce à l'utilisation de raccourcis (« *shortcuts* » en anglais). Dans Xu-Net, le bloc de pré-traitement prend en entrée une image « déquantifiée » (valeurs réelles), convolue l'image avec 16 bases DCT (dans le même esprit que le réseau de Zeng et al. (Zeng *et al.* 2017) (Zeng *et al.* 2018)), puis applique une valeur absolue, une troncature, et un ensemble de convolutions, BN, ReLU jusqu'à obtenir un vecteur caractéristique de dimension 384, qui est donné à un bloc entièrement connecté. On peut noter que le "max pooling" ou l'"average pooling" est remplacé par des convolutions. Ce réseau est vraiment simple et était, en 2017, la méthode la plus performante. De manière générale, les réseaux proposés par la communauté de l'apprentissage machine automatique sont souvent très compétitifs, avec peu de connaissances spécifiques au domaine de la stéganalyse à intégrer à la topologie d'un réseau pour obtenir un réseau très efficace[19].

En 2018, le CNN à l'état de l'art pour la stéganalyse JPEG (qui peut également être utilisé pour la stéganalyse spatiale) était SRNet (Boroumand *et al.* 2019). Notons que pour la version de SRNet qui a connaissance du canal adjacent (SCA), la probabilité de modification par coefficient DCTs est d'abord ré-exprimée dans le domaine spatial en appliquant une DCT inverse, et en utilisant les valeurs absolues pour la base DCTs. La *carte de sélection* ainsi obtenue, entre ensuite dans le réseau, et est convoluée avec chaque noyau (cette première convolution est équivalente au bloc de pré-traitement). Notons que les convolutions dans ce premier bloc de cette *carte de sélection* sont telles

19. la récente compétition ALASKA#2 (Cogranne *et al.* 2020*a*) a d'ailleurs montrée que les réseaux issues de la communauté « *Deep Learning* » peuvent être directement utilisés en stéganalyse avec d'excellents résultats.

que les noyaux des filtres sont modifiés à leurs valeurs absolues. Après avoir passé la convolution, les « *carte de caractéristiques* » sont additionnées avec la racine carrée des valeurs de la *carte de sélection* précédemment convoluée. Notons que cette idée reprend celle qui est exposée dans la version SCA Ye-Net (SCA-TLU-CNN) (Ye *et al.* 2017) avec intégration d'une information adjacente, et la récente proposition pour la stéganalyse avec connaissance du canal de sélection (SCA) dans JPEG avec des Rich Models (Denemark *et al.* 2016), où la construction de la *carte de sélection*, et plus particulièrement la quantité $\delta^{1/2}_{uSA}$[20].

7.6. Pistes de recherches actuelles

Nous terminons ce chapitre introductif à la stéganalyse par un bref inventaire des problèmes ouverts qui nous semblent les plus intéressants et les plus importants dans ce domaine.

7.6.1. *Le problème du* Cover-Source Mismatch

Le premier problème que nous allons décrire est celui du « *Cover Source Mismatch* »[21] (CSM) ; ce problème est en fait un cas assez général dans le domaine de l'apprentissage statistique où il y ait fait référence sous le terme de "généralisation". En pratique il s'agit de l'inadéquation entre la base d'apprentissage, sur lequel le classifieur est entraîné , et la base de test sur laquelle nous souhaitons détecter la présence d'informations cachées. La particularité de la stéganalyse est double par rapport au CSM. D'une part en stéganalyse nous souhaitons faire de la détection de signaux très faibles, d'autre part nous travaillons généralement avec des caractéristiques de (très) grandes dimensions. Conjointement ces deux particularités font que les méthodes de détection seront très sensibles aux bases d'apprentissages et difficilement généralisables.

Une étude assez exhaustive concernant l'évaluation des facteurs occasionnant les CSM a été menée dans (Giboulot *et al.* 2020) montrant par exemple le rôle prépondérant du traitement des images. Malheureusement il n'existe pas encore de travaux permettant d'en comprendre les causes profondes et, en conséquence, il reste très difficile de passer outre ce problème.

7.6.2. *Le problème de la stéganalyse* en conditions réelles

De façon assez similaire, la stéganalyse *en condition réelle* a été assez peu explorée (Ker and Pevný 2014). En fait, comme nous l'avons évoqué en introduction, la

20. uSA signifie Upper bounded Sum of Absolute values.

