REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université de Mohamed El-Bachir El-Ibrahimi - Bordj Bou Arreridj

Faculté *des Sciences et de la technologie*

Département électronique

Mémoire

Présenté pour obtenir

LE DIPLOME DE MASTER

FILIERE : télécommunication

Spécialité : système de télécommunication

Par

- **Benkhelfallah Wissem**
- **Beghoura lazhar ihab eddine**

Intitulé

Implémentation d'un algorithme de tatouage d'image en utilisant la technique PVO-RDH (Pixel-Value-Ordering-Reversible –Data-Hiding)

Soutenu le :29.06.2022

Devant le Jury composé de :

Nom & Prénom	*Grade*	*Qualité*	*Etablissement*
M.FLISSI Mustapha	*MCA*	*Président*	*Univ-BBA*
M. ROUABAH Khaled	*Prof*	*Encadreur*	*Univ-BBA*
Mme. MELIZI Nora	*MCA*	*Examinateur*	*Univ-BBA*
Mlle. FENENICHE Wafa	*Docteur*	*Invité*	*Univ-BBA*
M. ABED Terek	*MCB*	*Co-Encadreur*	*Univ-BBA*

Année Universitaire 2021/2022

Remercîment

En premier lieu, nous remercions ALLAH le tout puissant pour la force, la santé et la sagesse qu'il nous a toujours donné et que sans lui rien n'aurait été accompli.

Nous aimerions exprimer notre immense gratitude et notre plus profond respect à notre maître et encadreur ***Pr. ROUABAH Khaled***. Il a été pour nous un guide, un exemple et un mentor pendant notre cursus et durant l'accomplissement de ce mémoire. Les valeurs qu'il nous a inculqués et le savoir qu'il nous a transmis ont été inestimables.

Nous aimerions exprimer notre immense gratitude et notre plus profond respect aussi à notre maître et co-encadreur ***M. ABED Tarek***, Enseignant au niveau de l'université de Bordj Bou Arreridj pour la qualité des conseils, des aides et des soutiens consenties de sa part.

Nous aimerions également exprimer notre immense gratitude et notre plus profond respect à notre maître ***Dr. FENENICHE Wafa***. Un merci sincère pour ses conseils et ses aides précieuses. Nous ne saurons trop vivement le remercier pour le temps précieux qu'il a pu sacrifier au suivi de ce projet.

Nous adressons nos sincères remerciements aux membres du jury pour avoir pris le temps d'examiner et juger ce travail.

Nous voudrons exprimer finalement notre reconnaissance envers nos amis et collègues ***Mr. BOUDJELAL Ammar Ayoub*** et ***Mlle. BEN ARROUDJ Chahinez*** et qui nous ont apporté leur soutien moral et intellectuel tout au long de nos démarches.

Dédicace

Je dédie ce mémoire :

*A ma **chère maman** 'HARICHE Sonia' qui m'a soutenu et encouragé durant ces années d'études. Qu'elle trouve ici le témoignage de ma profonde reconnaissance.*

*A mon **cher père** 'BENKHELFALLAH Abdel Rezak' pour son encouragement permanent, et son soutien moral.*

*A **mes sœurs**, et **mon frère** et Ceux qui ont partagé avec moi tous les moments d'émotion lors de la réalisation de ce travail. Ils m'ont chaleureusement supporté et encouragé tout au long de mon parcours.*

*A **ma famille**, **mes proches** et à ceux qui me donnent de l'amour et de la vivacité.*

*A tous **mes amis** qui m'ont toujours encouragé, et à qui je souhaite plus de succès.*

BENKHELFALLAH Wissem.

*A mes **chers parents** 'Samira et Hellal' Pour tous leurs sacrifices, leur amour, leur tendresse, leur soutien et leurs prières tout au long de mes études.*

*A mes **chers frères** 'Madjid, Islam, zaki, fares ,abesse, ishak et abdeljalile ', et **mes petites sœurs** ' marwa, neila, asma, chaima, ansar et safa) pour leur appui et leur encouragement.*

*A toute **ma famille** pour le soutien tout au long de mon parcours universitaire.*

*A **toute personne** qui a contribué à la réussite de ce travail Que ce travail soit l'accomplissement de vos vœux tant allégués, et le fruit de votre soutien infaillible.*

BEGHOURA Lazher Ihab Eddine.

ملخص:
تعد بيانات المعلومات حاليًا أقوى سلاح في العالم. وبالفعل تم ابتكار طرق جديدة لحمايته، وابتكر البعض الآخر لحجبه وتشويهه. لذلك، تم اقتراح عدة طرق أخرى في الأدبيات العلمية لتعزيز أمن بيانات هذه المعلومات. من بين هذه الطرق، تعد تقنية PVO RDH تقنية مهمة جدًا لتكامل البيانات توفر أداءً أفضل. وهو يتألف من تقسيم الصورة إلى مجموعة من الكتل من نفس الحجم، وترتيب وحدات البيكسل وفقًا لقيم مستوى الرمادي الخاصة بها ثم دمج البيانات (في ظل ظروف معينة) في البيكسل الأول والأخير من هذه الكتل المرتبة. في مشروع الماجستير هذا، قمنا بدراسة وتنفيذ طريقة PVO-RDH في بيئة Python. وقد أثبتت نتائج التنفيذ والمحاكاة فاعلية هذه الطريقة في حماية المعلومات من جميع أشكال القرصنة.

كلمات مفتاحية: المعلومة، برامج، التطور التكنولوجي، القرصنة، الرقمنة، الخوارزميات.

Résumé
Actuellement, l'information est une arme plus puissante dans le monde. En effet, de nouvelles méthodes ont été créées pour la protéger, et d'autres ont été inventées pour la bloquer et la déformer. Par conséquent, plusieurs autres méthodes ont été proposées dans la littérature scientifique pour renforcer la sécurité de ces données. Parmi ces méthodes, la technique de PVO RDH (Pixel-Value-Ordering Reversible Data Hiding) est une technique d'intégration de données (tatouage) très importante qui donne de meilleures performances. Elle consiste à diviser une image en un ensemble de blocs de même taille, d'en ordonner les pixels selon leurs valeurs de niveau de gris et d'intégrer ensuite des données (sous certaines conditions) dans les, premier et dernier, pixels de ces blocs ordonnés. Dans ce travail de Master, nous avons étudié et implémenté sous environnement Python la méthode PVO-RDH. Les résultats de la programmation et de la simulation ont prouvé l'efficacité de cette méthode pour protéger les informations contre toutes les formes de piratage.

Mots-clés: Reversible data hiding, Pixel-value-ordering, Pairwise prediction-error expansion, Reversible 2D mapping.

Abstract
Currently, information data is the most powerful weapon in the world. Indeed, new methods have been created to protect it, and others have been invented to block and deform it. Therefore, several other methods have been proposed in the scientific literature to enhance the security of these information's data. Among these methods, the PVO RDH (Pixel-Value-Ordering Reversible Data Hiding) technique is a very important data integration (watermarking) technique that gives better performance. It consists of dividing an image into a set of blocks of the same size, ordering the pixels according to their gray level values and then integrating data (under certain conditions) into the first and last pixels of these ordered blocks. In this Master's project, we have studied and implemented the PVO-RDH method in a Python environment. The results of implementation and simulation have proven the effectiveness of this method to protect information against all forms of piracy.

Key-words: Information, software, technological development, hacking, digitization, algorithm

Table des matières

CHAPITRE II : Principe des méthodes DE et ses extensions et PVO

CHAPITRE III : Implémentation sur Python, Résultats et Discussions

Table des figures

CHAPITRE I : Principe et Etat de l'art sur la dissimulation des données

CHAPITRE II : Principes des méthodes DE et ses extensions et PVO

CHAPITRE III : Implémentation sur Python, Résultats et discussions

Liste des tableaux

Liste des abréviations

DH	Dissimulation De Donnée.
RDH	Dissimulation De Donnée Réversible.
PVO	L'ordonnancement de valeur de pixel.
PEE	Prédiction D'erreur D'Expansion.
PSNR	Rapport Signal /bruit de Crête.
CE	Capacité D'intégration.
MSE	Erreur Quadratique Moyenne.
NHIC	Coefficient D'intersection D'histogramme.
BC	Coefficient De Bhattacharyya.
MAE	Erreur Absolue Moyenne.
UIQI	Indice Universel De Qualité.
DE	Déférence D'expansion.
LM	Location Mapp.

Introduction Générale

Introduction Générale

La dissimulation des données est une nouvelle technologie qui intègre des données d'identification utiles directement dans des supports multimédias. Bien que les données intégrées soient ajoutées, cette nouvelle technologie peut toujours maintenir l'effet visuel premium sans affecter les performances des supports multimédias. Les avantages de la dissimulation de données comprennent principalement son imperceptibilité, sa robustesse et sa grande capacité [1].

Les techniques traditionnelles de dissimulation de données se préoccupent principalement de la robustesse des données intégrées contre diverses attaques. Parmi ces dernières, celles basées sur la dissimulation réversible des données (RDH) ont été proposées pour récupérer le contenu original à partir des données intégrées et du support marqué sans aucune distorsion.

Dans certains scénarios sensibles, tels que l'imagerie médicale, la télédétection et l'imagerie militaire, il est strictement interdit de mal interpréter le contenu numérique. En effet, le contenu original ainsi que les données intégrées sont tous les deux nécessaires [2].

La RDH, similaire à la compression sans perte, est un procédé qui vise à transformer le contenu original d'une image en une représentation de taille réduite dans un premier temps, puis d'utiliser l'espace économisé pour cacher le message secret. La principale différence par rapport aux méthodes classiques est que le processus de transformation dans la méthode RDH doit préserver l'apparence visuelle de l'image [3].

De nombreux algorithmes de dissimulation de données réversibles ont été proposés jusqu'à présent. Parmi ces derniers, l'expansion de la différence de Tian (DE) a suscité une grande attention de la part des chercheurs. L'idée de base de la DE est d'utiliser la différence entre les pixels au lieu du pixel lui-même pour intégrer les données. Par rapport à la méthode de dissimulation réversible des données basée sur la compression, la méthode de Tian peut offrir une capacité d'intégration (CE) beaucoup plus élevée tout en maintenant une faible distorsion. Par la suite, la technique DE a été largement étudiée et développée sous de nombreux aspects. Il existe trois extensions majeures de la DE. La première tente de généraliser la DE en transformation de plancher. La 2ème extension tente de réduire la taille de la carte de localisation dans la méthode DE. Dans la troisième, appelée expansion de l'erreur de prédiction (PEE), la valeur de la différence est remplacée par l'expansion de l'erreur de prédiction dans l'opération d'intégration [4].

Dans ce travail de Master, on va valider, par implémentation sous environnement Python, une nouvelle technique de traitement d'image, notée PVO-RDH, basée sur la combinaison de deux techniques qui sont la PVO et la PEE. Ces deux techniques permettent de divisée l'image hôte en blocs de tailles égales sans chevauchement. Les valeurs maximales et minimales de chaque bloc sont prédites afin d'améliorer le niveau du PSNR et augmenter par conséquent la capacité d'intégration.

Le présent manuscrit est structuré autour de trois chapitres :

Le premier chapitre sera consacré à donner un petit rappel sur Data Hiding et un état de l'art sur les méthodes de la dissimulation de donnée réversible.

Dans le chapitre deux, on présente la méthode DE et ses trois extensions. Puis, on va détailler la méthode PVO-RDH.

Dans le dernier chapitre, on présente le langage python avec ses outils de programmation. Comme ce dernier va être utilisé pour implémenter cette technique, on explique avant ça les différentes étapes de l'algorithme de construction du code PVO-RDH pour les deux côtés intégration et extraction. Enfin, on terminera ce mémoire par présenter les résultats de simulation basés sur la mesure de la CE et du niveau PSNR avec une étude comparative par rapport à d'autres méthodes.

On terminera ce travail par une conclusion générale et des perspectives.

Chapitre I

Principe et Etat de l'art sur la Dissimilation des Données

I.1 Introduction

Le multimédia numérique est devenu une partie importante de la vie moderne avec la disponibilité d'internet, des appareils électroniques et l'avancement rapide des systèmes de communication en réseau. Le contenu multimédia numérique (image, audio, vidéos) est largement utilisé dans divers domaines tels que le commerce électronique, la sécurité nationale, la télémédecine et les communications réseau [5].

La sécurité des données consiste à protéger ces informations numériques contre les accès non autorisés, les dommages ou le vol tout au long de leur cycle de vie. C'est un concept solide qui peut protéger les actifs informationnels d'une organisation contre les activités cybercriminelles mais également contre les menaces internes et les erreurs humaines [6].

Les algorithmes de dissimulation de donnée réversible RDH visent à transférer en toute sécurité des données secrètes sur des supports de communication (exemple : image, texte, ….etc.). Pour ce faire, les données secrètes sont intégrées dans le support de couverture qui est également récupéré exactement à l'extrémité du récepteur avec les données secrètes [7].

Dans ce chapitre, nous présentons tout d'abord une description générale de la dissimulation de données DH (Data Hiding). Ensuite, nous montrons les différents domaines d'application. Finalement, nous dévoilons un état de l'art sur les techniques du RDH ainsi que quelques critères de mesure de performances de ce type d'algorithmes.

I.2 Description générale du Data-Hiding

La dissimulation des données est un terme général qui englobe un large éventail des problèmes allant au-delà de l'intégration de messages dans le contenu [8]. Le DH traite le problème de la transmission des signaux, représentant généralement un message ou des données à transmettre, en utilisant comme média ou canal de communication un contenu hôte. Ce dernier peut prendre de nombreuses formes numériques notamment des vidéos, des images et des lettres d'un texte inoffensif utilisé par les services secrets [9]. Le DH se compose en deux catégories qui sont le DH réversible (RDH) et le DH non réversible. Dans notre travail de Master, on s'intéresse à la dissimulation des données RDH.

I.3 Principe du DH

Dans le processus DH, deux situations existent. En effet, dans la première, l'information à cacher doit être échangée discrètement sans être détectée par des inconnus. Ce cas peut être considéré comme une opération de stéganographie. La deuxième situation concerne la vérification par exemple d'un billet de banque en lisant le Watermark [9].

Le développement des systèmes DH a permis l'apparition de plusieurs techniques de construction du signal d'insertion et diverses autres techniques pour l'extraction de l'information. Ainsi, chaque technique est définie par le compromis entre ces différentes caractéristiques [9].

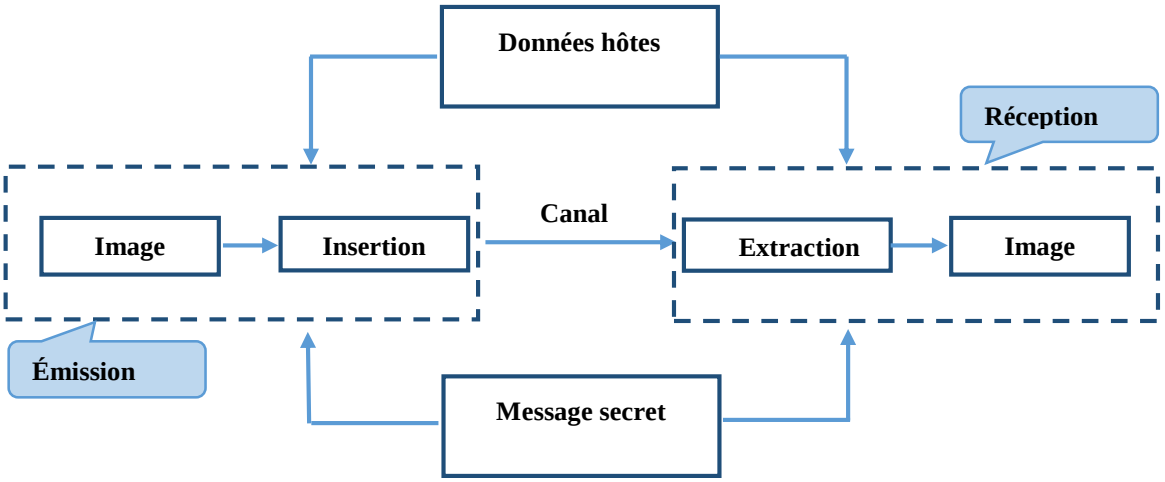


Figure. I. 1 Schéma général des étapes de transmission d'une image dans le domaine DH [8].

D'après la figure (I.1), l'image originale utilisée comme support pour transmettre (au niveau de l'émission) un message secret pour un utilisateur concerné (au niveau de la réception) doit suivre certaines étapes pour réaliser les opérations d'insertion au niveau de l'émission et d'extraction au niveau de la réception.

I.4 Caractéristiques du DH

Dans tous les schémas de dissimulation des données, il faut respecter un compromis entre leurs principales caractéristiques : robustesse, capacité, invisibilité et sécurité. Dans les sections suivantes, nous allons montrer brièvement ces dernières caractéristiques [9].

I.4.1 Capacité

C’est la quantité d'informations qui peut être cachée dans le support de couverture et le nombre moyen de bits qui peut être transmis par une image hôte. C’est également la taille du message qu’il est possible d’insérer sans erreur sur une longueur donnée du signal hôte pour une puissance d’attaque donnée [9].

I.4.2 Robustesse

Ce paramètre est réservé à la résistance de quelques traitements légaux notamment la compression, le recadrage, le changement du contraste...etc.... [9]. En effet, la robustesse se traduit par la capacité et la possibilité de récupérer l’information insérée même si l’image hôte a subi des modifications [8].

I.4.3 Invisibilité

Tout système de dissimulation d’information engendre inévitablement des modifications sur le document hôte. Ceci a un grand impact sur les caractéristiques perceptuelles et statistiques du signal original. Ainsi, le but de la majorité des systèmes de DH est d’impacter au minimum les caractéristiques du signal original [9].

I.4.4 Sécurité

La sécurité désigne l'incapacité du pirate à extraire ou de voir le contenu des informations cachées [5].

I.5 Domaines d’applications

Comme montre la figure (I.2), Il existe de nombreuses applications potentielles des techniques de dissimulation des données DH. Dans ce qui suit, nous résumons ceux les plus populaires.

