

الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement Supérieur et de la Recherche scientifique



Université Mohamed Khider Biskra
Faculté des Sciences et de la Technologie
Département de Génie Electrique
Filière : Électronique
Option : Télécommunication

Réf:.....

**Mémoire de Fin d'Etudes
En vue de l'obtention du diplôme:**

MASTER

Thème

COMPRESSION D'IMAGE SANS PERTES PAR JPEG-LS

Présenté par :
BELLABIDI ADEL MOUNIR
Soutenu le : 06 Juin 2013

Devant le jury composé de :

Mr Abdesselam Salim

MAA

Président

Mr Ouafi Abdelkrim

MCA

Encadreur

Melle Medouakh Sadia

MAB

Examinateur

Année universitaire : 2012 / 2013

الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement Supérieur et de la recherche scientifique



Université Mohamed Khider Biskra
Faculté des Sciences et de la Technologie
Département de Génie Electrique
Filière : Électronique
Option : Télécommunication

Mémoire de Fin d'Etudes
En vue de l'obtention du diplôme:

MASTER

Thème

COMPRESSION D'IMAGE PAR SANS PERTES JPEG-LS

Présenté par :

Bellabidi Adel Mounir

Avis favorable de l'encadreur :

Ouafi Abdelkrim

signature

Avis favorable du Président du Jury

Abdesselam Salim

Signature

Cachet et signature



Université Mohamed Khider Biskra
Faculté des Sciences et de la Technologie
Département de Génie Electrique
Filière : Électronique
Option : Télécommunication

Thème :

COMPRESSION D'IMAGE SANS PERTES PAR JPEG-LS

Proposé par : Bellabidi Adel Mounir

Dirigé par : Oaui Abdelkrim

Résumé

Depuis l'avènement du numérique, la compression d'images est une question centrale dans des domaines de plus en plus nombreux. Des algorithmes standards, comme le JPEG, proposent un moyen de compression des données avec pertes. Par ailleurs, le besoin en matière de compression sans perte se fait ressentir. C'est pourquoi, cette voie de recherche est intensivement explorée et donne naissance à un nombre d'algorithmes très efficaces.

Parmi les techniques connues dans la littérature, on trouve le codage connu sous le nom de DPCM sans perte ou encore le codage prédictif sans perte. C'est le cas aussi de LOCO-I implanté dans JPEG-LS, CALIC et d'autres.

Dans JPEG-LS (notre travail dans ce mémoire) la compression est réalisée par la combinaison d'un codage adaptatif (extension des codes de Golomb) avec un codeur entropique proche du codeur de Huffman pour les zones à faible entropie.

Mots clés : compression sans pertes, contexte, prédiction, codage de Golomb, codage de Huffman

الملخص

منذ ظهور ضغط الصور الرقمية هي قضية رئيسية في مجالات أكثر وأكثر. الخوارزميات القياسية، مثل JPEG، وتوفير وسيلة لضغط البيانات مع الخسائر. علاوة على ذلك، شعرت بالحاجة لضغط ضياع. ولذلك، هذا النوع من البحوث هو استكشاف على نطاق واسع ويثير عدد من خوارزميات فعالة جدا. من بين التقنيات المعروفة في الأدب، نجد الترميز DPCM المعروف دون فقدان أو ضياع الترميز التنبؤي. هذا هو الحال من LOCO-I تعمل في JPEG-LS، CALIC وغيرهم أيضا. (JPEG-LS عملنا في هذه المواصفات (يتم تنفيذ ضغط من قبل مجموعة من التكييف الترميز) رموز غولومب من التمديد (مع ترميز الكون بالقرب من ترميز هوفمان إلى مناطق الكون منخفضة. كلمات الدلالية: ضياع ضغط، سياق التنبؤ Golomb الترميز، ترميز هوفمان

Dédicace

Je dédie ce modeste travail à :

Ma chère mère

Mon père : Mohamed Sarbi

Mes frères : Chafik, Saïd, Babi, Lazi

Mes sœurs

Toute ma grande famille

Mes oncles : Ali, Mahmoud, Karim, Abdelaziz

Aux étudiants du groupe de télécommunications spécialement

Mes amis

*Abdelmoumen, Hicham, Alla, Nori, Salah, Soufisma, Soufiane, Mano
, Fares, Mounir, Mourad, Dohi, Hosem, Taki, Bilal, Fouad, Majdi,*

Moundir, Djoubir, Mouhsen,, Islem, Elshafed, Imad, Bachir

Et d'autres à tous que je connaisse de près ou de loin

Adel Mounir

Remerciements

Avant tous je remercie ALLAH tout puissant qui ma donnée la force d'avoir accomplir ce travail et d'être vécu jusqu'à ce jour.