21. pouvant être appelé "disparité des sources Cover" en français.

stéganalyse a été développée dans le but principal d'évaluer les différentes méthodes de stéganographie. Pour cela, la communauté travaille généralement en utilisant le scénario de stéganalyse ciblée et avec des bases d'images spécifiques, notamment la base BOSS (Bas *et al*. 2011) constituée d'images de 7 appareils photographiques reflex (pas de smartphone ni de compact) toutes traitées de la même façon, à partir du fichier RAW. Des travaux ont montré que développer ces images RAW de façon différente peut amener des résultats très différents (Sedighi, Fridrich and Cogranne 2016). Un premier concours a été lancé dans ce but avec une base d'image beaucoup plus hétérogène (Cogranne *et al*. 2019) et les propositions montrent la difficulté de procéder à de la stéganalyse sur des images hétérogènes. Les gagnants se sont notamment appuyés sur une multiplication de l'apprentissage en fonction des paramètres de compression JPEG (Yousfi *et al*. 2019), mais de nombreuses questions sur la stéganalyse en conditions réelles restent ouvertes.

7.6.3. *La stéganalyse* fiable

Un problème assez proche des deux précédents concerne la stéganalyse fiable. En effet comme nous l'avons évoqué dans la section 7.3.2, un test est nécessairement entaché d'erreurs (faux-positifs et faux-négatifs) et la conséquence dépend grandement du contexte applicatif. Il semble important en stéganalyse de minimiser les faux-positifs compte tenu du nombre important d'images que l'on peut avoir à analyser. Au contraire les méthodes basées sur l'apprentissage statistique visent généralement à minimiser le taux d'erreur global P_E. Or, à supposer qu'il est possible de concevoir une méthode de détection dont les probabilités de faux-positif et de faux-négatif soient toutes les deux d'environ 1%, il est clairement illusoire d'appliquer cette méthode en pratique. Il est pertinent de faire valoir, avec une approche Bayésienne, qu'une image jugée comme étant stéganographiée par un tel détecteur serait en fait plus probablement issue d'un faux-positif étant donné que, sur un site de partage d'images tel FlickR, il est vraisemblable que moins de 1% des images contiennent effectivement des informations cachées ...

Se pose donc la question cruciale, comment concevoir une méthode de stéganalyse *fiable* dans le sens où elle garantirait une très faible probabilité de faux-positif ; typiquement une probabilité de faux-positif inférieure à 10^{-6} (soit moins de 1 sur 1 million). Nous avons vu que la théorie des tests d'hypothèses permet de concevoir de telles méthodes de stéganalyse, mais cela repose sur un modèle statistique des images qui lui n'est pas nécessairement exact, *a fortiori* lorsque certains paramètres doivent être estimés. Une étude très intéressante dans (Pevný and Ker 2015) étudie la possibilité de faire de l'apprentissage avec pour critère de minimiser la probabilité de

faux-positif si la puissance ς est fixée à 50%. Cependant, ce travail préliminaire mérite d'être approfondi et des méthodes de stéganalyse fournissant une « p-valeur »[22] précise manquent cruellement.

7.6.4. *La stéganalyse d'images couleurs*

Comme indiqué dans le chapitre 2, la très grande majorité des images sont aujourd'hui en couleur ; pour une applications pratique, il est donc nécessaire d'étudier ce type d'images. Or, les travaux académiques diffèrent de l'utilisation pratique puisque la très grande majorité des travaux en stéganalyse portent sur des images en « niveaux de gris », voir (Cogranne *et al*. 2019, 2020*a*,*b*).

La détection d'informations cachées dans les images couleurs semble au contraire plus intéressante en pratique. Ce sujet demeure largement ignoré. En effet, d'une part une part importante des chercheurs considèrent qu'une image couleur peut être représentée par trois images en niveaux de gris (une rouge, une verte et une bleue) qui peuvent être analysés séparément.

Il semble intuitif de considérer cependant que les canaux de couleurs sont statistiquement corrélés, notamment en raison du « dématriçage » lors de l'acquisition et du traitement des images. Cependant les travaux qui ont été menés dans ce domaine n'ont pas montré une grande amélioration en analysant les composantes conjointement (Goljan *et al*. 2014 ; Abdulrahman *et al*. 2016). Le premier travail dans ce domaine (Goljan *et al*. 2014) a par exemple proposé d'ajouter des histogrammes obtenus à partir des valeurs de pixels dans différentes couleurs. Cela a permis, certes d'améliorer légèrement la détection, mais en deçà de ce qui pouvait être attendu.