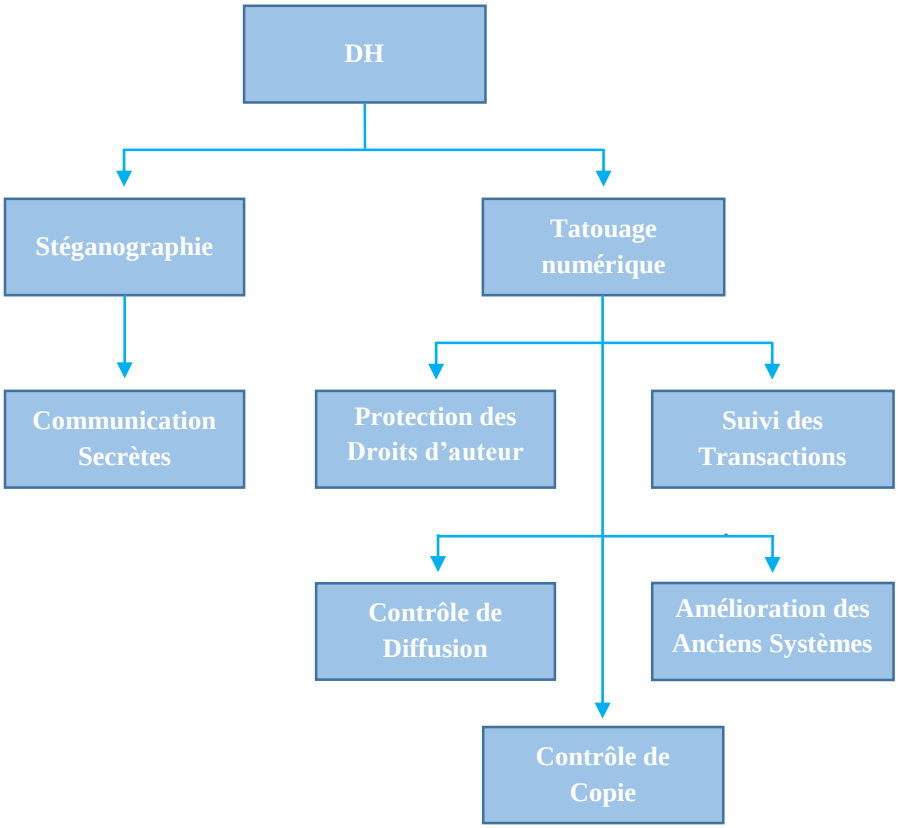


Figure.I. 2 Domaines d'application du Data Hiding [10].

I.5.1 Tatouage numérique

C'est l'application la plus connue du DH. D'ailleurs, le tatouage numérique (ou le marquage numérique, watermarking, filigranage…) est souvent utilisé comme le terme générique qui désigne tous les systèmes de DH. Il est défini comme l'ensemble des modifications robustes et invisibles dans un document hôte. Ces modifications permettent d'insérer une information qui concerne directement le document tatoué [9].

I.5.1.1 Applications du tatouage numérique

Une application du tatouage numérique peut avoir différents objectifs qui peuvent être des objectifs visible ou invisible [5]. Ces dernières sont données comme suit :

I.5.1.1.1 Protection des droits d'auteur

Dans le tatouage numérique, les données insérées dans une application, sont des informations sur le propriétaire légal ou le distributeur.

Elles sont utilisées pour notifier un utilisateur que l'article est protégé par des droits d'auteur, pour prouver la propriété de l'article, ou pour suivre des copies illégales de l'article [11].

I.5.1.1.2 Contrôle de diffusion

Le tatouage est considéré comme une solution pour fournir des services de surveillance de diffusion. Pour cette application, les informations insérées sont utilisées pour plusieurs buts liés à la diffusion des médias numériques (audio, vidéo). Les données insérées peuvent être utilisées pour vérifier si la diffusion réelle de contenu (par exemple des publicités) est conforme aux conditions convenues, c'est-à-dire, si le contenu a été diffusé au bon moment et pendant la bonne durée [11]. En plus, les informations insérées peuvent être utilisées pour calculer le taux de regarde/ écoute d'une certaine émission (mesure d'audience), ou pour la conception d'un système automatisé de collecte de redevances pour les données protégées par le droit d'auteur [11].

I.5.1.1.3 Suivi des transactions

Dans les applications de contrôle de diffusion et d'identification des propriétaires, le même watermark est inséré dans toutes les copies du même contenu. Cependant, dans l'application de suivi des transactions, un watermark unique est inséré dans chaque copie individuelle. Les watermarks utilisés dans ce cas sont souvent appelés empreintes digitales. En effet, ils permettent au propriétaire ou au distributeur de contenu d'identifier la source d'une copie illégale [12]. Dans ce cas, le watermark est inséré non seulement pour porter des informations sur le propriétaire ou le distributeur légal de l'article numérique, mais également pour marquer la copie de la transaction. [11].

I.5.1.1.4 Contrôle de copie

Le tatouage numérique est utilisé, au niveau du matériel, pour limiter la diffusion des données copiées. Dans ce scénario deux cas peuvent être envisagés :

1- Le tatouage numérique est présent dans le matériel fonctionne ;
2- Le tatouage numérique a disparu, donc la copie est illicite (moindre qualité du document) et le matériel refuse de fonctionner [8].

I.5.1.1.5 Amélioration des anciens systèmes

Dans cette application, le tatouage permet de développer la fonctionnalité des anciens systèmes. Les informations insérées peuvent être utilisées pour améliorer les fonctions ou les informations

fournies par les systèmes existants tout en préservant la compatibilité. Par exemple, on peut insérer dans une image numérique une adresse web URL (Uniform Resource Locator) liée à l'objet représenté. Ensuite, cette URL insérée peut être utilisée pour une connexion automatique à la page web correspondante. De plus, on peut insérer les dossiers des patients par stockage dans les images médicales radiographiques [13].

I.5.2 Stéganographie

La stéganographie (en anglais : steganography) est encore une technique peu connue du grand public. Le mot stéganographie vient de deux mots grecs, steganos signifiant "caché" ou "protégé" et graphein signifiant "écriture". Selon la définition de [14], la stéganographie est la dissimulation secrète des données dans un transporteur hôte donné pour échanger secrètement des informations. Le transporteur hôte est un message quelconque contenant une redondance ou une non-pertinence. Elle est la plus ancienne application du DH, puisqu'elle est retrouvée dans l'Antiquité avec l'histoire de Herodotus [15]. Comme le montre la figure (I.3), Elle sert généralement à communiquer secrètement à travers des réseaux publics. Ainsi, elle est souvent utilisée dans le domaine militaire ou le contre-espionnage [16].

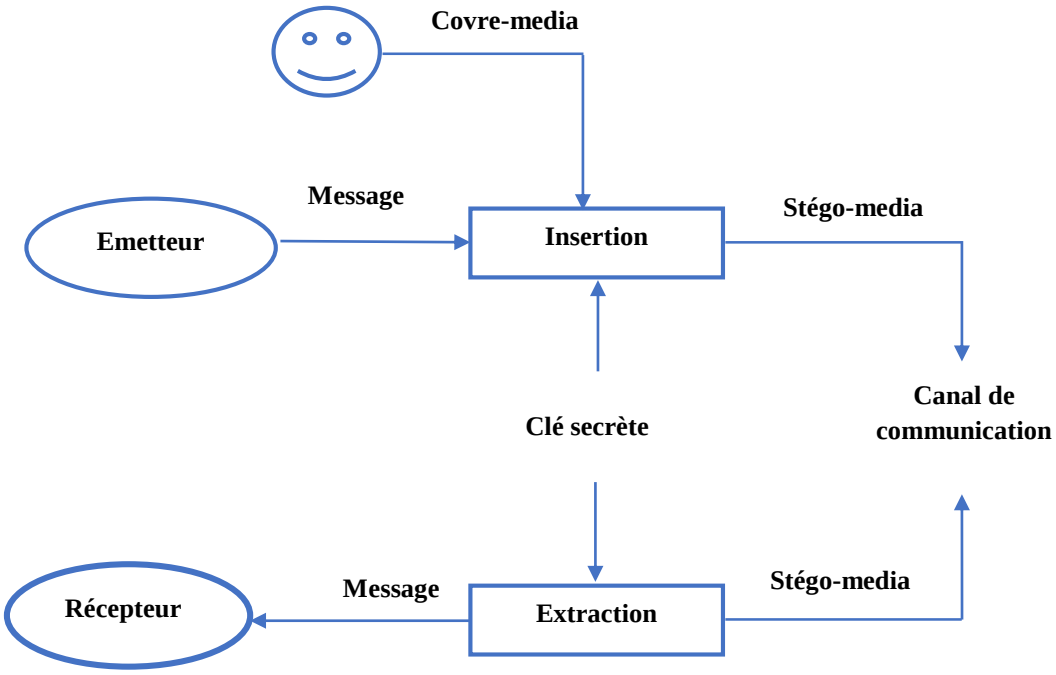


Figure.I. 3 Principe de base de la stéganographie [6].

D’après la figure (I.3), nous avons émetteur ‘’E’’ désirant envoyer un message secret à un récepteur ‘’R’’. Pour se faire, ‘’E’’, intègre dans un objet de couverture le message secret et obtient un objet-stégo qui sera ensuite envoyé par la voie publique.

I.5.2.1 Applications de la stéganographie

La stéganographie peut être utilisée pour la dissimulation des données hautement confidentielles ou interdites au grand public, par exemple dans le domaine militaire ou médical. Elle peut être aussi utilisée pour combattre l’espionnage industriel. En effet, toute entreprise a des secrets à protéger (information stratégique, formule chimique d’un nouveau produit, code source d’un logiciel propriétaire…). La stéganographie peut rendre le vol de ce genre d’information improbable voire impossible [17].

I.5.3 Comparaison entre Stéganographie et tatouage numérique

Une comparaison entre la stéganographie et le tatouage numérique, concernent le domaine RDH, est montrée dans le tableau (I.1).

Tableau. I. 1 Comparaison entre la Stéganographie et le tatouage numérique [8].

Stéganographie	**Tatouage numérique**
• L’existence d’un message caché doit rester secrète ; • Il est utilisé pour une communication secrète ; • Difficile à détecter et seul le récepteur peut le détecter.	• Existence d’un message connue doit rester secrète. • Il est utilisé pour la protection du contenu, protection des droits d’auteur, l’authentification du contenu ; • Possibilité de récupérer les données.

Enfin, la stéganographie et le tatouage numérique sont deux disciplines très proches l’une de l’autre puisque toutes les deux consistent à protège une donnée à caractère sensible [8].
Dans les sections suivantes, nous allons présenter un état de l’art sur les algorithmes RDH.

I.6 Etat de l’art sur les algorithmes RDH

Le RDH est un domaine confronté à un certain nombre de problèmes. En effet, le problème le plus pertinent est celui comment intégrer des données dans une image sans avoir une distorsion élevée sur l’image. De plus, le problème est d’avoir les solutions efficaces permettant de

récupérer les données et restaurer l'image originale. Comme il est résume dans la figure (I.4) Plusieurs algorithmes ont été proposés dans la littérature scientifique pour pallier à ces problèmes.

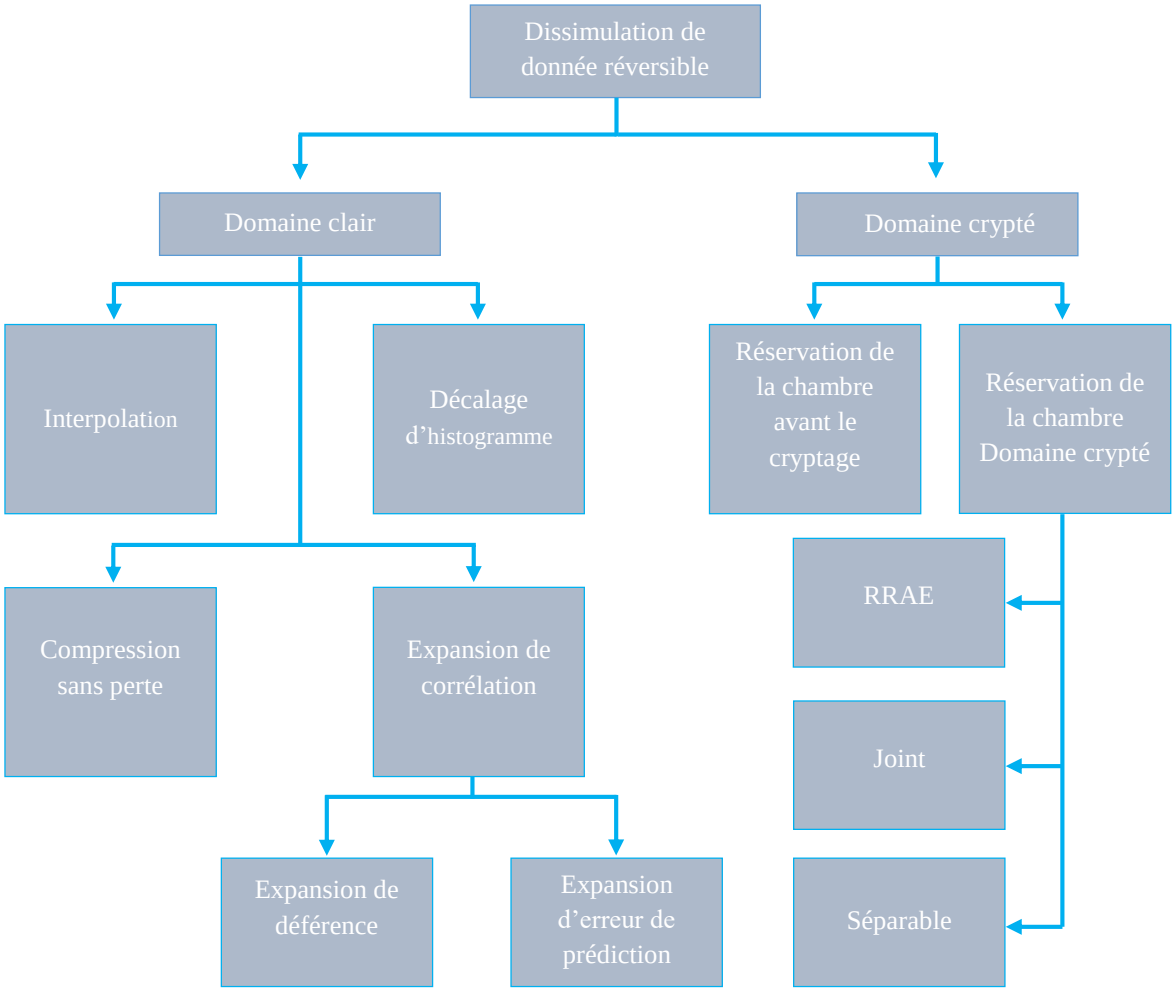


Figure. I. 4 Classification des méthodes RDH [7].

Les auteurs de la référence [7], ont classé ces algorithmes ont deux catégories. La première catégorie, notée par le domaine clair, regroupe tous les algorithmes qui permettent d'améliorer la capacité de la charge utile tout en garantissant une faible distorsion. La deuxième catégorie caractérise le domaine RDH des images cryptées. Dans notre travail on ne s'intéresse pas à cette catégorie. Concernant la première catégorie, le cadre général est illustré sur la figure (I.5).

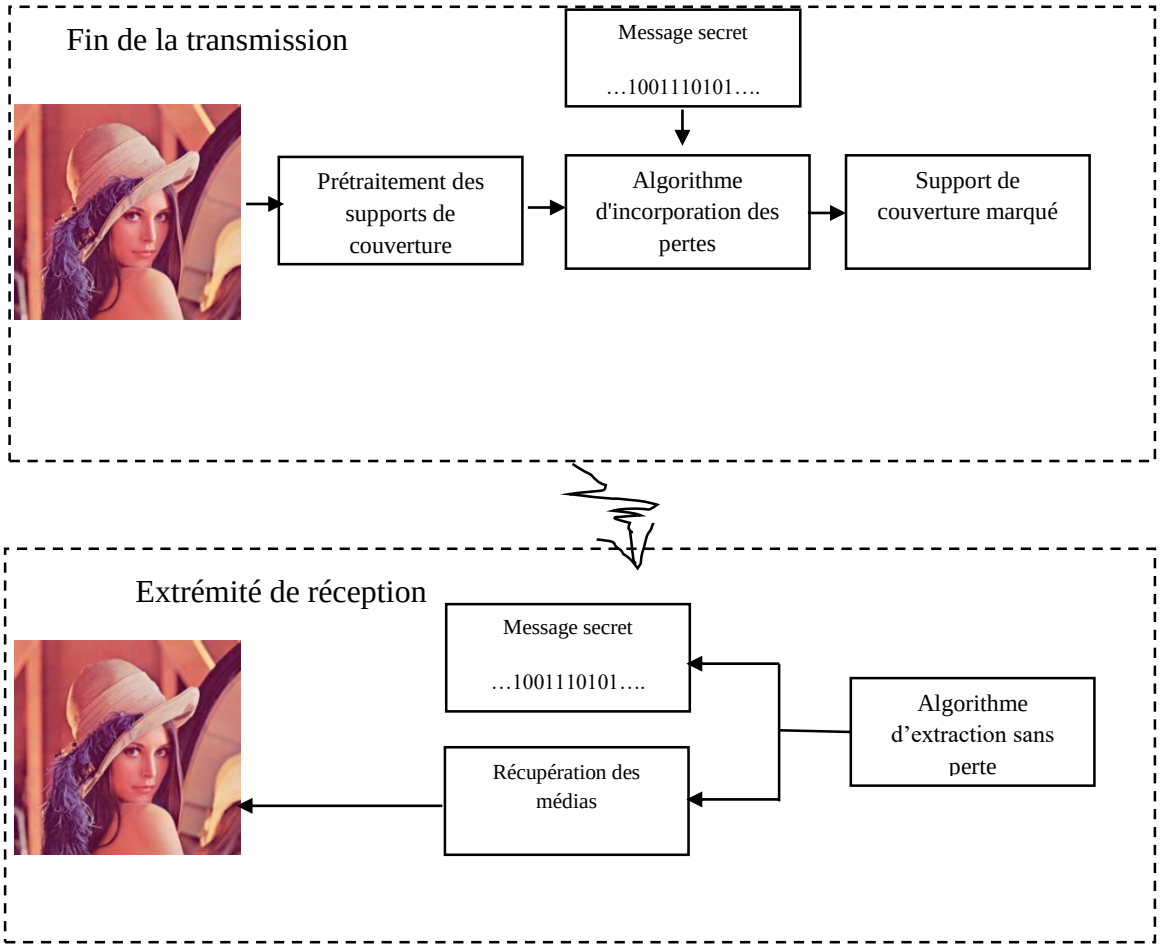


Figure. I. 5 Cadre général du RDH du domaine clair [7].

Les travaux dans cette première catégorie englobent les méthodes telles que la compression sans perte [32, 33, 34,35], l'expansion de corrélation [36-37,38], décalage d'histogramme [39,40,41,42-43], Interpolation [44, 45, 46,47].

Dans le cadre RDH du domaine clair, le support de couverture est d'abord prétraité afin de créer un espace pour l'intégration des données. Ensuite, les informations secrètes sont intégrées dans le support de couverture à l'aide d'un algorithme d'intégration sans perte pour obtenir un support de couverture marqué. L'expéditeur envoie le support marqué au destinataire à travers un canal de communication. Le destinataire récupère les informations secrètes ainsi que le support de couverture original en utilisant l'algorithme d'extraction des données. Dans la compression sans perte, une partie du support de couverture est compressée pour y intégrer des données. Ces algorithmes sont caractérisés par une plus grande puissance. De plus, une faible capacité d'intégration des données et sont donc utilisées pour l'authentification du support de

couverture. Il n'y a pas de perte de qualité mais la taille du support de couverture est réduite en raison de la compression. Ces algorithmes sont utilisés de manière limitée en raison de la faible compression, de la complexité du temps et de la procédure de décodage complexe. En revanche, les travaux d'expansion de corrélation peuvent être classés en expansion de différence [48-49] et expansion d'erreur de prédiction [50-51]. L'expansion de la différence utilise la corrélation de deux pixels voisins, alors que L'expansion d'erreur de prédiction prend en compte la corrélation locale de pixels voisins plus grands dans le support de couverture pour intégrer les données secrètes. Dans le décalage de l'histogramme, les informations secrètes sont intégrées dans la plus grande case de l'histogramme, ce qui permet d'obtenir une capacité d'intégration élevée et une faible complexité de calcul par rapport aux autres approches de compression sans perte [52, 53]. Dans les approches d'interpolation, les informations secrètes sont intégrées dans les pixels interpolés les pixels de support de couverture d'origine.