Ce mémoire achève mes études de master à l'université de Biskra, cela représente pour moi l'occasion d'exprimer ma reconnaissance pour les personnes qui m'ont aidé à arriver jusqu'à cette étape.

Tout d'abord, je souhaite exprimer ma vive gratitude au docteur Ouafi Abdelkrim pour m'avoir assuré un cadre de travail excellent et pour sa disponibilité, sa patience et sa grande expérience.

Je tiens particulièrement à remercier M. Boukhari Djamel d'avoir co-encadré cette mémoire dont l'aide, les indications, les remarques et les conseils ont été indispensables et précieuses pour la réalisation de ce travail.

Je remercie Monsieur Abdesslem Salim qui me fait l'honneur de présider ce jury de mémoire

Je tiens à remercier Mademoiselle Medouakh S membre de jury pour l'attention et le temps consacrés.

J'exprime également toute ma sympathie et ma gratitude à tous les enseignants du département de génie électrique

Résumé

Depuis l'avènement du numérique, la compression d'images est une question centrale dans des domaines de plus en plus nombreux. Des algorithmes standards, comme le JPEG, proposent un moyen de compression des données avec pertes. Par ailleurs, le besoin en matière de compression sans perte se fait ressentir. C'est pourquoi, cette voie de recherche est intensivement explorée et donne naissance à un nombre d'algorithmes très efficaces.

Parmi les techniques connues dans la littérature, on trouve le codage connu sous le nom de DPCM sans perte ou encore le codage prédictif sans perte. C'est le cas aussi de LOCO-I implanté dans JPEG-LS, CALIC et d'autres.

Dans JPEG-LS (notre travail dans ce mémoire) la compression est réalisée par la combinaison d'un codage adaptatif (extension des codes de Golomb) avec un codeur entropique proche du codeur de Huffman pour les zones à faible entropie.

Mots clés : compression sans pertes, contexte, prédiction, codage de Golomb, codage de Huffman.

ملخص

منذ ظهور ضغط الصور الرقمية هي قضية رئيسية في مجالات أكثر وأكثر. الخوارزميات القياسية، مثل JPEG، وتوفير وسيلة لضغط البيانات مع الخسائر. علاوة على ذلك، شعرت بالحاجة لضغط ضياع. ولذلك، هذا النوع من البحوث هو استكشاف على نطاق واسع ويثير عدد من خوارزميات فعالة جدا. من بين التقنيات المعروفة في الأدب، نجد الترميز DPCM المعروف دون فقدان أو ضياع الترميز التنبؤي. هذا هو الحال من LOCO-I تعمل في JPEG-LS، CALIC وغيرهم أيضا. (JPEG-LS عملنا في هذه المواصفات (يتم تنفيذ ضغط من قبل مجموعة من التكيف الترميز) رموز غولومب من التمديد (مع ترميز الكون بالقرب من ترميز هوفمان إلى مناطق الكون منخفضة. كلمات الدلالية: ضياع ضغط، سياق التنبؤ Golomb الترميز، ترميز هوفمان

Liste des Tableaux

Chapitre 1 :Généralité sur la compression d'image

Tableau 1.1 : Modèles prédictifs :.....15

Chapitre 2 :Algorithme de JPEG-LS

Tableau 2.1: Utilisation des 7 prédicteurs :.....21

Tableau 2.2 : Code unaire :.....28

Tableau 2.3 : Code binaire tronqué :.....29

Tableau 2.4: Codage de Golomb :.....30

Tableau 2.5: Code Golomb-Rice :.....32

Tableau 2.6 : Comparaison de performances de différents algorithmes :.....36

Chapitre 3 :Simulation et résultats

Tableau 3.1: Résultats et comparaison avec différents algorithmes :.....47

Liste des Figures

CHAPITRE 1 : Généralité sur la compression d'image

Figure .1.1 : Exemple de l'arbre de Huffman.....	9
Figure.1. 2 : La compression JPEG.....	13
Figure.1. 3 : Principe de la Compression JPEG sans perte :.....	13
Figure.1. 4 : JPEG sans perte :.....	14
Figure.1. 5 : Lossless versus Lossy :.....	16