Enfin ce domaine d'étude n'a que très peu été étudié pour les images couleurs compressées avec le standard JPEG. Cela est plus dommageable, car les images couleur JPEG ne représentent pas le vert, le rouge et le bleu mais des composantes Luminance/Chrominance dont les corrélations statistiques sont moins importantes. D'autre part, les canaux de couleurs sont traités de façon distincte dans le standard JPEG, il faudrait donc les analyser de façon différenciée. Malheureusement, la détection de la stéganographie dans les canaux couleur du JPEG n'a pratiquement jamais été étudiée (voir notamment (Taburet *et al*. 2018) et les articles (Cogranne *et al*. 2019 ; Yousfi *et al*. 2019) relatifs au challenge de stéganalyse ALASKA et portant sur des images JPEG couleurs).

22. c'est-à-dire une probabilité que cette détection corresponde à un faux-positif pour une image donnée.

7.6.5. *Prise en compte de l'adaptativité de la stéganographie*

Un problème très différent, déjà été évoqué dans la section 7.5.2.3 dans le cadre des réseaux de neurones, concerne la prise en compte du « canal de sélection » pour la stéganalyse . Le canal de sélection désigne les probabilités d'utiliser chaque pixel $\beta_{m,n}$. Clairement il semble intéressant (comme le montre d'ailleurs les équations [7.14] et [7.16]) qu'un détecteur puisse utiliser une estimation, même erronée, de ces probabilités d'utilisation. Mais cela est très peu efficace avec les méthodes de détection actuelles. L'une des premières avancées qui a été proposée dans ce domaine (Denemark *et al.* 2014) consiste par exemple à remplacer le calcul d'histogrammes des co-occurrences (voir section 7.4.1) par l'ajout, pour chacun des blocs considérés, du maximum de la probabilité de modification $\beta_{m,n}$ de cet échantillon. Dans cette étude (Denemark *et al.* 2014), les auteurs indiquent avoir essayé plusieurs fonctions de pondération et avoir finalement constaté que le *maximum* permet d'obtenir les meilleurs résultats en termes de performances. Par ailleurs, les auteurs indiquent que cette fonction est également robuste à une mauvaise estimation du canal de sélection $\beta_{m,n}$. Cela est particulièrement important, car cette quantité n'est pas directement accessible et il faut donc l'estimer. Ceci nécessite de connaître la méthode d'insertion et la taille du message, et en pratique les performances obtenues à partir de la probabilité estimée ou de la probabilité réelle sont très proches. Dans tous les cas, les méthodes SCA (« Selection Channel-Aware ») demeurent très ad-hoc et elles sont justifiées de façon empirique essentiellement par des résultats en terme de performance de la détection statistique.

7.6.6. *La stéganalyse groupée (par lot)*

Une problématique qui est restée également largement peu étudiée concerne la stéganalyse « groupée » qui répond à la stéganographie « par lots » (Ker and Pevný 2012). Il s'agit d'un scénario plus général où Alice peut « étaler » son message dans plusieurs images distinctes qui sont ensuite envoyées à Bob. La stéganalyste, Eve, doit alors observer un lot d'images q et prendre une décision « groupée » concernant l'ensemble de ces images : elles peuvent être toutes Cover, ou bien certaines peuvent contenir une portion variable du message à cacher. Pour la stéganographie[23], ce cas peut se modéliser comme un problème d'allocation d'un message qui doit être dispersé dans plusieurs images afin de minimiser la probabilité de détection. Pour la stéganalyste Eve, le travail qui consiste à analyser un lot d'images est nettement plus délicat. De nombreuses questions restent ouvertes notamment, comment analyser un nombre d'images *a priori* inconnu ? Est-il préférable d'utiliser une méthode conçue pour analyser plusieurs images conjointement ou bien faut-il procéder à q analyses d'images indépendantes ? Enfin, ne faut-il pas tenir compte de l'ordre des images ?

23. voir également la section 7.6.6 du chapitre précédant.