I.7 Critère de mesure des performances de l'RDH

Dans ce qui suit les critères de mesure des performances du domaine RDH seront décrits [6].

I.7.1 CE (Capacité d'intégration)

La capacité d'intégration peut être définie comme le rapport entre le nombre de bits pouvant être incorporés et le nombre total de bits. Cette analyse est essentiellement utilisée pour tester si une technique est capable d'incorporer une grande quantité des données ou non [6].

$$CE = \frac{Nombre\ de\ bits\ pouvant\ être\ intégrée}{Nombre\ totale\ de\ bits} \quad (1)$$

I.7.2 MSE (Erreur quadratique moyenne)

L'erreur quadratique moyenne est l'écart quadratique moyen entre une image de référence et une image déformée. Une technique de stéganographie image est efficace si elle donne MSE faible. Il est défini par la relation (2) [6].

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2 \quad (2)$$

Pour une image monochrome la taille est $m \times n$, et pour image couleur la taille est $m \times n \times 3$.

Où :

I : représente la valeur de pixel dans l'image original.

K : représente la valeur de pixel de l'image stégo.

I.7.3 PSNR (Rapport signal/bruit de crête)

C'est le rapport entre la puissance maximale possible d'un signal et la puissance du corrompre de bruit qui affecte la fidélité de sa représentation. Ce rapport est souvent utilisé comme une mesure de la qualité entre l'image originale et l'image stego. Plus de PSNR implique que la qualité de l'image compressée est meilleure. Le PSNR est défini comme :

$$PSNR = 10\log_{10}\left(\frac{{MAX_i}^2}{MSE}\right) \quad (3)$$

Où, MAX_ireprésente la valeur maximum du pixel de l'image. Dans les images avec des pixels ayant 8 bits par échantillon, sa valeur est de 255 [6].

I.7.4 Coefficient de corrélation

Ce paramètre est une mesure de la corrélation linéaire c'est-à-dire la dépendance entre deux images A et B. Sa gamme est comprise entre -1 et +1 les deux sont inclus, où 1 signifie le match parfait et −1 signifie décalage total. Le coefficient de corrélation peut être calculé comme suit :

$$\rho(A.B) = \frac{cov(A.B)}{\sigma_A \sigma_B} \quad (4)$$

Où, A est l'image de couverture et B est l'image stego, ρ(A, B) est le coefficient de corrélation entre les matrices d'image A et B, (A, B) est la covariance entre les matrices A et B, σA, σB sont l'écart-type A et B [6].

I.7.5 NHIC (Coefficient d'intersection d'histogramme normalisé)

Ce paramètre donne le compte de la même valeur des pixels entre les deux histogrammes. Si la distribution de probabilité des deux images est prise comme P et Q respectivement, donc cette mesure est donnée par [6] :

$$I(A.B) = \sum_{i=1}^{n} \min(P(i), Q(i)) \quad (5)$$

Où, A est l'image de couverture et B est l'image stego. L'échelle de valeur de ce coefficient est comprise entre 0 à 1. Où 0 représente le décalage et 1 représente la correspondance exacte.

I.7.6 BC (Coefficient de Bhattacharyya)

Ce paramètre mesure la proximité relative entre deux échantillons statistiques qui sont deux images dans ce cas. Le coefficient de Bhattacharyya est donné par l'équation (6), où *P* et *Q* sont des distributions de probabilité des deux images (image de couverture et image de stego) [6] :

$$BC(A.B) = \sum_{i=1}^{n} \sqrt{P(i)Q(i)} \tag{6}$$

I.7.7 MAE (Erreur absolue moyenne)

Représente l'erreur moyenne absolue entre la stego image et l'image originale. Il est défini par la relation suivante :

$$MAE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} |f(i,j) - y(i,j)| \tag{7}$$

Dans la formule ci-dessus, l'erreur moyenne absolue est une valeur moyenne des erreurs absolues. Où f est la valeur de pixel de l'image originale et y est la valeur réelle de l'image stego. La taille de l'image monochrome est m × n, et les images couleurs $m \times n \times 3$ [6].

I.7.8 UIQI (Indice universel de qualité d'image)

Dans une image, les valeurs des pixels disponibles à différentes positions montrent différents effets sur le système visuel humain (HVS). Si des distorsions ou des changements sont introduits dans l'image, une telle distorsion dans l'image est calculée comme une combinaison de trois facteurs : perte de corrélation, distorsion contrastée et distorsion de luminance [6].

$$L(A.B) = \frac{2\mu_A\mu_B}{\mu^2{}_A\mu^2{}_B} \tag{8}$$

$$C(A.B) = \frac{2\sigma_A\sigma_B}{\sigma^2{}_A\sigma^2{}_B} \tag{9}$$

$$S(A.B) = \frac{2\sigma_{AB}}{\sigma_A + \sigma_B} \tag{10}$$

$$UIQI(A.B) = L(A.B) * C(A.B) * S(A.B) \tag{11}$$

Ici:

L(A.B) : Distorsion de luminance.

C(A.B) : Distorsion de contraste.

S(A.B) : Perte de corrélation.

Où, *A* est l'image de couverture, μ_A et σ_A sont la moyenne et l'écart-type, respectivement de *A*. *B* est l'image de stego, μ_B et $\sigma_B$est la moyenne et l'écart-type, respectivement de B. σ_{AB} Est la covariance entre A et B [6].

I.7.9 Analyse qualitative

L'image de couverture peut-être subir de changement des valeurs de pixel au cours de l'opération d'incorporation, à la suite de laquelle la différence peut observer dans les deux images. Donc l'objectif d'analyse qualitative est d'observer tout changement dans la qualité visuelle [6].

I.8 Conclusion

Comme nous l'avons vu dans ce chapitre, la dissimulation réversible de données a fait l'objet de nombreuses recherches, principalement en raison de ses applications potentielles dans un monde numérique diversifié. Dans l'étude réalisée dans ce chapitre, nous avons présenté la DH, ses domaines d'application et un état de l'art sur la méthode RDH. Nous avons montré que les méthodes de dissimulation de données réversibles sont précisément classées en deux catégories notamment celles du domaine clair et celles caractérisant le domaine crypté. En s'intéressant au domaine clair, nous avons présenté plusieurs méthodes accompagnées de leurs avantages et inconvénient. Pour terminer ce chapitre, nous avons exposé quelques critères de mesure de la qualité et la quantité d'information des images après extraction des informations intégrées

Chapitre II

Principes des méthodes DE et ses extensions et PVO

II.1 Introduction

La dissimulation de données réversible RDH, également connue sous le nom d'intégration de données sans perte, est une technique qui permet d'intégrer des données à l'intérieur d'une image et plus tard, les données cachées peuvent être récupérées selon les besoins et permettre par conséquent de restaurer la copie exacte de l'image originale.

Comme nous l'avons vu déjà dans le premier chapitre, il existe plusieurs techniques de la dissimilation de donnée réversible. Parmi ces dernières, la RDH basée sur la méthode PVO (Pixel Value Ordering) et nommée PVO-RDH, donne une capacité d'intégration (CE) et un PSNR qui sont beaucoup élevés par rapport à d'autres méthodes.

Dans ce chapitre, on va étudier l'algorithme d'expansion de la différence et ses extensions notamment les techniques DE et PVO et on terminera par la méthode PVO-RDH.

II.2 Méthode DE

Les techniques basées sur l'expansion des différences (DE) ont beaucoup évolué au cours de la dernière décennie. Elles visent à intégrer de manière réversible des données dans une image hôte tout en réduisant la quantité d'informations auxiliaires à intégrer. Dans ce qui suit, nous passons en revue l'algorithme DE.

II.2.1 Principe de la méthode DE

La technique de l'expansion de la différence 'DE' est basée sur la sélection d'une paire deux pixels (P_1, P_2) avec :

$$0 \leq (P_1, P_2) \leq 255 \tag{12}$$

La moyenne L entière entre ces deux pixels est donnée par :

$$L = \frac{P_1 + P_2}{2} \tag{13}$$

Le principe de la DE est d'ajouter le bit d'information b à la valeur de la différence$h$. Son avantage est l'exploitation de la redondance dans le support de couverture. Cette méthode est constituée de deux étapes la 1ère étant l'intégration des données et la 2ème est l'extraction des données.

II.2.2 Exemple d'application de la DE

Dans cet exemple, on explique les étapes de la méthode DE.

II.2.2.1 Etape 1 (l'intégration)

- Sélection de deux pixels dans une image au niveau de gris.

  Exemple : $P_1 = 206$ et $P_2 = 201$ **;**

- Calcul de la moyenne entière :

$$L = \left\lfloor \frac{P_1 + P_2}{2} \right\rfloor = \left\lfloor \frac{206 + 201}{2} \right\rfloor = 203 \tag{14}$$

Où : $\lfloor . \rfloor$ est l'opérateur de la fonction plancher.

- Calcul de la différence entre les deux pixels : ($P_1 = 206$ et$P_2 = 201$**)**

$$h = P_1 - P_2 = 206 - 201 = (5)_{10} \tag{15}$$

- Transformer la différence h en binaire puis insérer le bit d'information (b) avec la suite résultante pour obtenir la nouvelle différence h'. Si on considère b=1, on arrive à :

$$h = (5)_{10} = (101)_2, Sans\ insertion\ du\ bit\ «\ b\ » \tag{16}$$

$$h' = (5)_{10} = (101b)_2, Avec\ insertion\ du\ bit\ «\ b\ » \tag{17}$$

- Convertir la nouvelle différance h' de la base 2 à la base 10 :

$$h' = (5)_{10} = (101b)_2, Avec\ insertion\ du\ bit\ «\ b\ » \tag{18}$$

$$h' = (101b)_2 = (1011)_2 = h' = (11)_{10} \tag{19}$$

Mathématiquement, cette opération est donnée comme suit :

$$h' = 2.h + b = 2 * 5 + 1 = 11 \tag{20}$$

- Enfin les nouvelles valeurs (P'_1 et P'_2**)** des pixels deviennent :

$$P'_1 = L + \left\lceil \frac{h' + 1}{2} \right\rceil = 209 \tag{21}$$

$$P'_2 = L - \left\lceil \frac{h'}{2} \right\rceil = 198 \tag{22}$$

La figure (II.1), présente un schéma de principe de l'opération d'intégration d'un bit d'information pour la méthode DE.

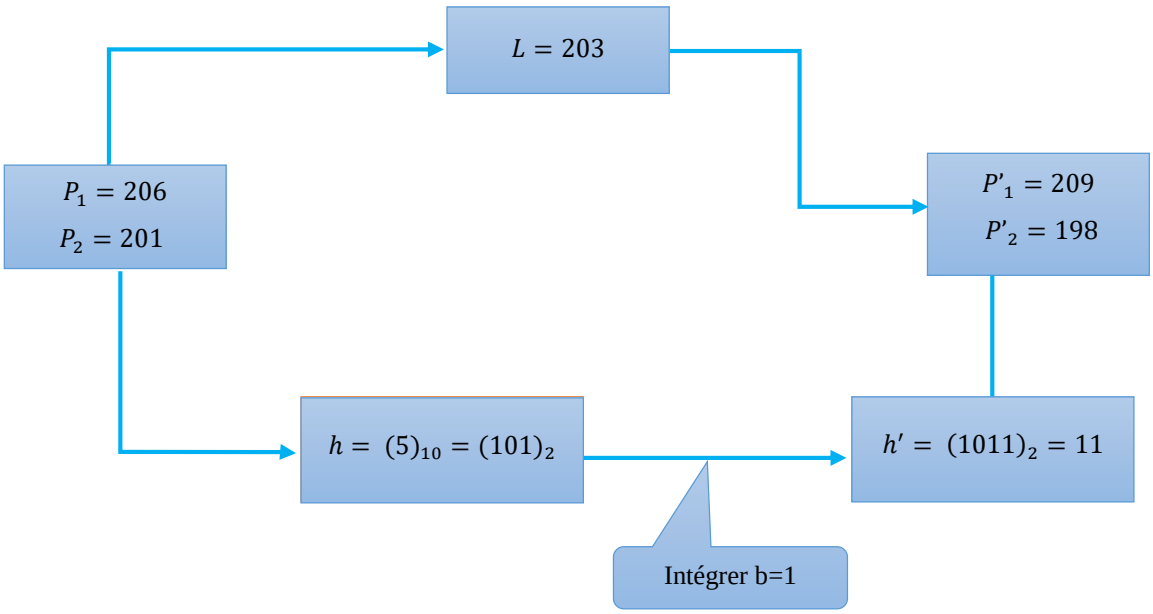


Figure. II. 1 Schéma de principe de l'opération d'intégration pour la méthode DE.

II.2.2.2 Etape 2 (extraction)

- La sélection de deux pixels dans une image au niveau de gris ($P'_1 = 209$ et$P'_2 = 198$)
- Calcul de la moyenne entière :

$$L' = \left\lfloor \frac{P'_1 + P'_1}{2} \right\rfloor = \left\lfloor \frac{209 + 198}{2} \right\rfloor = 203 \quad (23)$$

- Calcul de la différence entre ($P'_1 = 209$ et$P'_2 = 198$**)**

$$d = P'_1 - P'_2 = 209 - 198 = (11)_{10} \quad (24)$$

- Transformer la différence d en binaire puis extraire le bit d'information b.

$$d = (11)_{10} = (1011)_2, Sans\ extraction \quad (25)$$

$$d' = (101)_2 = (5)_{10}, Avec\ extraction\ ou\ b = 1 \quad (26)$$

- Convertir la nouvelle différance h' de la base 2 à la base10.

$$d' = (101)_2 = (5)_{10} \quad (27)$$

Mathématiquement, cette opération est donnée comme suit:

$$d' = \left\lfloor \frac{h'}{2} \right\rfloor = \left\lfloor \frac{11}{2} \right\rfloor = 5 \quad (28)$$

- Enfin les nouvelles valeurs (P_1 etP_2**)** des pixels deviennent :

$$P_1 = L' + \left\lfloor \frac{d' + 1}{2} \right\rfloor = 206 \quad (29)$$

$$P_2 = L' + \left\lfloor \frac{d'}{2} \right\rfloor = 201 \quad (30)$$

La figure (II.2) présente un schéma de principe de l'opération d'extraction d'un bit d'information pour la méthode DE.

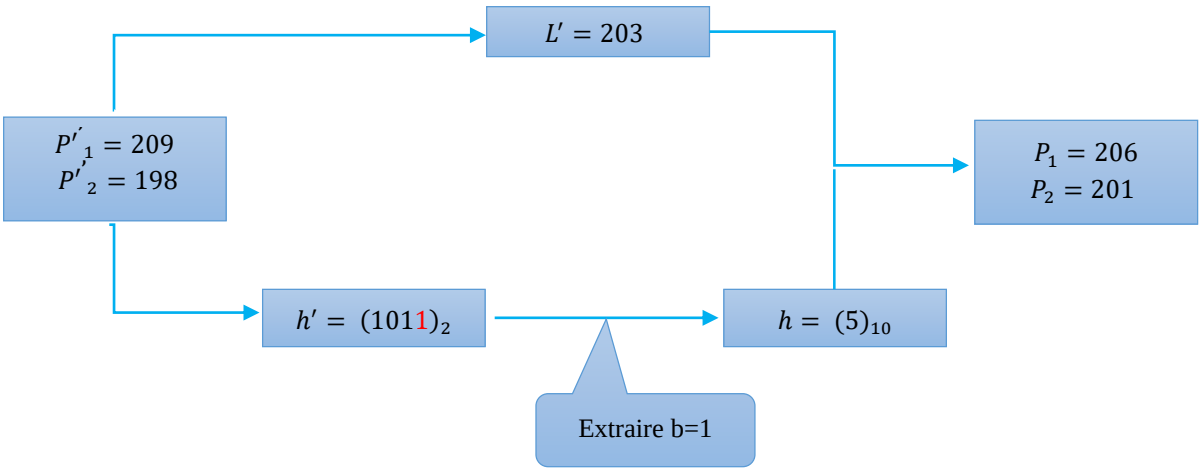


Figure. II. 2 Schéma de principe de l'opération d'extraction pour la méthode DE.

II.2.3 Les extensions de la méthode DE

La méthode DE a présenté plusieurs problèmes plus particulièrement sa capacité d'intégration qui est faible avec une distorsion énorme. Subséquemment, l'expansion de la différence a été développée en trois extensions majeures pour augmenter la capacité d'intégration et le niveau du PSNR dans le but d'avoir une bonne qualité d'image. La figure (II.3) montre les extensions de la méthode DE.

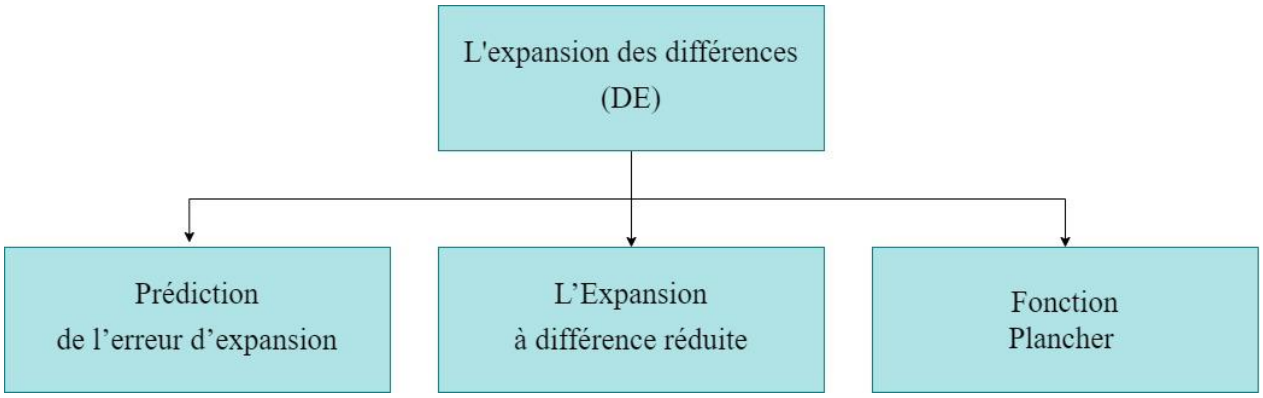


Figure. II. 3 Les extensions de la méthode DE.

II.2.3.1 Extension « Fonction Plancher »

Cette méthode consiste à utiliser un vecteur des pixels au lieu d'utiliser une paire de pixels comme c'était le cas dans la partie précédente. En effet, le principe de cette méthode est de diviser l'image en blocs de taille $a \times b$ non chevauchés puis les transformer en vecteur u tel que :

$$\begin{cases} 1 < a < h \\ 1 < b < w \end{cases} et\ a + b \neq 2 \qquad (31)$$

La figure (II.4) montre une image divisée en blocs.