CHAPITRE 2 : Algorithme de JPEG-LS

Figure.2.1 : Diagramme en bloc pour LOCO-I :.....	19
Figure.2.2 : Model causal du JPEG-LS :.....	20
Figure.2.3 : diagramme en bloc de JPEG-LS :.....	22
Figure.2.4 : Modèle de codeur DPCM :.....	25
Figure.2.5 : Codage prédictif sans perte :.....	25
Figure.2.6 : La distribution géométrique TSGD :.....	27
Figure.2.7 : Sélection du Mode :.....	33
Figure.2.8 : Niveau bit :.....	35

CHAPITRE 3 : Simulation et résultats

Figure.3.1 : Compléxécité algorithmique :.....	41
Figure.3.2 : Organigramme global de JPEG-LS:	43
Figure.3.3 : Organigramme de calcul de l'erreur de prédiction :.....	44
Figure.3.4 : Organigramme de codage de golomb :.....	45
Figure.3.5 : Organigramme du mode run :.....	46

Liste des abréviations

DCT: Discrete Cosine transform

DWT: Discret Wavelet Transform.

DPCM : Difference Pulse Coded Modulation

GPO2 : Golomb-Power-Of-2

IEC :Images Exigences Compression

ISO: International Standarts Organisation

JPEG : Joint Photographic Experts Group.

JPEG 2000: Joint Photographic Experts Group 2000

JPEG-LS : Joint Photographic Experts lossless

LOCO-I : Low Complexity Lossless Compression for Images

LZ: : Lempel-Ziv

LZW: Lempel-Ziv-Welch.

MCU : Minimal Coded Unit

PM: Context Mixing

PNG: Portable Network Graphic

PPM : Prediction by Partial Matching

RC: Ratio Compression

RLE : Run Length Encoding

TIFF: Tagged Image File Format

TSGD : Two- sided géometric distribution

Sommaire

Sommaire

Introduction générale.....	01
<i>Chapitre1 : Généralité sur la compression d'image</i>	
1.1 Introduction.....	04
1.2 Types de Compression.....	04
1.2.1 Compression avec pertes.....	04
1.2.2 Compression sans pertes	04
1.2.3 Compression symétrique et asymétrique.....	05
1.3 Techniques de compression sans pertes.....	05
1.3.1 Compression différentiel sans pertes.....	06
1.3.2 Les transformées sans pertes.....	06
1.3.3 Codages Sans pertes	07
1.4 Compression presque sans pertes.....	11
1.5 La Norme JPEG.....	12
1.6 Le Mode sans pertes.....	13

1.6.1 Mode de fonctionnement.....	14
I.6.2 Le prédicteur JPEG sans pertes.....	14
I.7 Inconvénients de l’algorithme JPEG.....	15
I.8 Lossless versus Lossy.....	16
1.9 JPEG 2000 sans pertes.....	16
1.10 Conclusion	17

Chapitre 2 : Algorithme de JPEG-LS

2.1 Introduction.....	19
2.2 Algorithme LOCO-I	19
2.2.1 Décorrélation	20
2.2.2 Le prédicteur médian.....	20
2.3 Description détaillée de l'algorithme JPEG-LS.....	21
2.3.1 Détermination contexte.....	22
2.3.2 Calcul du nombre contexte.....	23
2.3.3 Gradients locaux	23
2.3.4 Modélisation du contexte	24
2.4 Codage différentiel	24
2.5 Codage prédictif	25
2.6 Calcul du résiduel.....	26

Sommaire

2.7 paramétrage.....	26
2.8 Algorithme de codage.....	26
2.8.1 Le Model TGSD.....	27
2.8.2 Codage des distributions.....	27
2.8.3 Correction adaptative.....	28
2.9 Codage de golomb.....	28
2.9.1 Principe.....	28
2.9.2 Codage binaire tronqué.....	29
2.9.3 Optimalité.....	30
2.9.4 Codage de Rice	31
2.10 La sélection du mode.....	33
2.10.2 Extension d'alphabet	34
2.10.3 L'optimisation des étapes de modélisation	34
2.11 Le codage RLE	34
2.11.1 Le niveau Bit.....	35
2.11.2 Codage RLE dans les zones uniformes.....	35
2.12 Avantages de l'algorithme JPEG-LS.....	36
2.13 Conclusion.....	37