Quelques travaux ont été proposés, notamment (Cogranne *et al*. 2017 ; Pevny and Nikolaev 2015) qui reposent sur un détecteur adapté à l'analyse de chaque image individuellement. Le premier article étudie comment pondérer les « scores (continus) » obtenus pour chaque image. Le second travail (Pevny and Nikolaev 2015) opère une détection en deux étapes, un histogramme des mêmes « scores » est d'abord construit puis un second classifieur est entraîné à identifier les lots d'images Cover. Plus récemment une étude relative à la non connaissance de la stratégie d'étalement du message a été proposé dans (Zakaria *et al*. 2019).

7.6.7. *La stéganalyse universelle*

Le dernier problème que nous souhaitons aborder est celui de la stéganalyse universelle. Là encore il s'agit d'un cadre plus général que celui de la décision binaire ciblé, qui comme nous l'avons exposé permet davantage d'évaluer une méthode précise de stéganographie. En pratique, il est intéressant de concevoir des méthodes de stéganalyse opérationnelles lorsque ni la taille du message ni l'algorithme d'insertion ne sont connus. La stéganalyse sans connaissance de la taille du message fait référence à des approches dites « quantitatives », c'est-à-dire visant à estimer la taille du message inséré (Pevný *et al*. 2009). L'étude de la performance du classifieur d'ensemble (Cogranne and Fridrich 2015) dans ce contexte où la taille du message est inconnue montre notamment que ce dernier (Kodovskỳ *et al*. 2012) ne peut être utilisé directement, car le seuil de détection est fixé en fonction des images stéganographiées utilisées durant l'apprentissage. L'approche proposée dans (Pevný *et al*. 2009) consiste essentiellement à utiliser des images contenant des messages de tailles différentes durant l'apprentissage.

De façon analogue, la stéganalyse multi-classes a été très peu étudiée. Là encore l'extension du classifieur d'ensemble a été envisagée à cette fin (Cogranne and Fridrich 2015) en utilisant deux approches simples et relativement efficaces ; la première dite « media Cover *vs.* le reste » consiste à créer une hypothèse alternative regroupant l'ensemble des algorithmes d'insertion possible et donc, au final, de réaliser un apprentissage statistique visant à détecter n'importe laquelle des méthodes de stéganographie. La seconde approche consiste au contraire à considérer $H + 1$ hypothèses, correspondants aux H algorithmes d'insertion considérés auxquels est ajouté le cas d'images Cover. Pour détecter la stéganographie et identifier la méthode d'insertion il est possible de créer de nombreux classifieurs spécifiquement entraînés à distinguer deux hypothèses. L'analyse d'une image nécessite alors d'utiliser tous les classifieurs pour ensuite choisir l'hypothèse qui regroupe le plus de « votes » de la part de l'ensemble de tous les classifieurs.

7.7. Conclusions

Nous avons présenté dans ce chapitre les différentes méthodes de stéganalyse. De façon générale, nous avons présenté les méthodes de détection « par signature » qui vise essentiellement à détecteur des traces liées à l'utilisation d'un logiciel spécifique. Très efficaces, ces méthodes ne sont guère généralisables et peu intéressantes sur le plan méthodologique.

Nous avons également étudié les méthodes de stéganalyse statistique, qui nécessitent de disposer d'un modèle précis de distribution statistique des images Cover et Stégo. Ces méthodes ne sont pas très efficaces en terme de détection, mais permettent de comprendre comment nous pouvons procéder pour faire de la stéganalyse. Cela permet notamment d'améliorer la stégananalyse, voir de concevoir une méthode de stéganographie.

Enfin, nous avons vu que sur le plan pratique les méthodes les plus efficaces reposent sur l'apprentissage statistique. Cela peut se faire par extraction de caractéristiques puis classification. Récemment, les méthodes reposant sur des réseaux de neurones profonds ont permit d'améliorer de façon considérable les performances des méthodes de stéganalyse, au prix d'une complexité de calculs accrue.

Nous avons également vu que la stéganalyse est un domaine en constante évolution. Nous avons en particulier insisté sur le fait que la plupart des travaux académiques sont difficilement exploitables en pratique. Les principaux verrous qui restent à étudier ont notamment été décrits. Dans ce domaine, les récents « challenges » de stéganalyse que nous avons organisé en 2019 (Cogranne *et al*. 2019) et en 2020 (Cogranne *et al*. 2020*a*) ont permit de faire des progrès importants et auront certainement des conséquences sur les travaux des prochaines années.