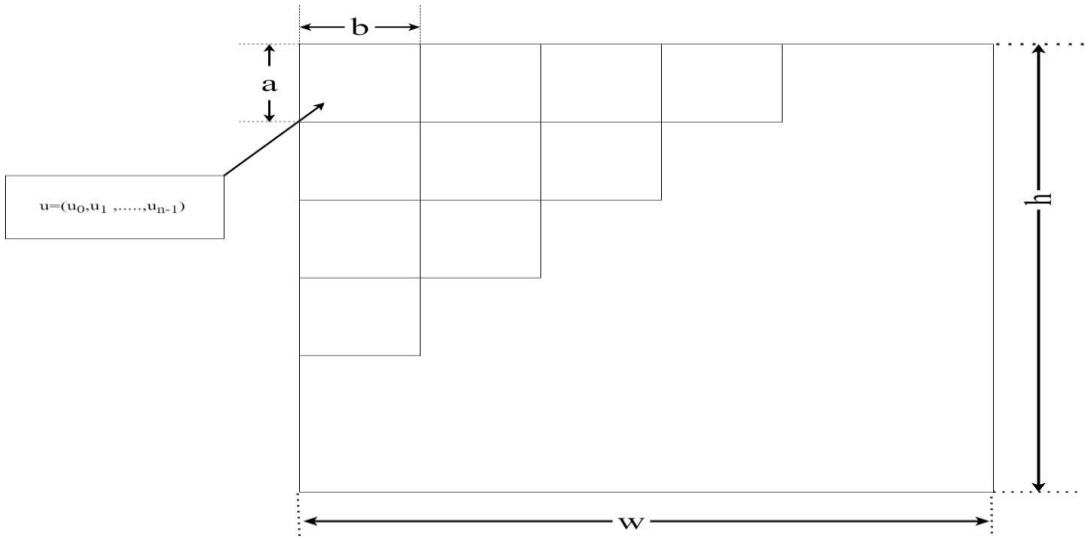


Figure. II. 4 La division d'une image en blocs.

Avec :

h : Longueur du bloc ;

w : Largeur du bloc ;

u : Vecteur des pixels.

Dans ce qui suit, on donne le principe de cette première extension.

II.2.3.1.1 Etape1 (l'intégration)

- Diviser l'image en blocs non chevauchés contenant chacun $N = a \times b$ pixels selon la condition donnée plus haut ;
- Transformer les valeurs de chaque bloc en vecteur $u = (u_0, u_1, u_2, \dots u_{n-1})$;
- Calculer la moyenne des pixels v_0 pour chaque vecteur u et la différence entre les pixels selon les relations suivantes :

$$\begin{cases} v_0 = \left\lfloor \frac{\sum_{i=0}^{N-1} a_i u_i}{\sum_{i=0}^{N-1} a_i} \right\rfloor \\ v_1 = u_1 - u_0 \\ v_2 = u_2 - u_0 \\ v_{N-1} = u_{N-1} - u_0 \end{cases} \quad (32)$$

- Une fois la différencev_{N-1}, pour tous les pixels, calculée, le processus d'intégration des données $(b_1, b_2, \ldots\ldots, b_{N-1})$ commence sachant que$b_0 \in \{0,1\}$. Il existe deux méthodes de l'insertion de données. Ces dernières sont montrées dans les équations (33) et (34).
  - ✓ La 1ère, concernant la méthode de la différence, est donnée par :

$$\begin{cases} v'_1 = 2 \times v_1 + b_1 \\ v'_{N-1} = 2 \times v_{N-1} + b_{N-1} \end{cases} \quad (33)$$

  - ✓ La 2ème, concernant la méthode du changement des LSBs, est donnée par :

$$\begin{cases} v'_1 = 2 \times \left\lfloor \frac{v_1}{2} \right\rfloor + b_1 \\ v'_{N-1} = 2 \times \left\lfloor \frac{v_{N-1}}{2} \right\rfloor + b_{N-1} \end{cases} \quad (34)$$

- Après l'intégration des bits, la nouvelle valeur des pixels devient :

$$\begin{cases} u'_0 = v_0 - \left\lfloor \frac{\sum_{i=0}^{N-1} a_i v_i}{\sum_{i=0}^{N-1} a_i} \right\rfloor \\ u'_1 = v'_1 + u'_0 \\ u'_{N-1} = v'_{N-1} + u'_0 \end{cases} \quad (35)$$

La figure (II.5) montre organigrame de la methode Extension « Fonction Plancher ».

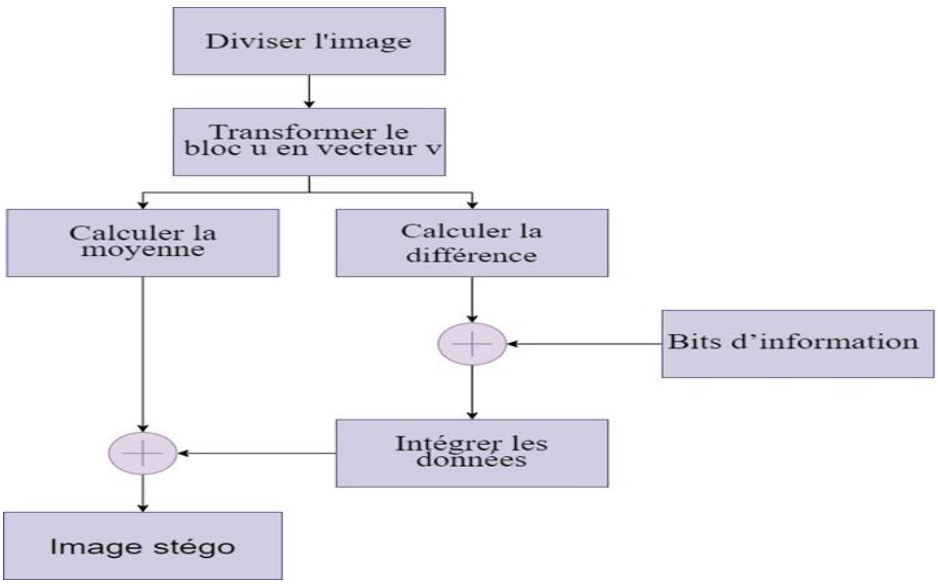


Figure. II. 5 Le principe de l'intégration des données par la méthode de la fonction Plancher.

II.2.3.1.2 Etape 2 (Extraction)

- Diviser l'image en blocs non chevauchés de taille $a \times b$ chacun ;
- Transformer chaque bloc de cette image en vecteurs u'selon les mêmes conditions données plus haut ;
- Prendre les LSBs à partir la 2ème valeur des pixels du vecteur u'_1 en utilisant l'équation suivante :

$$b = \begin{cases} b_1 = \lfloor u'_1 \rfloor mod[2] \\ b_2 = \lfloor u'_2 \rfloor mod[2] \\ b_3 = \lfloor u'_3 \rfloor mod[2] \end{cases} \quad (36)$$

- Afin d'extraire les valeurs de b, on peut récupérer les valeurs initiales de la différence $(v_1, v_2, \ldots . v_{n-1})$ comme le montre l'équation (37).

$$\begin{cases} v_1 = \dfrac{(u'_1 - b_1)}{2} \\ v_{n-1} = \dfrac{(u'_{n-1} - b_{n-1})}{2} \end{cases} \quad (37)$$

- Pour récupérer les valeurs originales du vecteur u, il faut passer par les équations de l'expression suivante :

$$\begin{cases} u_0 = u'_0 + \left\lfloor \dfrac{\sum_{i=0}^{N-1} a_i v_i}{\sum_{i=0}^{N-1} a_i} \right\rfloor \\ u_1 = v_1 + u_0 \\ u_{n-1} = v_{n-1} + u_0 \end{cases} \quad (38)$$

Comme nous allons le voir dans le chapitre III, cette méthode n'a pas donné de bons résultats soit dans la capacité d'intégration ou bien concernant le PSNR. Pour remédier à ce problème, les auteurs de la référence [19] ont proposé une amélioration de cette technique nommée par méthode d'Expansion à différence réduite. Dans ce qui suit, nous allons donner le principe et les étapes de cette extension.

II.2.3.2 Méthode d'Expansion à différence réduite

Cette méthode consiste à réduire la différence v avant l'intégration de données. Donc, on obtient une nouvelle différence réduite $\overline{v}$ plus petite par rapport à v. Elle est donnée par :

$$\overline{v} = \begin{cases} v \quad si\ v < 2 \\ v - 2^{\lfloor log_2 v \rfloor - 1}\ si\ v \geq 2 \end{cases} \quad (39)$$

Pour arriver à rétablir la valeur de la différence originale, il faut créer une carte de localisation qui va permettre de marquer les différences pour les récupérer selon les conditions suivantes :

$$\begin{cases} v = \overline{v} - 2^{\lfloor log_2 |\overline{v}| \rfloor - 1} \; si \; LM = 0 \\ v = \overline{v} - 2^{\lfloor log_2 |\overline{v}| \rfloor} \; si \; LM = 1 \end{cases} \quad (40)$$

Dans ce qui suit, on donne les étapes de cette deuxième méthode.

II.2.3.2.1 Etape 1 (intégration)

- Diviser l'image en blocs de dimensions $a \times b$ et créer le vecteur $u = (u_0, u_1, u_2, \dots u_{n-1})$ correspondant à chaque bloc ;
- Calculer la différence v_n selon l'équation suivante :

$$\begin{cases} v_1 = u_1 - u_0 \\ v_2 = u_2 - u_0 \\ v_{N-1} = u_{N-1} - u_0 \end{cases} \quad (41)$$

- Réduire la différence $\overline{v}$ comme suit :

$$\begin{cases} v' = v \; si \; v < 2 \\ v' = v - 2^{\lfloor log_2 (|v|) \rfloor - 1} si \; v \geq 2 \end{cases} \quad (42)$$

- Construire la carte de localisation et marquer la différence originale selon les conditions suivantes :

$$\begin{cases} v = \overline{v} + 2^{\lfloor log_2 |\overline{v}| \rfloor - 1} \; par \; LM = 0 \\ v = \overline{v} + 2^{\lfloor log_2 |\overline{v}| \rfloor} \; par \; \; LM = 1 \end{cases} \quad (43)$$

- Intégrer les données $(b_1, b_2, \dots\dots, b_{N-1})$ sachant que $b_i \in \{0,1\}$;
- Après avoir intégré la charge utile b_i dans la nouvelle valeur de différence, on trouve $\overline{v_i}$ (transformé en $\overline{v_i}'$) ;
- Enfin, les valeurs de $u = (u'_0, u'_1, u'_2, \dots u'_{n-1})$ sont donnés par :

$$\begin{cases} u'_0 = \; u_0 \\ u'_1 = \overline{v_i}' + u'_0 \\ u'_{N-1} = \overline{v_i}' + u'_0 \end{cases} \quad (44)$$

II.2.3.2.2 Etape 2 (Extraction)

La procédure d'extraction est un processus inverse de l'intégration. Elle peut être détaillée comme suit :

- En premier lieu, il faut extraire la carte de la localisation pour restaurer les valeurs originales de l'image ;

- Diviser l'image marquée en blocs, chacun d'eux ayant N pixels. Après cela, la différence entre les paires de pixels est calculée à l'aide de l'équation suivante :

$$\begin{cases} \overline{v'_1} = u'_1 - u'_0 \\ \overline{v'_2} = u'_2 - u'_0 \\ \overline{v'_{N-1}} = u'_{N-1} - u'_0 \end{cases} \quad (45)$$

- Récupérer la suite binaire du contenu cachet $(b_1, b_2, \ldots\ldots, b_{N-1})$ selon l'équation (46) et les valeurs de la différence réduite v'_{N-1} par l'équation (47)

$$\begin{cases} b_1 = \left[\overline{v'_1}\right] mod[2] \\ b_2 = \left[\overline{v'_2}\right] mod[2] \\ b_{N-1} = \left[\overline{v'_{N-1}}\right] mod[2] \end{cases} \quad (46)$$

$$\begin{cases} v'_1 = \dfrac{(v'_1 - b_1)}{2} \\ v'_2 = \dfrac{(v'_2 - b_2)}{2} \\ v'_{N-1=} \dfrac{(v'_{N-1} - b_{N-1})}{2} \end{cases} \quad (47)$$

- Après avoir récupéré les bits d'information, en utilisant la carte de la localisation, on peut restaurer l'image originale comme suit :

$$LM = \begin{cases} 0 \rightarrow v = \overline{v} - 2^{\lfloor log_2 |\overline{v}| - 1} \\ 1 \rightarrow v = \overline{v} - 2^{\lfloor log_2 |\overline{v}| \rfloor} \end{cases} \quad (48)$$

$$\begin{cases} u_0 = u'_0 \\ u_1 = v_1 + u_0 \\ v_{N-1} = v_{N-1} + u_0 \end{cases} \quad (49)$$

II.2.3.3Méthode de Prédiction de l'erreur d'expansion

L'approche d'expansion des erreurs de prédiction a été introduite pour la première fois par Thodi et all [20]. Ici, l'erreur de prédiction est déterminée pour chaque pixel du support de couverture en utilisant le pixel prédit. Ce dernier est calculé à partir des pixels voisins afin de créer un espace libre pour cacher des informations secrètes.

Considérons un pixel d'intensité x dans une image en niveaux de gris pour calculer $\hat{x}_i$ en utilisant l'algorithme [21] tel que :

$$\hat{x} = \begin{cases} \max(a,b) \ \ si\ c \leq \min(a,b) \\ \min(a,b)\ si\ \ c \geq \max(a,b) \\ a + b - c \ \ ailleurs \end{cases} \quad (50)$$

Où x est le pixel courant, x est la valeur prédite et a, b, c est le contexte. Le contexte comme indiqué dans la figure (II.6).

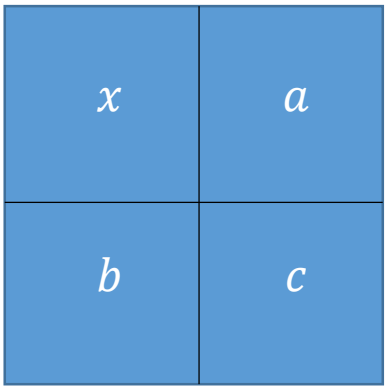


Figure. II. 6 Prédiction pour le cas d'un un bloc d'une image de Taille $2 \times 2 = 4$.

Le pixel de la dernière colonne de la dernière ligne n'est pas pris en compte pour l'incorporation de données.

La prédiction d'erreur est définie par :

$$p_{ee} = x_i - \widehat{x_i} \tag{51}$$

Pour incorporer un bit i dans x, nous représentons son erreur de prédiction en utilisant une représentation binaire, nous le décalons vers la gauche d'un bit et nous insérons le bit i dans le LSB vacant Si l est la longueur de la représentation binaire f p_{ee} alors

$$\begin{cases} p_{ee} = b_{l-1}, b_{l-2}..b_0 \\ p'_{ee} = b_{l-1}, b_{l-2}..b_0 i = 2 \times p_{ee} + i \end{cases} \tag{52}$$

Où p'_{ee} est l'erreur de prédiction modifiée. La valeur modifiée est t.

$$x' = p'_{ee} + \widehat{x_i} = x + p_{ee} + i \tag{53}$$

Au niveau du décodeur, l'extraction se fait en calculant d'abord l'erreur de prédiction , p'_{ee}. Le bit incorporé est extrait du LSB. La valeur du pixel d'origine est restaurée après avoir calculé l'erreur de prédiction originale comme suit :

$$\begin{cases} p_{ee} = \left\lfloor \frac{p'_{ee}}{2} \right\rfloor \\ x = x' - p_{ee} - i \end{cases} \tag{54}$$

Pour les autres cas. C'est-à-dire pour $a \times b = N$ tel que :$n > 4$, $\widehat{x_i}$ est donné par :

$$\widehat{x_i} = \frac{x(i-1,j-1) + x(i-1,j+1) + x(i+1,j-1) + x(i+1,j+1)}{4} \tag{55}$$

Le principe de calcul de $\widehat{x_i}$ est montré sur la figure (II.7).

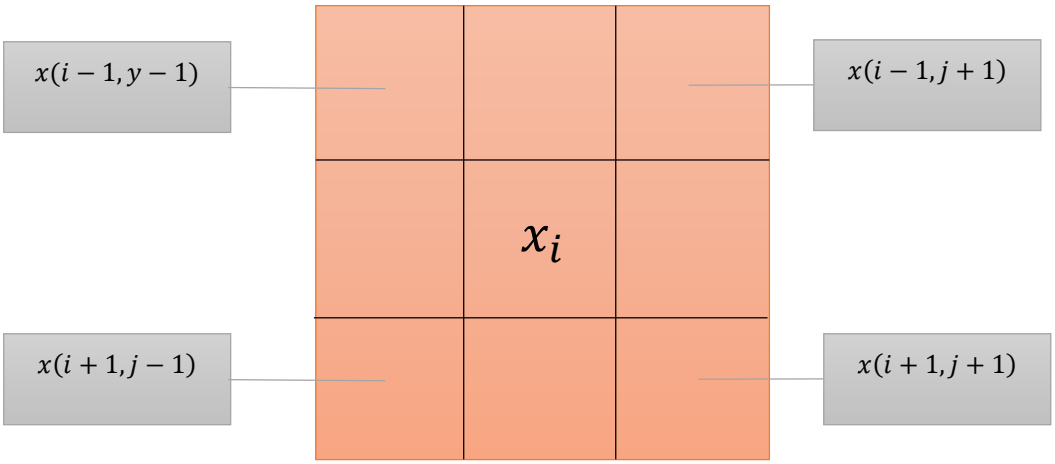


Figure. *II. 7* Prédiction pour le cas d'un bloc d'une image de taillea × b > 4.