Chapitre 3 : Simulation et résultats

3.1 Introduction.....	39
3.2 Paramètres de validation	39
3.2.1 Quantité d'information	39
3.2.2 L'entropie.....	39
3.2.3 Taux de compression (RC)	40
3.3.5 Complexité Algorithmique	40
3.2.6 Fonctionnalités.....	41
3.3 Mise en oeuvre de JPEG-LS.....	42
3.4 Organigramme Global de l'algorithme JPEG-LS.....	42
3.4.1 Calcul de l'erreur de prédiction.....	44
3.4.2 Organigramme de codage de golomb.....	45
3.4.3 Mode run.....	46
3.5 Résultats obtenus	47
3.6 Discussion	48
3.7 Conclusion	48
Conclusion générale	49
Perspectives.....	50
Bibliographie	51

Introduction générale

Introduction générale

L'homme a toujours voulu découvrir la beauté des planètes, vues de l'espace. Il a donc envoyé des satellites capables de photographier celles-ci. Mais l'un des problèmes majeur est la transmission de ces photos vers la terre. Une technique employée dans de nombreux domaines est la compression sans perte des informations. Bien que peu utilisé par la communauté informatique en général, sert surtout pour la transmission d'images médicales et militaires pour éviter de confondre des artefacts purement liés à l'image et à son stockage.

Malgré l'augmentation de la taille des medias, qu'ils soient disques durs internes, CDs, ou DVDs, il est clair que ce volume est impossible de donnée à stocker sur des supports actuels à accès convenable, alors il faut faire une compression pour atteindre un meilleur taux de fidélité d'une capacité de stockage ou de transmission disponible. L'utilisation d'algorithmes de compression d'images permettent en effet, une réduction importante de la quantité de données aussi que la bande passante.

Il est à noter qu'il existe une forme de codage JPEG sans perte appelé JPEG Lossless qui est à quelques points de pourcentage des meilleurs taux de compression disponibles à un niveau de complexité beaucoup plus faible. L'algorithme JPEG-LS est l'une des normes récemment désignés pour la compression sans perte de niveaux de gris et des images en couleur. Il est basé sur un modèle de contexte fixe simple, qui se rapproche de la capacité des techniques plus complexes universels pour capturer les dépendances d'ordre supérieur.

Ce mémoire est consacré à l'étude et l'implémentation de l'algorithme JPEG sans perte dit JPEG-LS.

Organisation du mémoire

Nous avons choisi d'articuler notre étude autour de trois chapitres principaux :

Le premier chapitre est consacré à la présentation générale concernant la compression sans perte, particulièrement la compression JPEG dont une étude complète de cette norme de compression nous permet de bien comprendre la structure et la procédé de la JPEG sans pertes.

Le second chapitre présente un détail sur l'algorithme JPEG-LS, nous discutons les principes qui sous-tendent à la conception de LOCO-I et sa normalisation en JPEG-LS.

Introduction générale

Dans le troisième chapitre, nous nous intéressons aux résultats de simulation pour la compression sans perte. Les résultats de différents types d'images sont présentés afin d'étudier l'efficacité de l'algorithme pour un certain nombre d'applications, nous aborderons un aspect plus mathématique, concernant les théorèmes fondamentaux utilisés pour la compression. Enfin, la conclusion générale résumera nos contributions et donnera quelques perspectives sur les travaux futurs.

Chapitre I

1.1 Introduction

La compression est l'action utilisée pour réduire la taille physique d'un bloc d'information. En compressant des données, on peut placer plus d'informations dans le même espace de stockage, ou utiliser moins de temps pour le transfert à travers d'un réseau téléinformatique. Parce que généralement les images requièrent une place importante, la compression est devenue part intégrante des données graphiques. Presque tous les formats de fichiers graphiques utilisent l'une ou l'autre méthode de compression. On rencontre souvent la compression de données comme étant une partie de l'encodage de données au même titre que le cryptage de données et la transmission de données.