7.8. Bibliographie

Abdulrahman, H., Chaumont, M., Montesinos, P., Magnier, B. (2016), Color image steganalysis based on steerable gaussian filters bank, *in* Proc. ACM Workshop on Information Hiding and Multimedia Security, ACM, pp. 109–114.

Auguste, K. (1883), La cryptographie militaire, *Journal des sciences militaires*, 9(538), 5.

Bas, P., Filler, T., Pevný, T. (2011), Break our steganographic system — the ins and outs of organizing boss, *in* Proc. Information Hiding, Lecture Notes in Computer Science, LNCS, Prague, Czech Republic, pp. 59–70.

Bengio, Y., Courville, A. C., Vincent, P. (2013), Representation Learning : A Review and New Perspectives, *IEEE Transaction on Pattern Analysis and Machine Intelligence, PAMI*, 35(8), 1798–1828.

Boroumand, M., Chen, M., Fridrich, J. (2019), Deep Residual Network for Steganalysis of Digital Images, *IEEE Trans. on Information Forensics and Security*, 14(5), 1181 – 1193.

Butora, J., Fridrich, J. (2020), Reverse jpeg compatibility attack *(available as Early Access)*, *IEEE Trans. on Information Forensics and Security*, 15, 1444–1454.

Chaumont, M. (2020), Deep Learning in steganography and steganalysis, *in* M. Hassaballah, (ed.), Digital Media Steganography : Principles, Algorithms, Advances, Elsevier, chapter 14, pp. 321–349.
URL: *http ://arxiv.org/abs/1904.01444*

Chen, M., Sedighi, V., Boroumand, M., Fridrich, J. (2017), JPEG-Phase-Aware Convolutional Neural Network for Steganalysis of JPEG Images, *in* Proc. ACM Workshop on Information Hiding and Multimedia Security, IH&MMSec'2017, Drexel University in Philadelphia, PA, pp. 75–84.

Cogranne, R. (2011), Détection statistique d'informations cachées dans une image naturelle à partir d'un modèle physique, PhD thesis, Troyes University of Technology (UTT).

Cogranne, R. (2020*a*), Selection-channel-aware reverse jpeg compatibility for highly reliable steganalysis of jpeg images, *in* ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 2772–2776.

Cogranne, R. (2020*b*), Selection-channel-aware reverse jpeg compatibility for highly reliable steganalysis of jpeg images, *in* (*submitted*).

Cogranne, R., Fridrich, J. (2015), Modeling and extending the ensemble classifier for steganalysis of digital images using hypothesis testing theory, *Information Forensics and Security, IEEE Trans. on (in press)*, .

Cogranne, R., Giboulot, Q., Bas, P. (2019), The alaska steganalysis challenge : A first step towards steganalysis *into the wild*, *in* Proc. ACM Workshop on Information Hiding and Multimedia Security, IH&MMSec'19, ACM, New York, NY, USA, pp. 125–137.

Cogranne, R., Giboulot, Q., Bas, P. (2020*a*), Challenging academic research on steganalysis with realistic images, *in* IEEE Intl' Workshop on Information Forensics and Security (WIFS), *to be published*, IEEE.

Cogranne, R., Giboulot, Q., Bas, P. (2020*b*), Steganography by Minimizing Statistical Detectability : The cases of JPEG and Color Images., *in* ACM Information Hiding and MultiMedia Security (IH&MMSec), Denver, CO, United States.

Cogranne, R., Sedighi, V., Fridrich, J. (2017), Practical strategies for content-adaptive batch steganography and pooled steganalysis, *in* 2017 IEEE Intl' Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 2122–2126.

Cogranne, R., Sedighi, V., Fridrich, J., Pevný, T. (2015), Is ensemble classifier needed for steganalysis in high-dimensional feature spaces ?, *in* Information Forensics and

Security (WIFS), IEEE 7th Intl' Workshop on, pp. 1–6.

Denemark, T., Boroumand, M., Fridrich, J. (2016), Steganalysis Features for Content-Adaptive JPEG Steganography, *IEEE Trans. on Information Forensics and Security*, 11(8), 1736–1746.

Denemark, T., Sedighi, V., Holub, V., Cogranne, R., Fridrich, J. (2014), Selection-channel-aware rich model for steganalysis of digital images, *in* Information Forensics and Security (WIFS), 2014 IEEE 6th Intl' Workshop on, pp. 48–53.