Pour la procédure d'extraction des bits et la récupération de la valeur originale du support de couverture, on procède de la façon suivante :

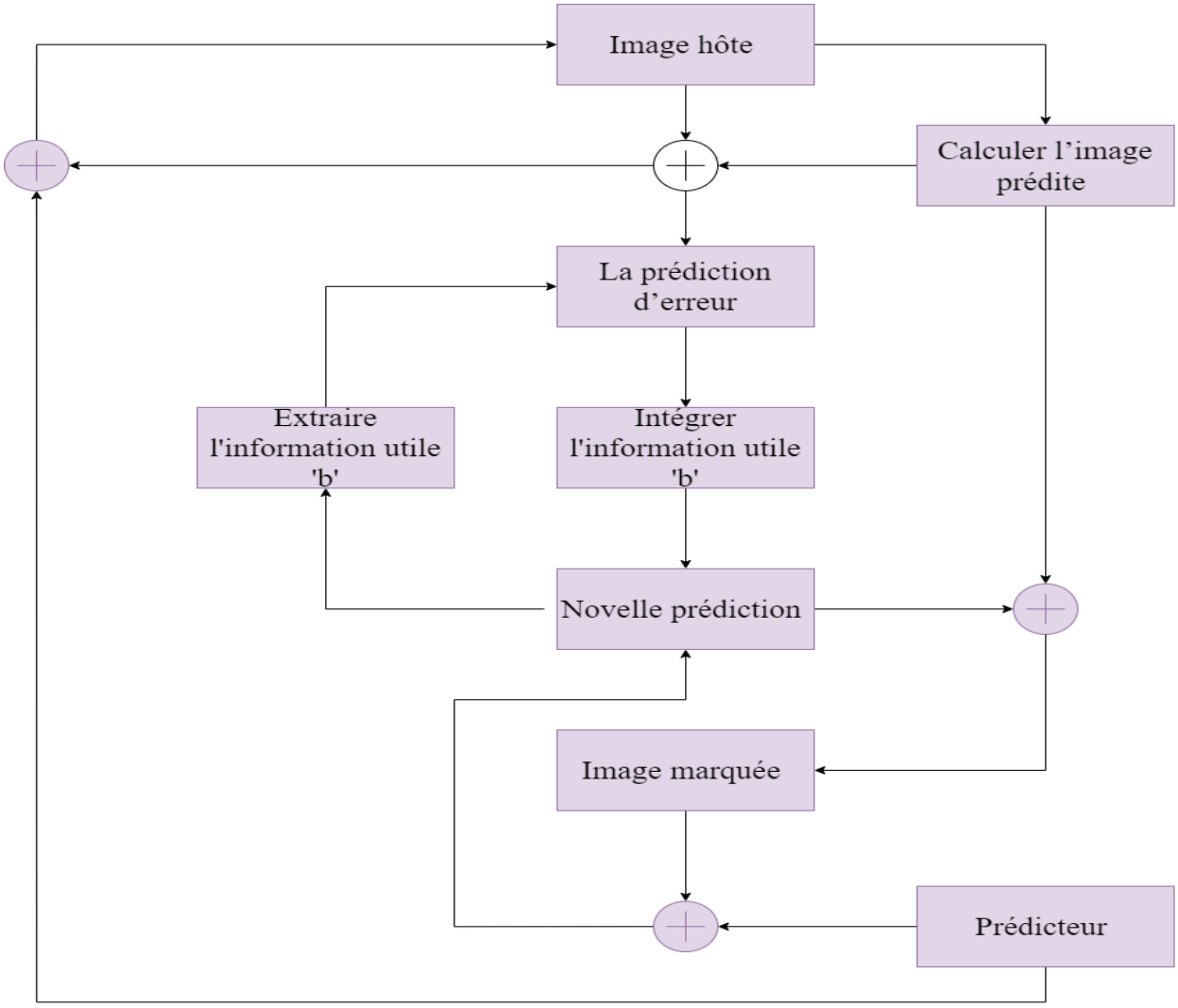
- Calculer la prédiction d'erreur p'_{ee} et b en utilisant l'équation (56)

$$\begin{cases} p'_{ee} = x_i - \widehat{x_i} \\ b = \lfloor p'_{ee} \rfloor mod[2] \end{cases} \quad (56)$$

Après le processus précédant, on passe à la récupération du support de couverturex_i. Avant cela, il faut restaurer la valeur originale de la prédiction d'erreur p_{ee} comme suit :

$$\begin{cases} p_{ee} = \dfrac{p'_{ee} - b}{2} \\ x_i = \widehat{x_i} + p_{ee} \end{cases} \quad (57)$$

La figure (II.8), montre l'organigramme de cette méthode.



Fi

gure. *II. 8* Organigramme de la méthode PEE.

II.3. Prédicteur PVO

L'algorithme PVO a été proposé par Li et al dans la référence [4]. Comme le montre la figure (11), son principe de base est de diviser l'image en **k** blocs de mêmes tailles (ou k est le nombre total des blocs) non chevauchés pour permettre de regrouper, dans un seul vecteur $X_k = (X_0, \dots, X_N)$ de taille N ($N = n \times n$, n signifie le nombre de lignes et le nombre de colonnes du bloc), les valeurs de chaque bloc. Ces valeurs seront ensuite triées selon un ordre croissant. Pour chaque bloc, les auteurs ont utilisé la deuxième plus grande valeur pour prédire son maximum et la deuxième plus petite valeur pour prédire son minimum. Ensuite, ils ont appliqué la PEE aux histogrammes d'erreur de prédiction afin de permettre l'intégration des données de

manière en blocs. Dans l'exemple suivant, on va montrer le principe du prédicteur PVO pour le cas de $N = 4$ ($N = 2 \times 2$).

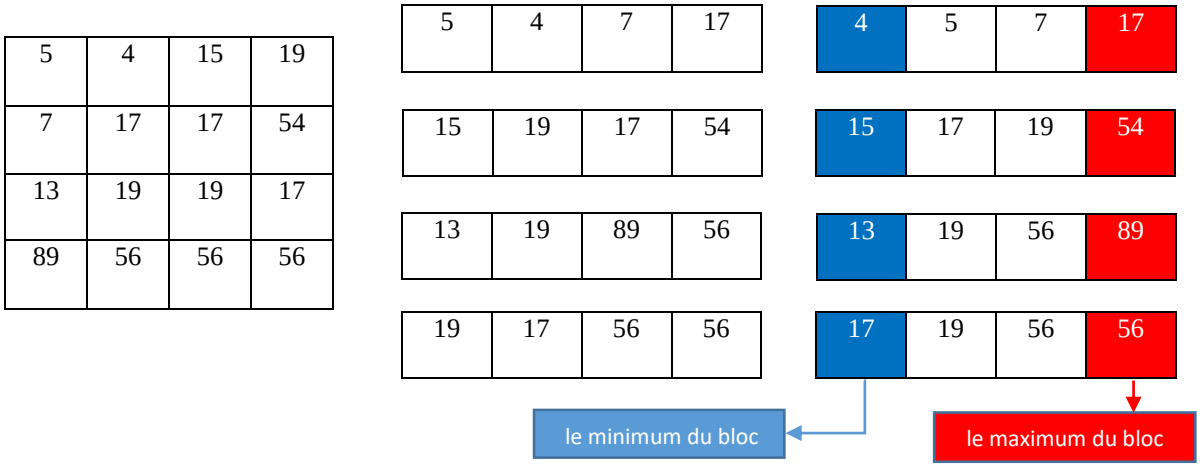


Figure. II. 9 Principe de fonctionnement de la méthode PVO.

II.4 Méthode PVO-RDH

Dans ce qui suit, nous allons donner le principe détaillé de la méthode d'intégration et d'extraction des données basée sur le prédicteur PVO-RDH. Le principe est illustré sur la figure (II.11).

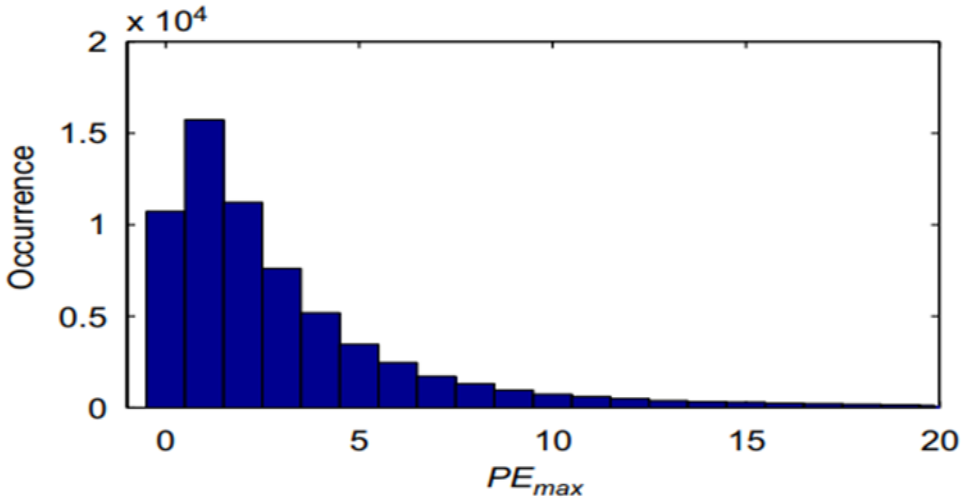


Figure.II.10 Histogramme de l'erreur de prédiction.

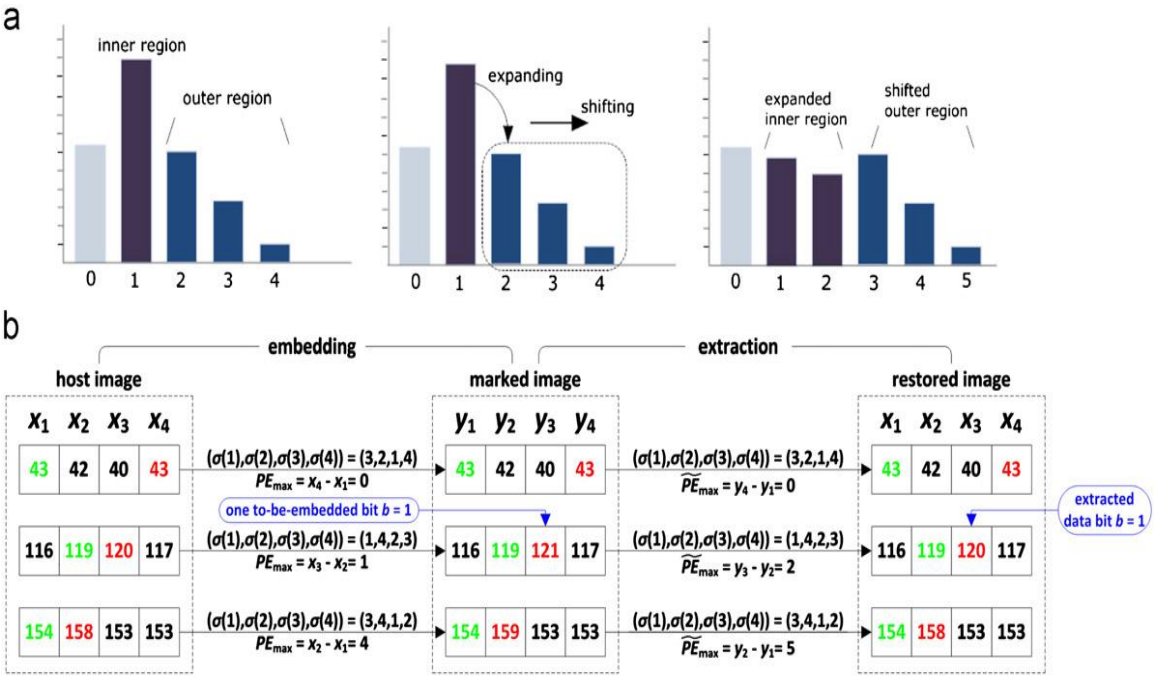


Figure. II. 11 Principe de fonctionnement de la méthode PVO-RDH [4].

II.4.1 Intégration des données

Comme nous l'avons vu dans la section précédente, l'image hôte tout d'abord divisée en blocs, non chevauchés, de taille égale n chacun. Pour chacun bloc X donné, contenant N pixels, la technique PVO permet de trier les valeurs de chaque bloc $(x_1, \dots, x_n)$ selon un ordre croissant pour obtenir les valeurs $\left(x_{\sigma(1)}, \dots, x_{\sigma(n)}\right)$. On utilise la 2ème plus grande valeur $x_{\sigma(n-1)}$ pour prédire le maximum$x_{\sigma(n)}$. L'erreur de prédiction PE_{max} est donnée par :

$$PE_{max} = x_{\sigma(N)} - x_{\sigma(n-1)} \tag{58}$$

Pour l'exemple $n = 4$ ($n = 2 \times 2$) précédent, la figure suivante montre l'histogramme de PE_{max} pour l'image Lena.

Comme le montre cette figure, dans l'histogramme de la prédiction PEE, il y a toujours une région qui peut être utilisée pour l'intégration des données. Cette région est appelée « région utile ». Elle correspond à la case pour laquelle PE_{max} est égale à 1. Pour les autres cases, la région est considérée comme étant une région inutile. Dans ces conditions, pour intégrer les données en utilisant la méthode PEE, PE_{max} doit être modifiée comme suit :

$$\widetilde{PE}_{max} = \begin{cases} PE_{max}\ si\ PE_{max} = 0 \\ PE_{max} + b\ si\ PE_{max} = 1 \\ PE_{max} + 1 si\ PE_{max} > 1 \end{cases} \tag{59}$$

Où : $b \in [0,1]$ est un bit de données à intégrer. Par conséquent, $x_{\sigma(n)}$ peut être modifiée comme suit :

$$\tilde{x} = x_{\sigma(n-1)} + \widetilde{PE}_{max} = \begin{cases} x_{\sigma(n)}; & si\ PE_{max} = 0 \\ x_{\sigma(n)} + b; & si\ PE_{max} = 1 \\ x_{\sigma(n)} + 1; & si\ PE_{max} > 1 \end{cases} \quad (60)$$

Les autres valeurs $x_{\sigma(1)}, \dots, x_{\sigma(n-1)}$ restent inchangées.

La valeur marquée de X est $(y_1, y_2, \dots, y_n)$, où : $y_\sigma = \tilde{x}$ et $y_i = x_i$, pour chaque$i \neq \sigma(n)$.

Dans la procédure ci-dessus, étant donné que le maximum $x_{\sigma(n)}$ est soit inchangé ou augmenté, l'ordre des valeurs des pixels (i, e. .Le mappage) reste inchangé. Par conséquent, pour un bloc marqué dont la valeur est $(y_1, \dots, y_n)$, on calcule l'erreur de prédiction comme suit :

$$\widetilde{PE}_{max} = y_{\sigma(n)} - y_{\sigma(n-1)} \quad (61)$$

II.4.2 Extraction des données

L'extraction des données et la restauration de l'image peuvent être réalisées comme suit :

- Si $\widetilde{PE}_{max} = 0$, le bloc est inchangé dans l'intégration des données et sa valeur originale est simplement $(y_1, \dots, y_n)$ elle-même ;
- Si $\widetilde{PE}_{max} \epsilon \{1,2\}$, le bloc est étendu pour transporter des données cachées dans l'intégration des données. Le bit de données intégré est $b = \widetilde{PE}_{max} - 1$ et la valeur originale est $(x_{\sigma(1)}, \dots, x_{\sigma(n)})$ ou $x_{\sigma(n)} = y_{\sigma(n)} - b$ et $y_i = x_i$ pour chaque $i \neq \sigma(n)$.
- Si $\widetilde{PE}_{max} > 2$ le bloc est décalé lors de l'intégration des données et sa valeur originale est : $(x_{\sigma(1)}, \dots, x_{\sigma(n)})$, ou $x_{\sigma(n)} = y_{\sigma(n)} - 1$ et $y_i = x_i$ pour chaque $i \neq \sigma(n)$.

Similairement, au cas de la 2ème plus grande valeur$x_{\sigma(n-1)}$, on peut considérer la 2ème plus petite valeur $X_{\sigma(2)}$ pour prédire le minimum$X_{\sigma(1)}$. L'erreur de prédiction PE_{min} donnée par :

$$PE_{min} = x_{\sigma(2)} - x_{\sigma(1)} \quad (62)$$

Avec la même procédure utilisée précédemment et en considérant à la fois le maximum et le minimum, au plus, deux bits peuvent être intégrés dans un bloc en même temps, ce qui peut augmenter efficacement la CE.

II.5 Conclusion

Dans ce chapitre, on a étudié le principe de fonctionnement de la méthode RDH basée sur la méthode de différence d'expansion avec ses domaines d'utilisation. Puis, une étude détaillée sur les extensions majeures de la méthode DE ont été présentées. Après, la nouvelle stratégie de la dissimilation de donnée basée sur la méthode PVO a été détaillée. Dans le chapitre suivant on va implanter sous environnement Python cette dernière méthode.

Chapitre III

Implémentation sur Python, Résultats et discussions

III.1 Introduction

Nous avons présenté dans le chapitre précédent les méthodes de la dissimulation des données réversibles et plus précisément la méthode PVO-RDH. Dans ce dernier chapitre on va présenter et implémenter l'algorithme PVO-RDH sous environnement Python. A la fin, nous présentons les résultats de la simulation et les performances du système d'intégration et d'extraction des données ainsi que la sécurité et l'efficacité de cet algorithme.

III.2 Environnement de développement

Dans cette partie, nous allons présenter l'environnement logiciel (Software) et matériel (Hardware) utilisés pour la réalisation du côté pratique de ce projet.

III.2.1 Environnement matériel

Ce travail a été réalisé sur un PC (Laptop HP modèle : Probook) ayant les caractéristiques suivantes :

- Processeur : Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz, 2.40 GHz ;
- Mémoire installée (RAM) : 4,00 Go(DDR4) ;
- Disque Dure : SSD 256 Go ;
- Carte graphique : Intel r HD graphics5500 ;
- Système d'exploitation : Windows 10 Professionnel 64 bit.

III.2.2 Environnement logiciel

Nous avons utilisé l'environnement Python pour l'implémentation de l'algorithme proposé.

III.2.2.1 Description générale sur le Python

Python est un langage de programmation puissant et facile à apprendre, idéal pour l'écriture des scripts et le développement rapide d'applications dans de nombreux domaines et sur la plupart des plates-formes [21]. Il possède des structures de données de haut niveau et permet une approche simple et efficace de la programmation orientée objet [21]. C'est un langage très riche et populaire classé parmi les premiers langages de programmation.

III.2.2.2 Historique de python

Python, qui tire son nom de l'émission Monty Python's Flying Circus, a été créé
par le programmeur Guido van Rossum en 1991. La première version publique de ce Langage a

été publiée en 1991 et sa dernière version est la version 3. Plus précisément, la version 3.7 est sortie en juin 2018 [22].

III.2.2.3 Version de python

Ce langage continu toujours à évoluer, mais cette évolution ne vise qu'à améliorer ou perfectionner le produit. Il est donc très rare qu'il faille modifier les programmes afin de les adapter à une nouvelle version qui serait devenue incompatible avec les précédentes [23]

Chaque passage à une version supérieure s'est fait avec une compatibilité ascendante. Cela signifie par exemple que le code Python qui s'exécute avec Python 3.2, marchera également avec Python 3.3 ou versions suivantes. Il y a cependant une exception à cela : le passage de Python 2.x (2000) à Python 3.x (dès 2008) ! Pour corriger certains défauts de jeunesse du langage, il a en effet été décidé de faire quelques changements importants qui rendent le code Python 2 non exécutable avec Python 3 sans certaines adaptations. Ce changement a été fait sur plusieurs années. Cela a donné à chacun ayant écrit du code le temps de migrer ses projets vers la nouvelle version [23]. Aujourd'hui, Python 2 n'existe que d'un point de vue historique, et nous allons donc faire du Python 3.

Le langage de programmation présente de nombreuses caractéristiques intéressantes :

- Il est multiplateforme. C'est-à-dire qu'il fonctionne sur de nombreux systèmes d'exploitation : Windows, Mac OS X, Linux, Android, iOS, depuis les mini-ordinateurs Raspberry Pi jusqu'aux supercalculateurs [22];
- Il est gratuit. Vous pouvez l'installer sur autant d'ordinateurs que vous voulez (même sur votre téléphone !) [22];
- C'est un langage de haut niveau. Il demande relativement peu de connaissances sur le fonctionnement d'un ordinateur pour être utilisé [22];
- C'est un langage interprété. Un script Python n'a pas besoin d'être compilé pour être exécuté, contrairement à d'autres langages comme le C ou le C++ [22];
- Il est orienté objet. C'est-à-dire qu'il est possible de concevoir en Python des entités qui miment celles du monde réel (une cellule, une protéine, un atome, etc.) avec un certain nombre de règles de fonctionnement et d'interactions [22];

III.2.2.4 Domaines applications du Python

Les 3 domaines principaux où Python est utilisé sont :

- Développement web : Python est très utilisé dans le développement des applications web, à savoir :
    - **Google** : qui est l'une des plus grandes entreprises de l'informatique au monde. Leur première version de moteur de recherche et la totalité de leurs technologies ont été développées en Python [25];
    - **YouTube :** il a été créé à la base en PHP et a décidé de se convertir en Python après la hausse remarquable des nombres d'utilisateurs et la nécessité de développer de nouvelles fonctionnalités [25];
    - **Facebook :** le plus grand réseau social jamais développé au monde utilise aussi Python pour améliorer le fil d'actualité et servir à la détection et la reconnaissance faciale [25]!
    - **Netflix :** la petite star du WEB**,** qui utilise Python partout, soit pour établir ses algorithmes de recommandation ou gérer son réseau de distribution.
- Data Science – incluant le machine learning, l'analyse de données ainsi que la visualisation de données [25];
- Le Scripting pour automatiser certaines tâches sur des systèmes d'exploitation et sous la forme de langages de développement [25].

III.2.2.5 Bibliothèque de python

- **Tkinter**

Tkinter est un module de base intégré dans Python, normalement vous n'avez rien à faire pour pouvoir l'utiliser. L'un des avantages de Tkinter est sa portabilité sur les OS les plus utilisés par le grand public [26].

- **MSVCRT**

Ces fonctions donnent accès à certaines fonctionnalités utiles sur les plates-formes Windows. Certains modules de niveau supérieur utilisent ces fonctions pour construire les implémentations Windows de leurs services.

Le module implémente à la fois les variantes normale et large de l'API d'E/S de la console. L'API normale ne traite que les caractères ASCII et est d'une utilité limitée pour les applications internationalisées. L'API wide char doit être utilisée dans la mesure du possible.

Modifié dans la version 3.3 : Les opérations de ce module génèrent maintenant une erreur OSError au lieu d'une erreur IOError [27].

- **Math**

Math a été conçu pour être utilisé comme un schéma d'authentification par courrier électronique uniquement (mail auth -> mauth -> moth), mais il est suffisamment générique pour être utilisé pour pratiquement toute authentification basée sur des jetons [28].

- **Matplotlib**

C'est une bibliothèque de graphiques et de traçages Python qui peut générer une variété de différents types de graphiques ou de graphiques dans une variété de formats. Il peut être utilisé pour générer des graphiques linéaires, des graphiques de dispersion, des cartes thermiques, des graphiques à barres, des diagrammes circulaires et des graphiques 3D. Il peut même soutenir des animations et des affichages interactifs [29].

- **Numpy**

C'est une bibliothèque pour le langage de programmation Python qui prend en charge les grands tableaux et matrices multidimensionnels, ainsi qu'un certain nombre de fonctions mathématiques avancées [30].

- **Open cv**

Open cv (open computer vision) est une bibliothèque open source spécialisée dans le traitement et l'analyse d'image en temps réel. C'est une bibliothèque libre dédiée à la vision par ordinateur, elle est compatible avec divers systèmes d'exploitation [31].

III.3 Application Numériques des méthodes de base de la RDH-PVO

Dans cette section, on va présenter quelques applications numériques concernant les méthodes de dissimulation de données de base de l'algorithme PVO-RDH. Les résultats de ces applications ont été reportés des références [4].

III.3.1 Application Numérique de l'Extension « Fonction Plancher »

III.3.1.1 L'intégration des données:

Dans cet exemple, on va étudier un bloc d'image vectorisé en, $u = (8,10,14,15)$. On va donc suivre l'algorithme étape par étape.

Calcul des différences entre u_0 et les autres pixels voisins. Le résultat de cette opération est

montré dans l'équation (63) :

$$\begin{cases} v_0 = \left\lfloor \frac{8\times 1 + 10\times 2 + 14\times 2 + 15\times 1}{1+2+2+1} \right\rfloor = \left\lfloor \frac{71}{6} \right\rfloor = 11 \\ v_1 = 10 - 8 = 2 \\ v_2 = 14 - 8 = 6 \\ v_3 = u_3 - u_0 = 15 - 8 = 7 \end{cases} \quad (63)$$

- Considérant maintenant la suite binaire $b = (1,0,1)$ qu'on veut intégrer dans l'image hôte. L'opération l'intégration de cette suite d'information est montrée par l'équation (64) :

$$\begin{cases} v'_1 = 2\times v_1 + b_1 = 2\times 2 + 1 = 5 \\ v'_2 = 2\times v_2 + b_2 = 2\times 6 + 0 = 12 \\ v'_3 = 2\times v_3 + b_3 = 2\times 7 + 1 = 15 \end{cases} \quad (64)$$

- Les nouvelles valeurs des pixels u' deviennent donc :

$$\begin{cases} u'_0 = 11 - \left\lfloor \frac{5\times 2 + 12\times 2 + 15\times 1}{1+2+2+1} \right\rfloor = 3 \\ u'_1 = 5 + 3 = 8 \\ u'_2 = 12 + 3 = 15 \\ u'_3 = 15 + 3 = 18 \end{cases} \quad (65)$$

- Par conséquent, u' devient : $u' = (3,8,15,18)$. C'est ce bloc qui sera ensuite transmis à travers un canal de communication.

III.3.1.2 L'extraction des données

- Côté récepteur, après division de l'image marquée en blocs, on obtient le vecteur $u' = (3,8,15,18)$ résultant.
- Récupération de la valeur de la différence :

  Cette dernière est donnée par l'équation (4) suivante :

$$\begin{cases} v'_1 = 8 - 3 = 5 \\ v'_2 = 15 - 3 = 12 \\ v'_2 = 18 - 3 = 15 \end{cases} \quad (66)$$

- On récupère les bits d'information. Ceux-ci sont montrés dans l'équation (67) :

$$\begin{cases} b_1 = \lfloor v'_1 \rfloor \bmod [2] = \lfloor 5 \rfloor \bmod [2] = 1 \\ b_2 = \lfloor v'_2 \rfloor \bmod [2] = \lfloor 12 \rfloor \bmod [2] = 0 \\ b_3 = \lfloor v'_3 \rfloor \bmod [2] = \lfloor 15 \rfloor \bmod [2] = 1 \end{cases} \quad (67)$$

- On récupère maintenant les valeurs avant l'intégration de données. Celles-ci sont illustrées dans l'équation (68) :

$$\begin{cases} v_1 = \dfrac{(v'_1 - b_1)}{2} = \dfrac{5-1}{2} = 2 \\ v_2 = \dfrac{(v'_2 - b_2)}{2} = \dfrac{12-0}{2} = 6 \\ v_3 = \dfrac{(v'_3 - b_3)}{2} = \dfrac{15-1}{2} = 7 \end{cases} \quad (68)$$

- Restauration des pixels initiaux u qui sont égaux à $(8,10,14,15)$. Le résultat est montré par l'équation (69) :

$$\begin{cases} u_0 = 3 + \dfrac{2\times 2 + 6\times 2 + 7\times 1}{2+2+1} = 8 \\ u_1 = 2 + 8 = 10 \\ u_2 = 6 + 8 = 14 \\ u_3 = 7 + 8 = 15 \end{cases} \quad (69)$$

III.3.2 Application Numérique de l'Extension à « Différence Réduite »

III.3.2.1 L'intégration des données

Similairement au premier cas, nous prenons un vecteur de pixel$u = (108,110,134,115)$ et nous appliquons les étapes de l'algorithme.

- En premier lieu, on calcul la différence. Celle-ci est donnée par :

$$\begin{cases} v_1 = 110 - 108 = 2 \\ v_2 = 134 - 108 = 26 \\ v_3 = 115 - 108 = 7 \end{cases} \quad (70)$$

- On réduit ensuite cette différence. Le résultat est montré dans l'équation (71) :

$$\begin{cases} v'_1 = 1 \\ v'_2 = 18 \\ v'_3 = 5 \end{cases} \quad (71)$$

- A ce stade, on construit la carte de localisation comme l'illustre l'équation (72) :

$$\begin{aligned} v'_1 = 1 \rightarrow LM = 0 \\ v'_2 = 18 \rightarrow LM = 1 \\ v'_3 = 5 \rightarrow LM = 1 \end{aligned} \quad (72)$$

- On passe maintenant à l'intégration des données

$$\begin{cases} \overline{v'_1} = 2\times 1 + 1 = 3 \\ \overline{v'_2} = 2\times 18 + 1 = 37 \\ \overline{v'_3} = 2\times 7 + 1 = 15 \end{cases} \quad (73)$$

- On obtient donc les nouvelles valeurs u' qui sont données par :

$$\begin{cases} u'_0 = 108 \\ u'_1 = 108 + 3 = 111 \\ u'_2 = 108 + 37 = 145 \\ u'_3 = 108 + 7 = 115 \end{cases} . \quad (74)$$

III.4.2 Extraction des données

- Côté récepteur, après division de l'image marquée en blocs, on obtient le vecteur $u' = (108,111,145,115)$ résultant.

$$\begin{cases} LM_1 = 0 \\ LM_2 = 1 \\ LM_3 = 1 \end{cases} \quad (75)$$

- Récupération de la LM :

$$\begin{cases} \overline{v'_1} = 111 - 108 = 3 \\ \overline{v'_2} = 145 - 108 = 37 \\ \overline{v'_3} = 115 - 108 = 7 \end{cases} \quad (76)$$

- Calcul la différence entre les pixels de l'image marquée :

$$\begin{cases} b_1 = [3] mod [2] = 1 \\ b_2 = [37] mod [2] = 1 \\ b_3 = [7] mod [2] = 1 \end{cases} \quad (77)$$

- Récupération des données :

Donc $b = (1,1,1)$

$$\begin{cases} v'_1 = \dfrac{3-1}{2} = 1 \\ v'_2 = \dfrac{37-1}{2} = 18 \\ v'_3 = \dfrac{7-1}{2} = 3 \end{cases} \quad (78)$$

- Récupération de la différence réduite :

$$\begin{cases} LM = 0 \rightarrow v_1 = 2 \\ LM = 1 \rightarrow v_2 = 26 \\ LM = 1 \rightarrow v_3 = 7 \end{cases} \quad (79)$$

- Récupération de la différence originale et restauration des valeurs des pixels de l'image hôte. Celles-ci sont données comme suit : $u = (108,110,134,115)$.

$$\begin{cases} u_0 = 108 \\ u_1 = 108 + 2 = 110 \\ u_2 = 108 + 26 = 134 \\ u_3 = 108 + 7 = 115 \end{cases} \quad (80)$$

III.4 Implémentation de la méthode PVO-RDH sous environnement Python

L'objectif de cette partie est de montrer les résultats de l'implémentation de la méthode PVO-RDH sous environnement Python.

III.4.1 Procédure d'intégration

Pour pouvoir intégrer les bits de l'information secrètes selon l'algorithme proposé par [4], il est nécessaire de posséder, comme paramètres d'entrée, l'image originale, le message secret et les conditions de travail pour avoir en sortie l'image stégo. Les étapes du processus d'intégration sont illustrées par la figure(III.8), et sont détaillées selon l'algorithme suivant :

III.4.1.1 Algorithme d'intégration concernant la méthode PVO-RDH

- **Etape 1 (Partitionnement de l'image)**

Deviser l'image originale en k blocs non chevauchés de taille égale n $(n_1 \times n_2)$ pour chaque bloc. Ces blocs seront réorganisés comme suit : $X_i = \{ X_0 \dots . X_{k-1}\}$, $i = 0,1,2, \dots, k-1$.
Après, les valeurs de chaque bloc sont triées selon un ordre croissant pour obtenir $\{x_1, \dots, x_n\}$. La notation $\{x_1, \dots, x_i\}$ est utilisée ici pour désigner les valeurs de pixel ordonnées.

- **Étape 2 (Construction de la carte de localisation LM)**

La carte de localisation LM (overflow-underflow) est une séquence de 0 et 1. Ici, les 1 représentent les blocs où on peut intégrer des données.
Pour chaque bloc X_i, si l'une des deux cas suivants se produit,

$$\begin{cases} x_n - x_{n-1} \geq 1 \text{ et } x_n = 255 \\ x_1 - x_2 \leq -1 \text{et } x_1 = 0 \end{cases} \quad (81)$$

Alors, $LM(i) = 1$. Sinon$LM(i) = 0$.

- **Etape 3 (intégration des donnes)**

Avant de commencer l'intégration, il faut déterminer le seuil $\boldsymbol{T}$ (Threshold) qui permet de déterminer le nombre de blocs dans lesquels on peut intégrer des données. Ensuite, on intègre

successivement les données dans l'image hôte selon la procédure suivante :

- si $LM(i) = 0$, (overflow/underflow) X_i est ignoré de l'opération d'intégration;
- si $LM(i) = 1$ et $x_{n-1} - x_2 > T$, X_i est un bloc brut est exclu de l'opération d'intégration ;
- si $LM(i) = 1$ et $x_{n-1} - x_2 \leq T$, il n'y a pas de (overflow/underflow). Dans ce cas, le maximum et le minimum de X_i seront décalés ou étendus pour transporter des données :

  - Pour le maximum x_n:

$$\begin{cases} si\ x_n - x_{n-1} = 0\ donc\ x_i \text{ ne sera pas modifié} \\ si\ x_n - x_{n-1} = 1\ donc\ \ x_n\ sera\ élargi \text{ à } x_i + b \\ si\ x_n - x_{n-1} > 1\ donc\ x_n\ sera\ decalé \text{ à } \ x_n + 1 \end{cases} \quad (82)$$

  - Pour le minimum x_1:

$$\begin{cases} si\ x_1 - x_2 = 0\ donc\ x_1 \text{ ne sera pas modifié} \\ si\ x_1 - x_2 = -1\ donc\ \ x_1\ modifié\ par\ \ x_i - b \\ si\ x_1 - x_2 < -1\ donc\ x_1\ sera\ decalé\ par\ \ x_i - 1 \end{cases} \quad (83)$$

Cette étape s'arrête si tous les bits des données sont intégrés. Par conséquent, nous désignons par k_{end} l'indice du dernier bloc dernier bloc porteur de données.

- **Etape 4 (Intégration des données auxiliaires)**

On commence cette étape par l'enregistrement des bits LSB des pixels pour obtenir une séquence binaire $\boldsymbol{s_{lsb}}$ tel que :

$$len(s_{lsb}) = 16 + 2\lceil log_2(N) \rceil + I_{clm} \quad (84)$$

Avec :

I_{clm} Est le vecteur LM compréssé ;

$N = kn$ Est le nombre de pixels de l'image.

Remplacer ensuite ces LSB par les informations auxiliaires suivantes et la LM compressée définie à l'étape 2 :

- La taille des blocs (8bits) ;
- La taille de la séquence d'information (18bits) ;
- Le seuil T (8 bits) ;
- La taille de la LM (18bits) ;
- La LM compressée (m bits).

Enfin, intégrer la séquence s_{lsb} dans les blocs restants $\{X_{k_{end+1}}, \dots, X_k\}$ en utilisant le même

principe utilisé dans l'étape 3.

La figure suivante montre l'organigramme de l'intégration des données par la méthode PVO-RDH.

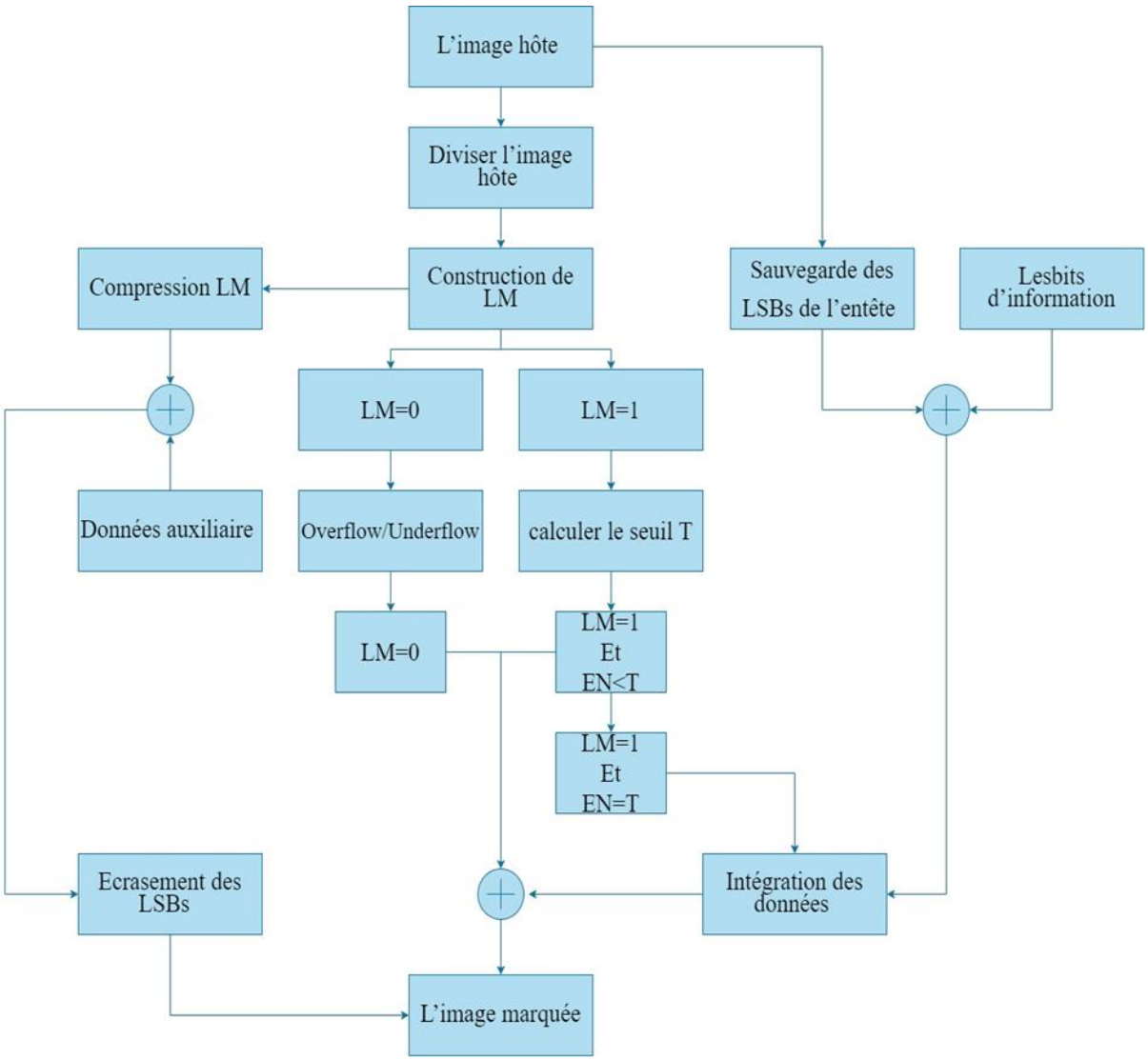


Figure.III. 1 Organigramme de l'intégration des données par la méthode PVO-RDH.

III.4.2 Procédure d'extraction

Dans ce qui suit, nous allons présenter les différentes étapes permettant de faire l'extraction des données par la méthode PVO-RDH.