Fridrich, J., Kodovskỳ, J. (2012), Rich models for steganalysis of digital images, *Information Forensics and Security, IEEE Trans. on*, 7(3), 868 –882.

Fuji-Tsang, C., Fridrich, J. J. (2018), Steganalyzing Images of Arbitrary Size with CNNs, *in* Proc. IS&T Electronic Imaging, Burlingame, California, USA, pp. 121(1)–121(8).

Giboulot, Q., Cogranne, R., Borghys, D., Bas, P. (2020), Effects and solutions of cover-source mismatch in image steganalysis, *Signal Processing : Image Communication*, 86, 115888.
URL: *http ://www.sciencedirect.com/science/article/pii/S0923596520300941*

Goljan, M., Fridrich, J. (2015), Cfa-aware features for steganalysis of color images, *in* Media Watermarking, Security, and Forensics 2015, vol. 9409, Intl' Society for Optics and Photonics, p. 94090V.

Goljan, M., Fridrich, J., Cogranne, R. (2014), Rich model for steganalysis of color images, *in* 2014 IEEE Intl' Workshop on Information Forensics and Security (WIFS), IEEE, pp. 185–190.

He, K., Zhang, X., Ren, S., Sun, J. (2014), Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition, *in* Proc. of the European Conference on Computer Vision, ECCV'2014, Zurich, Switzerland, pp. 346–361.

He, K., Zhang, X., Ren, S., Sun, J. (2016), Deep Residual Learning for Image Recognition, *in* Proc. of IEEE Conference on Computer Vision and Pattern Recognition, CVPR'2016, Las Vegas, Nevada, pp. 770–778.

Hinton, G. E., Salakhutdinov, R. R. (2006), Reducing the Dimensionality of Data with Neural Networks, *Science*, 313(5786), 504–507.

Holub, V., Fridrich, J. (2015), Low-complexity features for jpeg steganalysis using undecimated dct, *Information Forensics and Security, IEEE Trans. on*, 10(2), 219–228.

Ker, A. D., Pevný, T. (2012), Batch steganography in the real world, *in* Proc. of the on Multimedia and Security, IH&MMSec 12, ACM, New York, NY, USA, pp. 1–10.

Ker, A. D., Pevný, T. (2014), The steganographer is the outlier : realistic large-scale steganalysis, *Information Forensics and Security, IEEE Trans. on*, 9(9), 1424–1435.

Kodovskỳ, J., Fridrich, J., Holub, V. (2012), Ensemble classifiers for steganalysis of digital media, *Information Forensics and Security, IEEE Trans. on*, 7(2), 432–444.

Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012), ImageNet Classification with Deep Convolutional Neural Networks, *in* Proceeding of Advances in Neural Information Processing Systems 25, NIPS'2012, Curran Associates, Inc., Lake Tahoe, Nevada, USA, pp. 1097–1105.

LeCun, Y., Bengio, Y., Hinton, G. (2015), Deep learning, *Nature*, 521(7553), 436–444.

Lehmann, E., Romano, J. (2005), *Testing Statistical Hypotheses, Second Edition*, 3rd edition, Springer.

Pevný, T., Bas, P., Fridrich, J. (2010), Steganalysis by subtractive pixel adjacency matrix, *IEEE Trans. Inform. Forensics and Security*, 5(2), 215–224.

Pevný, T., Fridrich, J., Ker, A. D. (2009), From blind to quantitative steganalysis, *in* Proc. IS&T Electronic Imaging.

Pevný, T., Ker, A. D. (2015), Tow²ysis, *in* Proc. SPIE, vol. 9409, pp. 94090I–94090I–14.

Pevny, T., Nikolaev, I. (2015), Optimizing pooling function for pooled steganalysis, *in* Proc. IEEE Intl' Workshop on Information Forensics and Security, pp. 1–6.

Qian, Y., Dong, J., Wang, W., Tan, T. (2015), Deep Learning for Steganalysis via Convolutional Neural Networks, *in* Proc. IS&T Electronic Imaging, vol. 9409, San Francisco, California, USA.

Sedighi, V., Cogranne, R., Fridrich, J. (2016), Content-adaptive steganography by minimizing statistical detectability, *IEEE Trans. on Information Forensics and Security*, 11(2), 221–234.