III.4.2.1 Algorithme d'extraction concernant la méthode PVO-RDH

- **Etape1 (Lire et extraire les données auxiliaires) :**

Lire les LSB des premiers $16 + 2\lceil log_2(N) \rceil$ pixels de l'image marquée pour obtenir les

informations auxiliaires, y compris les valeurs de $n_1,\ n_2, T, k_{end}$ et l_{clm}. Ensuite, lire les LSBs des l_{clm} pixels suivants pour obtenir la LM compressée. La LM est ensuite obtenu par décompression de la CLM.

- **Etape2 (extraction LSB de la séquence S et restauration de l'image)**

Comme c'était le cas pour la procédure d'intégration des données, commençant dans cette étape par diviser l'image marquée en k blocs non chevauchés $\{X_0 \ldots . X_{k-1}\}$ telle que chaque X_i Contient n pixels. Dans cette étape, pour les blocs $\{X_{k_{end+1}}, \ldots, X_k\}$, on doit extraire la séquence s_{lsb} définie dans l'étape 4 donnée plus haut. En même temps, on procède à la réalisation de la reconstruction. En effet, pour un bloc X_i $(i > k_{end})$, dont les valeurs dans l'ordre croissant sont $(y_1, y_2, \ldots y_n)$:

- Si $LM(i) = 1$ et $x_{i-1} - x_2 \leq T$ donc :

Pour le maximum :

- Si $y_n - y_{n-1} > 2$: il n'y a pas de données cachées et la valeur d'origine y_n sera y_{n-1} ;
- Si $y_n - y_{n-1} \epsilon \{1,2\}$: la donnée cachée est $b = y_n - y_{n-1} - 1$ et la valeur originale de y_n est $y_n - \mathrm{b}$;
- Si $y_n - y_{n-1} = 0$: il n'y a pas de données cachées et la valeur originale de y_n est elle-même.

Pour le minimum :

- Si $y_1 - y_2 < -2$: il n'y a pas de données cachées et la valeur d'origine y_1 sera $y_1 + 1$;
- si $y_1 - y_2 \epsilon \{-2, -1\}$: la donnée cachée $b = y_2 - y_1 - 1$ et la valeur originale de y_1 est $y_1 + \mathrm{b}'$;
- $y_1 - y_2 = 0$: il n'y a pas de données cachées et la valeur originale de y_1 est elle-même.

- Sinon, il n'y a pas de données cachées et les valeurs d'origine de y_n et y_1 sont elles-mêmes.

Dans tous les cas, pour chaque valeur $j \in \{2, \ldots, n-1\}$, la valeur originale de y_j est elle-même puisqu'elle est inchangée dans l'intégration des données. Cette étape s'arrêtera si la séquence S_{LSB} est extraite.

- **Etape 3 (restaurer l'image hôte)**

Récupérer les valeurs originales des pixels concernant les blocs porteurs, on remplace les LSBs des premiers $16 + 2\lceil log_2(N) \rceil + l_{clm}$ pixels par la séquence S_{lsb} extraite à partir de l'étape 2.

Utiliser ensuite la même méthode de l'étape 2 pour extraire les données cachées des blocs $\{X_1 \ldots\ldots X_{k_{end}}\}$, et pendant ce temps réaliser la restauration pour ces blocs. Enfin, les données embarquées sont extraites et l'image hôte est récupérée. L'organigramme de l'extraction des données par la méthode PVO-RDH est montré dans la figure (III.2).

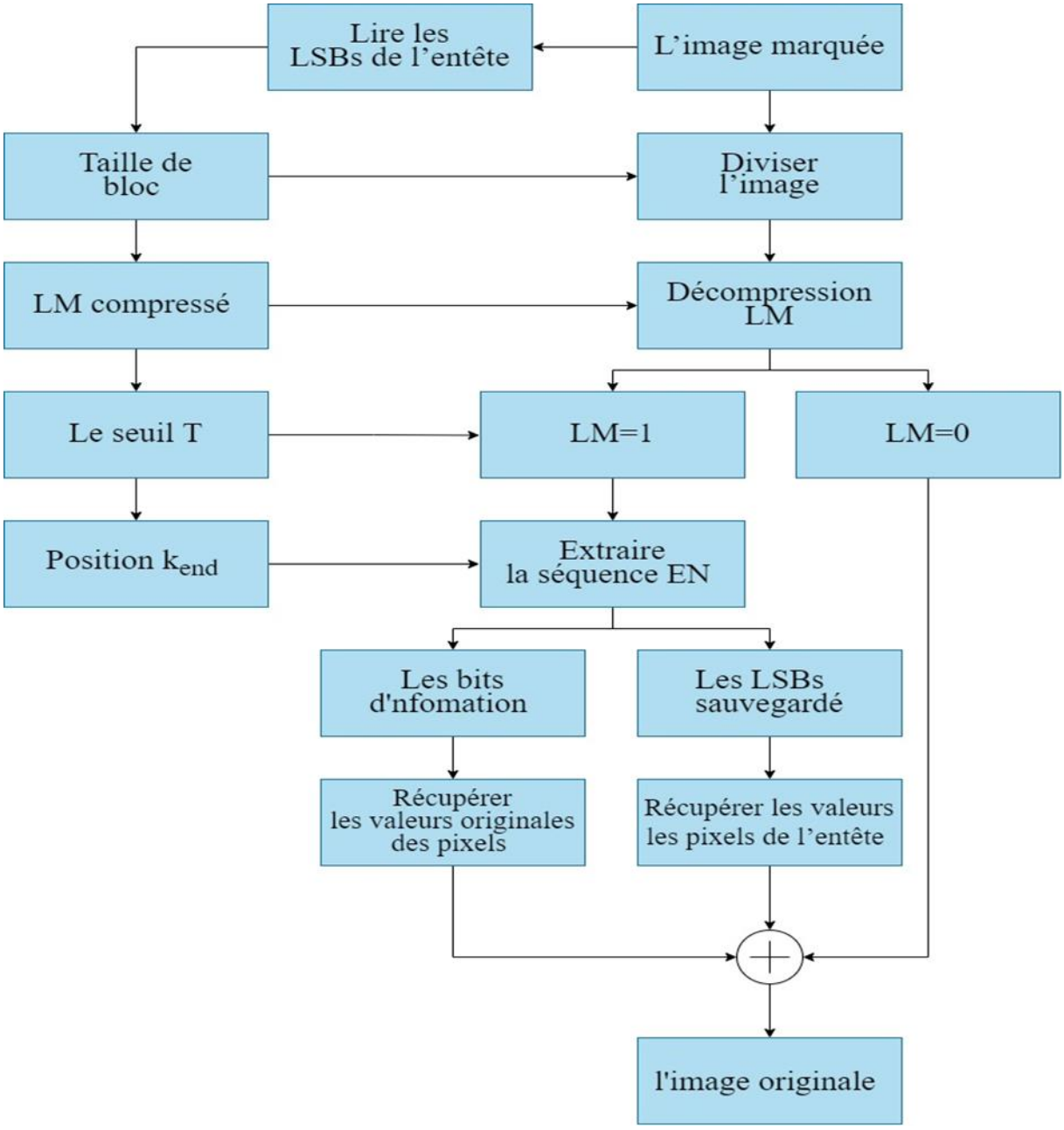


Figure.III. 2 Organigramme de l'extraction des données par la méthode PVO-RDH.

III.5 Résultats et discussion

Dans cette section, les simulations ont été effectuées sur Python pour tester les performances de la méthode PVO-RDH. En effet, en premier lieu nous voulons intégrer un message, avec une capacité d'intégration (EC) égale à 10000 bits, dans l'image Lena en niveau de gris. Par la suite, on a calculé le niveau du PSNR après intégration des données sur cette image. En deuxième lieu, on a varié la valeur de la taille EC dans l'intervalle [5000 :40000] bits dans le but de voir son effet sur la dégradation de la qualité de l'image. Pour chacun des deux cas, nous avons mesuré le rapport PSNR. Les résultats correspondant à chacune des étapes données précédemment seront montrés dans ce qui suit.

- Importation et affichage de l'image de Lena de taille(512×512).

Figure.III. 3 Image originale.

Diviser l'image en blocs de taille 2×2 chacun ($\{X_0 \ldots . X_{k-1}\}$). Cette image va donner donc :

$$\frac{512 \times 512}{4} = 65536\ blocs \tag{85}$$

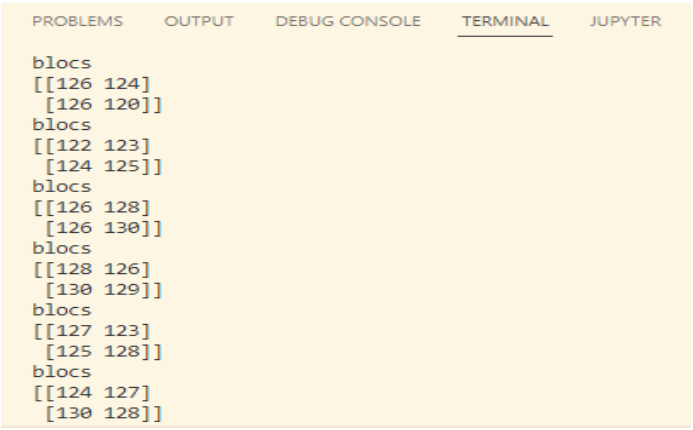


Figure. III. 4 Division de l'image en blocs.

- Ordonner les valeurs de chaque bloc selon un ordre croissant :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    JUPYTER

[[120 122 123 126]]
triage des valeurs
[[114 119 119 121]]
triage des valeurs
[[119 121 122 123]]
triage des valeurs
[[114 114 119 119]]
triage des valeurs
[[115 117 117 118]]
triage des valeurs
[[110 110 112 114]]
triage des valeurs
[[109 111 111 112]]
triage des valeurs
[[106 107 110 111]]
triage des valeurs
[[104 106 107 110]]
triage des valeurs
[[106 108 110 116]]
triage des valeurs
[[116 120 120 124]]
triage des valeurs
[[127 128 131 136]]
```

Figure. III. 5 Organisation des pixels selon un ordre croissant.

- Construire la carte de localisation LM :

Chaque valeur dans la carte de localisation représente un bloc qui satisfasse aux conditions d'overflow/underflow.

```
la carte de localisation=
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

Figure. III. 6 Carte de localisation.

- Calculer le seuil T et compresser la carte de localisation LM à l'aide de l'algorithme arithmétique. Dans notre cas, on a utilisé le codage Run-length (RLE).

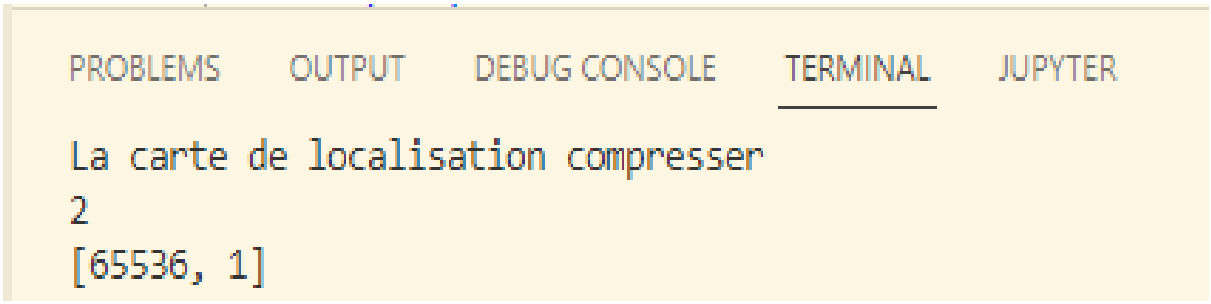


Figure. III. 7 Résultat de la compression de la carte de localisation

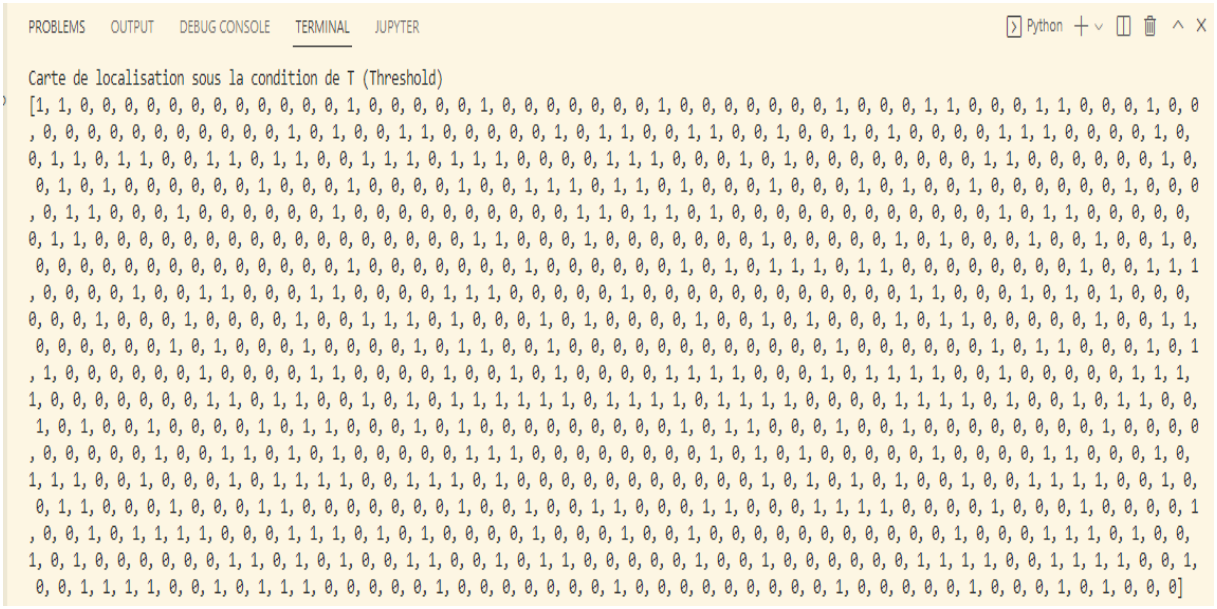


Figure. III. 8. Carte de localisation sous la condition de *T* (Threshold).

- Convertir les donnes auxiliaires en binaire puis les écraser dans les LSBs des pixels.

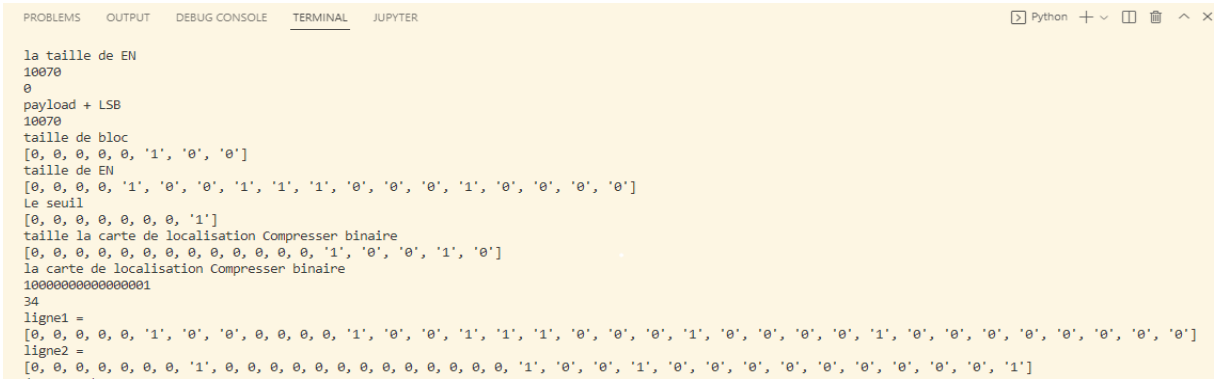


Figure. III. 9 Conversion des valeurs en binaire.

- Construction de la séquence binaire (EN) qui contient les bits d'information puis sauvegarder ca taille dans les lignes de l'entête.

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    JUPYTER                                        Python

Séquence binaire EN
[0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0
, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0,
1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0
, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0,
1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0
, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0,
1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0
, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0,
1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0
, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]
```

Figure. III. 10 Séquence binaire EN.

- Le résultat de l'écrasement de l'entête est affiché dans l'image suivante :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    JUPYTER                                        Python

ligne1 =
[0, 0, 0, 0, 0, '1', '0', '0', 0, 0, 0, 0, '1', '0', '0', '1', '1', '1', '0', '0', '0', '1', '0', '0', '0', '0', '1', '0', '0', '0', '0', '0', '0', '0', '0']
ligne2 =
[0, 0, 0, 0, 0, 0, 0, '1', 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, '1', '0', '0', '1', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1']
(512, 512)
image originale
[[160 160 160 159 161 156 161 159 162 159 160 158 154 162 158 154 156 155
  160 160 153 156 154 156 154 156 154 152 155 153 153 155 153 157 155 158
  157 164 160 165 166 165 165 171 170 174 172 169 171 171]
 [161 160 159 159 161 155 161 159 163 159 160 158 155 162 158 154 156 155
  160 160 153 156 154 156 154 156 155 152 154 154 153 156 154 157 154 158
  157 165 159 165 165 165 165 170 170 174 172 169 171 171]]
image apres ecrasement
[[160. 160. 160. 158. 160. 157. 160. 158. 162. 158. 160. 158. 155. 162.
  158. 155. 157. 155. 160. 160. 152. 157. 154. 156. 154. 156. 155. 152.
  154. 152. 152. 154. 152. 156. 154. 158. 157. 164. 160. 165. 166. 165.
  165. 171. 170. 174. 172. 169. 171. 171.]
 [160. 160. 158. 158. 160. 154. 160. 159. 162. 158. 160. 158. 154. 162.
  158. 154. 156. 154. 160. 160. 152. 157. 154. 156. 155. 156. 154. 152.
  154. 154. 152. 156. 154. 156. 155. 158. 157. 165. 159. 165. 165. 165.
  165. 170. 170. 174. 172. 169. 171. 171.]]
```

Figure.III. 11 Représentation matricielle de l'image après l'écrasement des données auxiliaires.

Figure. III. 12 Image originale après l'écrasement de l'entête.