Sedighi, V., Fridrich, J. (2015), Effect of Imprecise Knowledge of the Selection Channel on Steganalysis, *in* Proc. ACM Workshop on Information Hiding and Multimedia Security, IH&MMSec'2015, Portland, Oregon, USA, pp. 33–42.

Sedighi, V., Fridrich, J. J., Cogranne, R. (2016), Toss that bossbase, alice !, *in* Media Watermarking, Security, and Forensics, pp. pp. 1–9.

Sharma, G., Bala, R. (2017), *Digital color imaging handbook*, CRC press.

Simmons, G. (1984), The prisoners' problem and the subliminal channel, *in* Proc. of CRYPTO '83, Plenum Press, pp. 51–67.

Song, X., Liu, F., Yang, C., Luo, X., Zhang, Y. (2015), Steganalysis of adaptive jpeg steganography using 2d gabor filters, *in* Proc. ACM Workshop on Information Hiding and Multimedia Security, ACM, pp. 15–23.

Taburet, T., Filstroff, L., Bas, P., Sawaya, W. (2018), An Empirical Study of Steganography and Steganalysis of Color Images in the JPEG Domain, *in* IWDW, International Workshop on Digital Forensics and Watermarking, Jeju, South Korea.
URL: *https ://hal.archives-ouvertes.fr/hal-01904482*

Westfeld, A. (2001), F5–a steganographic algorithm, *in* Proc. Information Hiding, vol. 2137 of *Lecture Notes in Computer Science*, LNCS, pp. 289–302.

Westfeld, A., Pfitzmann, A. (1999), Attacks on steganographic systems, *in* Proc. Information Hiding, pp. 61–76.

Xu, G. (2017), Deep Convolutional Neural Network to Detect J-UNIWARD, *in* Proc. ACM Workshop on Information Hiding and Multimedia Security, IH&MMSec'2017, Drexel University in Philadelphia, PA, pp. 67–73.

Xu, G., Wu, H. Z., Shi, Y. Q. (2016), Structural Design of Convolutional Neural Networks for Steganalysis, *IEEE Signal Processing Letters*, 23(5), 708–712.

Ye, J., Ni, J., Yi, Y. (2017), Deep Learning Hierarchical Representations for Image Steganalysis, *IEEE Trans. on Information Forensics and Security, TIFS*, 12(11), 2545–2557.

Yedroudj, M., Chaumont, M., Comby, F. (2018), How to Augment a Small Learning Set for Improving the Performances of a CNN-Based Steganalyzer ?, *in* Proc. IS&T Electronic Imaging, Burlingame, California, USA, p. 7.

Yedroudj, M., Comby, F., Chaumont, M. (2018), Yedroudj-Net : An Efficient CNN for Spatial Steganalysis, *in* Proc. of IEEE Intl' Conference on Acoustics, Speech and Signal Processing, Calgary, Alberta, Canada, pp. 2092–2096.

Yousfi, Y., Butora, J., Fridrich, J., Giboulot, Q. (2019), Breaking alaska : Color separation for steganalysis in jpeg domain, *in* Proc. ACM Workshop on Information Hiding and Multimedia Security, ACM, pp. 138–149.

Zakaria, A., Chaumont, M., Subsol, G. (2019), Pooled Steganalysis in JPEG : how to deal with the spreading strategy ?, *in* IEEE International Workshop on Information Forensics and Security, WIFS 2019, Delft, The Netherlands, December 9-12, 2019, IEEE, pp. 1–6.
URL: *https ://doi.org/10.1109/WIFS47025.2019.9035096*

Zeng, J., Tan, S., Li, B., Huang, J. (2017), Pre-Training via Fitting Deep Neural Network to Rich-Model Features Extraction Procedure and its Effect on Deep Learning for Steganalysis, *in* Proceedings of Media Watermarking, Security, and Forensics 2017, MWSF'2017, Part of IS&T Symposium on Electronic Imaging, EI'2017, Burlingame, California, USA, p. 6.

Zeng, J., Tan, S., Li, B., Huang, J. (2018), Large-Scale JPEG Image Steganalysis Using Hybrid Deep-Learning Framework, *IEEE Trans. on Information Forensics and Security*, 13(5), 1200–1214.

Zhang, R., Zhu, F., Liu, J., Liu, G. (2020), Depth-wise separable convolutions and multi-level pooling for an efficient spatial cnn-based steganalysis, *IEEE Trans. on Information Forensics and Security*, 15, 1138–1150.