- L'étape finale pour avoir l'image marquée, est d'insérer les bits d'information EN. Les figures (III.13) et (III.14) montrent respectivement la vue matricielle et visuelle de l'image originale après l'intégration des données.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER                    Python + ^ X

image stego
[[160 161 161 158 161 156 163 159 163 159 160 158 155 161 158 153 156 153
  159 160 153 157 154 157 154 156 154 152 155 153 153 155 153 157 155 158
  158 165 158 165]
 [161 161 159 159 161 156 161 159 162 159 160 158 153 161 158 154 156 155
  159 159 153 155 154 156 154 156 156 152 155 153 153 155 154 157 153 158
  157 165 160 165]]
image originale
[[160 161 160 158 161 156 162 159 163 159 160 158 155 161 158 153 156 154
  159 160 153 156 154 157 154 156 154 152 155 153 153 155 153 157 155 158
  158 165 159 165]
 [161 161 159 159 161 156 161 159 162 159 160 158 154 161 158 154 156 155
  159 159 153 155 154 156 154 156 155 152 155 153 153 155 154 157 154 158
  157 165 160 165]]
```

Figure.III. 13 la vue matricielle de l'image marquée.

Figure.III. 14 la vue visuelle de l'image marquée.

La figure (III.14), montre l'image resultante qui est marquée (EC=10000bits et blocs de taille 2×2). L'image marquée présente un niveau PSNR egal à 58.97dB. La courbe dans la figure 15 présnte la variation du PSNR en fonction de EC pour l'image Lena.

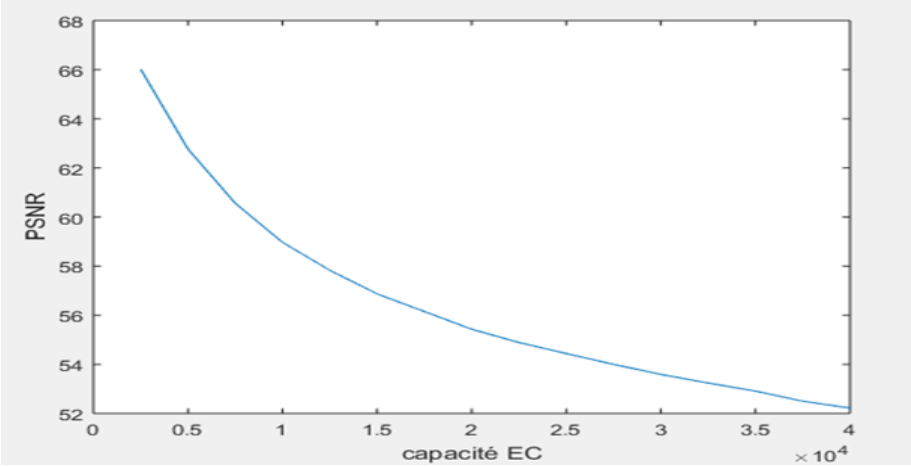


Figure.III. 15 Variation du PSNR en fonction EC pour l'image Lena.

D'après la figure (III.15), on remarque que l'augmentation d'EC entraine une réduction du PSNR. La figure16 représente les mêmes mesures de performances (PSNR en fonction de EC) pour cinq images différentes de même taille(512×512). Ces images ont été téléchargées depuis l'USC-SIPI data base.

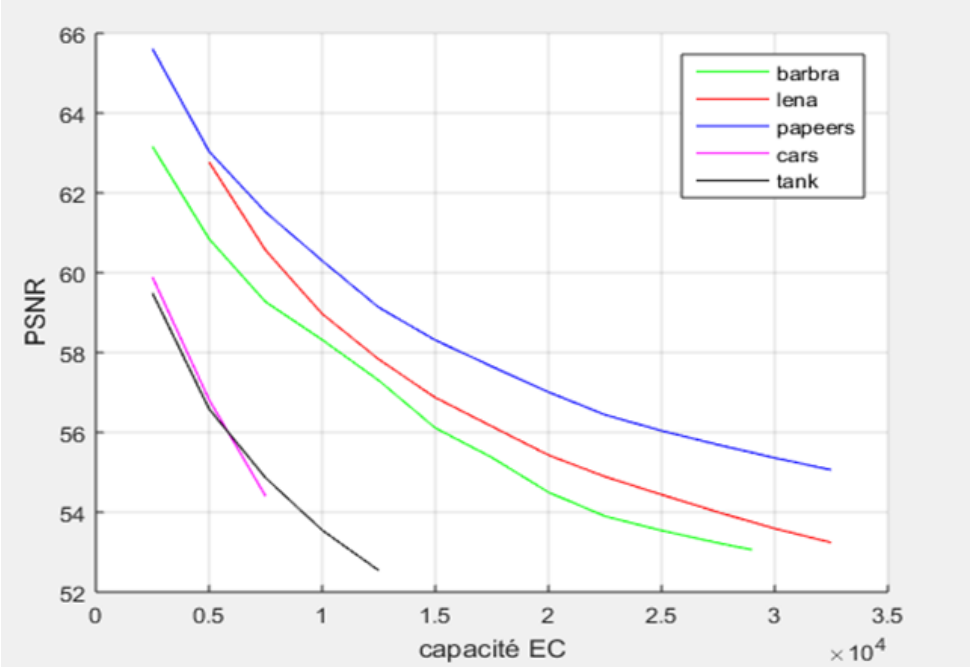


Figure.III. 16 Variation de PSNR en fonction EC.

D'après cette figure, nous remarquons que l'image poivre donne de meilleur résultat dans la qualité du PSNR par rapport aux autres images. En outre, cette image offre une capacité EC plus grande en comparaison avec les autres images. Par exemple, quand EC=40000 bits, cette image assure un PSNR supérieur à 54dB. Par contre, les autres images offrent des PSNR qui sont nettement inférieurs à 54dB. De ce fait, la nature de l'image influe directement sur la capacité EC.

III.6 Etude comparative

Dans cette partie, nous prenons les résultats de l'implémentation PVO-RDH et nous les comparons avec les résultats de quatre méthodes différentes basées sur la prédiction d'erreur et qui sont proposées dans les références Hu et al. [54], Sachnev et al. [56], Luo et al. [57], and Li et al. [55]. La méthode PVO-RDH a été appliquée sur plusieurs images en niveau de gris. Les résultats obtenus pour différentes images classiques (de taille 512 x 512 portant chacune des données d'information de 10000 bits) sont montrés dans le tableau.

Tableau III. 1 Etude comparative de la méthode PVO-RDH avec quatre autres méthodes.

Image	Hu et al. [54],	Sachnev et al. [56]	Luo et al. [57]	Li et al. [55],).	Méthode PVO-RDH
Lena	56.46	58.18	57.31	58.20	58.97
Babon	50.29	54.15	51.06	54.03	54.50
Barbara	55.21	58.15	55.74	58.61	58.32
Avion f16	57.24	60.38	57.97	61.26	59.78
Boat	53.09	55.55	54.04	55.52	56.12
Poivre	53.36	56.15	55.29	56.12	60.30
Moyenne	54.28	57.09	55.24	57.29	58.00

Concernant cette expérience, les résultats obtenus, concernant les images respectivement Lena, Babon, Boat, et Poivre, montrent des valeurs du PSNR respectivement de 58.97 dB, 54.50 dB, 56.12 dB et 60.30 dB pour la méthode PVO-RDH. Ces valeurs sont élevées par rapport à autres méthodes à l'exception les images Barbara et Avion f16 le niveau du PSNR est respectivement de 58.32dB et 59.78dB qui est légèrement inférieur par rapport aux deux méthodes Li et al. [55] et Sachnev et al. [56].

III.7 Conclusion

Dans ce chapitre, nous avons commencé par définir le langage python et ces outils de programmation. Ensuite, nous avons montré l'algorithme détaillé de la méthode PVO-RDH et le principe de son implémentation dans l'environnement Python. Nous avons exposé dans une troisième phase les résultats obtenus étape par étape. Les résultats finaux, concernant l'intégration des données, ont démontré la capacité de la méthode PVO-RDH qui a présenté des niveaux PSNR acceptables pour des ECs élevés en comparaison avec les autres méthodes.

Conclusion Générale

Conclusion Générale

Comme nous l'avons vu tout au long de ce mémoire de Master, dans le domaine de RDH, plusieurs problèmes existent notamment celui comment intégrer des données dans une image sans avoir une distorsion élevée sur l'image elle-même.

Nous avons montré que les algorithmes RDH peuvent être classés en deux catégories qui peuvent être appliqués séparément dans le domaine clair et le domaine crypté. Comme notre travail s'intéresse sur le premier domaine, nous avons étudié plusieurs techniques permettant d'intégrer des données d'information tout en améliorant le niveau du PSNR et la capacité d'intégration CE.

Le travail présenté dans ce mémoire de Master avait pour objectif d'étudier et d'implémenter l'algorithme PVO-RDH basé sur la méthode PEE. Pour réaliser ce but, nous avons étudié, en premier lieu, les versions de base de l'algorithme PVO-RDH plus particulièrement la Fonction « Plancher » et l'Extension à « Différence Réduite ». En deuxième lieu, nous avons implémenté sous environnement Python l'algorithme PVO-RDH.

Après une étude approfondie des résultats obtenus, concernant les performances du système d'intégration et d'extraction des données ainsi que la sécurité et l'efficacité de cette opération, nous avons montré que quelques images donnent de meilleurs résultats en termes de qualité du PSNR par rapport à d'autres images. En outre, elles offrent des capacités EC plus grandes en comparaison avec les autres images. D'autres images offrent des PSNR qui sont nettement très petit. Nous avons conclu donc que la nature de l'image a une influence directe sur les performances et la capacité EC.

Une étude comparative a été également réalisée pour comparer la méthode PVO-RDH avec d'autres techniques de littérature en utilisant les images Lena, Babon, Barbara, Avion f16, Boat et Poivre. Les résultats obtenus, concernant les images respectivement Lena, Babon, Boat, et Poivre, ont montré des valeurs du PSNR élevés de la méthode PVO-RDH. En revanche, deux images où le niveau du PSNR était légèrement inférieur par rapport à deux méthodes.

Référence

[1] Wengui Su1,2,3 & Xiang Wang1 & Fu Li2 & Yulong Shen3 & Qingqi Pei1,'' Reversible data hiding using the dynamic block-partition strategy and pixel-value-ordering," Multimed Tools Appl https://doi.org/10.1007/s11042-018-6410-x.

[2] J. Tian, "Reversible data embedding using a difference expansion,"IEEE Transactions on Circuits and Systems for Video Technology 13 (8) (2003) 890–896.

[3] Fridrich, M. Goljan, R. Du, "Lossless data embedding – new paradigm in digital watermarking," EURASIP Journal on Applied Signal Processing 2002 (2) (2002) 185–196.

[4] Xiaolong Li, Jian Li, Bin Li, Bin Yang, "High-fidelity reversible data hiding scheme based on pixel-value-ordering and prediction-error expansion," Signal Processing, volume 93, (2013) pp.198–205. http://dx.doi.org/10.1016/j. sigpro.2012.07.025.

[5] HAMAMA Siham, "Authentification des images médicales couleurs en utilisant le tatouage numérique, Département des Mathématiques et d'Informatique Université de Ghardaia Faculté des Sciences et de la Technologie.

[6] Ali Pacha, Adda & Said, Naima & BELGORAF, A. & M'Hamed, Abdallah. (2006). Stéganographie : Sécurité par Dissimulation.

[7] Sanjay Kumar1 · Anjana Gupta1 · Gurjit Singh Walia2," Reversible data hiding: A contemporary survey of state-of-the-art,opportunities and challenges," Applied Intelligence (2022) 52:7373–7406Applied Intelligence (2022) 52:7373–7406.

[8] Chikhi Samia née Boucherkha, ''Contribution à l'authentification souple d'images digitales par des techniques de marquage numérique Application aux images médicales,'' Département de linformatique, Université Mentouri de Constantine Faculté des sciences de l'ingénieur.

[9] Sofiane Braci. Etude des méthodes de dissimulation informées de données appliquées aux supports multimédias. Traitement du signal et de l'image [eess.SP]. Université Paris Sud - Paris XI, 2010.Français. fftel-00866202

[10] David Megías 1,Wojciech Mazurczyk and Minoru Kuribayashi,''Data Hiding and Its Applications: Digital Watermarking and Steganography,''applied sciences. 2021, 11, 10928. https://doi.org/10.3390/app112210928.

[11] Tefas, A., Nikolaidis, N., and Pitas, I. (2009).Chapter 22 –image watermarking : Techniques and applications. In The Essential Guide to Image Processing, pages 597 – 648. Academic Press.

[12] Cox, I. J., Miller, M. L., and Bloom, J. A. (2000). Watermarking applications and their properties. In Proceedings International Conference on Information Technology : Coding and Computing (Cat. No. PR00540),pp 6–10. IEEE.

[13] Belferdi, W. (2019)," A Robust Watermarking Approach for Images Authentication and Traceability,"PhD thesis, Université de Batna 2.

[14]

[15] F. Hartung et M.Kutter,"Multimedia watermarking techniques," Proceedings of the IEEE, 87(7) :1079-1107, juillet 1999.

[17] Hyoung Joong Kim,'' A Novel Difference Expansion Transform for Reversible Data Embedding,'' IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 3, NO. 3, SEPTEMBER 2008

[18] Ahmad, Tohari & Holil, Muhammad. Increasing the Performance of Difference Expansion-based Steganography when Securing Medical Data. The Smart Computing Review.(2014). 4. 10.6029/smartcr.2014.04.007.

[19] Diljith M. Thodi and Jeffrey J. Rodriguez. Reversible Watermarking by Prediction-Error Expansion

[20] M. Weinberger, G. Seroussi, and G. Sapiro, “LOCOI: A Low Complexity, Context-Based, Lossless Image Compression Algorithm:’ in Pmc.of die IEEE Data Coinpression Col$ 1996, pp. 140-149.

[21] https://python.developpez.com/tutoriels/debuter-avec-python-au-lycee/.

[22] https://creativecommons.org/licenses/by-sa/3.0/fr/

[23] de Allen B. Downey, Jeffrey Elkner & Chris Meyers disponible sur : http://thinkpython.com

[24] https://docs.python.org/fr/3/howto/pyporting.html.

[25] https://www.lebigdata.fr/python-langage-definition

[26] https://python.doctor/page-tkinter-interface-graphique-python-tutoriel.

[27] https://docs.python.org/3/library/msvcrt.html.

[28] https://pypi.org/project/moth/.

[29] John Hunt. Advanced Guide to Python 3 Programming

[30] https://en.wikipedia.org/wiki/NumPy#Basic_operations.

[31] OpenCV Dcvelopcrs Tcam : itseez. About (official website) .opencv.org/about .html. 2015-04-20..

[32] Barton JM Method and apparatus for embedding authentication information within digital data. U.S. Patent, (1997) . 5,646,997

[33] Fridrich J, Goljan M, Du R (2001) Invertible authentication. Proc SPIE Security and Watermarking of Multimedia Contents 4314:197–208

[34] Fridrich J, Goljan M, Lossless data EmbeddingNew paradigm in digital watermarking. EURASIP J Adv Signal Process . Du R (2002) .2:185–196.

[35] Celik MU, Sharma G, Tekalp AM, Saber E Lossless generalized-LSB data embedding. IEEE Transactions on Image Processing. (2005) . 14:253–266. https://doi.org/10.1109/TIP.2004.840686.

[36] Tian J Wavelet-based reversible watermarking for authentication. Proceedding SPIE. (2002) .4675:679–690. https://doi.org/ 10.1117/12.465329

[37] Alattar AM Reversible watermark using the difference expansion of a generalized integer transform. IEEE Trans Image Process. (2004).13(8):1147–1156. https://doi.org/10.1109/TIP.2004.828 418

[38] Hu Y, Lee H, Li J DE-based reversible data hiding with improved overflow location map. IEEE Trans Circ Syst Vid Technol. (2009). 19(2):250–260. https://doi.org/10.1109/TCSVT.2008. 2009252

[39] Barton JM .Method and apparatus for embedding authentication information within digital data. U.S. Patent, (1997). 5,646,997

[40] Vleeschouwer CD, Delaigle JE, Macq B Circular interpretation of histogram for reversible watermarking. 2001 IEEE Fourth Workshop on Multimedia Signal Processing (Cat. No.01TH8564). (2001). pp 345–350. https://doi.org/10.1109/MMSP. 2001.962758

[41] Lee S, Suh Y, Ho Y (2006) Reversible image authentication based on watermarking. In: 2006 IEEE International Conference on Multimedia and Expo, pp 1321–1324. https://doi.org/10.1109/ ICME.2006.262782.

[42] Shaji C, Sam S (2019) A new data encoding based on maximum to minimum histogram in reversible data hiding. The Imaging Journal 67(4):202–214. https://doi.org/10.1080/13682199.2019. 1592892.

[43] Manikandan VM, Renjith P An efficient overflow handling technique for histogram shifting based reversible data hiding. In: 2020 International Conference on Innovative Trends in Information Technology,(2020.(ICITIIT). pp 1–6. https://doi.org/10.1109/ICITIIT49094.2020.9071553.

[44] Jung KH, Yoo KY (2009) Data hiding method using image interpolation. Computer Satandards and Interfaces 31(2):465– 470. https://doi.org/10.1016/j.csi.2008.06.001

[45] Loua DC, Choub CL, Weia HY, Huang HF (2013) Active steganalysis for interpolation-error based reversible data hiding. Pattern Recogn Lett 34(9):1032–1036. https://doi.org/10.1016/j. patrec.2013.01.023

[46] Wahed MA, Nyeem H Efficient LSB substitution for interpolation based reversible data hiding scheme. In: 2017 20t h International Conference of Computer and Information Technology(ICCIT),(2017).pp1–6.https://doi.org/10.1109/ICCITECHN. 2017.8281771.

[47] Khosravi MR, Yazdi M A lossless data hiding scheme for medical images using a hybrid solution based on IBRW error histogram computation and quartered interpolation with greedy weights. Neural Computing and applications 30:2017–2028. https://doi.org/10.1007/s00521-018-3489-y

[48] Tian J Reversible data embedding using a difference expansion. IEEE Trans Circ Syst Vid Technol. (2003) .13(8):890–896. https://doi.org/10.1109/TCSVT.2003.815962

[49] Kamstra L, Heijmans HJAM Reversible data embedding into images using wavelet techniques and sorting. IEEE Trans Image Process. (2005). 14(12):2082–2090. https://doi.org/10.1109/TIP. 2005.859373

[50] Thodi DM, Rodriguez JJ Prediction-error based reversible watermarking.International Conference on Image Processing, 2004 ICIP ’04 3:1549–1552. https://doi.org/10.1109/ICIP.2004. 1421361

[51] Sachnev V, Kim HJ, Nam J, Suresh S, Shi YQ Reversible watermarking algorithm using sorting and prediction. IEEE Trans Circ Syst Vid Technol 19(7):989–999. https://doi.org/10.1109/TCSVT.2009.2020257

[52] Ni N, Shi YQ, Ansari N, Su W Reversible data hiding. IEEE Trans Circ Syst Vid Technol 16(3):354–362. https://doi.org/10.1109/TCSVT.2006.869964

[53] Vleeschouwer CD, Delaigle JE, Macq B Circular interpretation of histogram for reversible watermarking. 2001 IEEE Fourth Workshop on Multimedia Signal Processing (Cat. No.01TH8564), pp 345–350. https://doi.org/10.1109/MMSP. 2001.962758

[54] Y. Hu, H.K. Lee, J. Li, DE-based reversible data hiding with improved overflow location map, IEEE Transactions on Circuits and Systems for Video Technology 19 (2) (2009) 250–260.

[55] X. Li, B. Yang, T. Zeng, Efficient reversible watermarking based on adaptive prediction-error expansion and pixel selection, IEEE Transactions on Image Processing 20 (12) (2011) 3524–3533.

[56] V. Sachnev, H.J. Kim, J. Nam, S. Suresh, Y.Q. Shi, Reversible watermarking algorithm using sorting and prediction, IEEE Transactions on Circuits and Systems for Video Technology 19 (7) (2009) 989–999.

[57] L. Luo, Z. Chen, M. Chen, X. Zeng, Z. Xiong, Reversible image watermarking using interpolation technique, IEEE Transactions on Information Forensics and Security 5 (1) (2010) 187–193.