1.2 Types de Compression

On considère généralement la compression comme un algorithme capable de comprimer énormément de données dans un minimum de place (compression physique), mais on peut également adopter une autre approche et considérer qu'en premier lieu un algorithme de compression a pour but de recoder les données dans une représentation différente plus compacte contenant la même information (compression logique).

La distinction entre compression physique et logique est faite sur la base de comment les données sont compressées ou plus précisément comment est-ce que les données sont réarrangées dans une forme plus compacte. Afin de gagner de la place, on est amené à comprimer les fichiers, mais le résultat n'est pas toujours idéal. Il faut donc faire un choix en connaissance de cause, il y a deux types de compression :

- La compression avec pertes
- La compression sans pertes

1.2.1 Compression avec pertes

Les algorithmes de compression avec pertes s'appliquent généralement aux données ayant de forts taux de redondance, comme les images, ou les sons. L'œil humain est limité dans le nombre de couleurs qu'il est capable de percevoir simultanément et particulièrement si ces couleurs ne sont pas adjacentes dans l'image ou sont très contrastées. Un algorithme de compression intelligent peut tenir compte de ces limitations, analyser une image sur ces bases et effectuer une réduction significative de la taille des données basée sur la suppression de l'information de certaines couleurs difficilement perceptibles par la plupart des gens.

1.2.2 Compression sans pertes

La compression est dite sans perte lorsqu'il n'y a aucune perte de données sur l'information d'origine. Il y a autant d'information après la compression qu'avant, elle est seulement réécrite d'une manière plus concise (c'est par exemple le cas de la compression gzip pour n'importe quel type de données ou du format PNG pour des images synthétiques destinées au Web). La compression sans perte est dite aussi compactage.

Il n'existe pas de technique de compression de données sans perte universelle, qui pourrait compresser n'importe quel fichier : si une technique sans perte compresse au moins un fichier, alors elle en « grossit » également au moins un. Les formats de fichier de compression sans perte sont connus grâce à l'extension ajoutée à la fin du nom de fichier (« nomdefichier.zip » par exemple), d'où leur dénomination très abrégée.

1.2.3 Compression symétrique et asymétrique

Les algorithmes de compression peuvent être divisés en deux catégories distinctes symétrique et asymétrique. Une méthode de compression symétrique utilise le même algorithme, et demande la même capacité de calcul, aussi bien pour la compression que pour la décompression. Les méthodes de compression asymétriques demandent plus de travail dans une direction que dans l'autre. Normalement, l'étape de compression demande beaucoup plus de temps et de ressources systèmes que l'étape de décompression.

Dans la pratique, cela prend tout son sens : par exemple, imaginons que nous ayons une base de données où les données seront compressées une seule fois mais décompressées un grand nombre de fois pour la consultation, alors on pourra certainement tolérer un temps beaucoup plus grand pour la compression, dans le but de trouver le taux de compression le plus élevé, que pour la décompression, où la vitesse est prédominante. Un algorithme asymétrique qui utilise beaucoup plus de temps CPU pour la compression mais qui est beaucoup plus rapide à la décompression serait un bon choix dans ce cas là.

1.3 Techniques de compression sans pertes

Parmi les algorithmes de compression presque sans perte, on retrouve la plupart des algorithmes de compression sans perte spécifiques à un type de données particulier. Par exemple, JPEG-LS [1] permet de compresser presque sans perte du Windows bitmap ,et Monkey's Audio [2] permet de compresser sans perte les données audio du wave PCM : il n'y a pas de perte de qualité, l'image et le morceau de musique sont exactement ceux d'origine. Les algorithmes tels

que Lempel-Ziv [3] ou le codage RLE [4] consistent à remplacer des suites de bits utilisées plusieurs fois dans un même fichier. Dans l'algorithme de codage de Huffman plus la suite de bits est utilisée souvent, plus la suite qui la remplacera sera courte.

Les algorithmes tels que la transformée de Burrows-Wheeler sont utilisés avec un algorithme de compression [5]. De tels algorithmes modifient l'ordre des bits de manière à augmenter l'efficacité de l'algorithme de compression, mais sans compresser par eux-mêmes.

1.3.1 Compression différentiel sans pertes

Le principe est de comparer chaque pixel p à un pixel de référence, qui est un des pixels précédemment encodés dans les voisins immédiats, et de l'encoder en deux parties :

- Un préfixe, qui est le nombre des bits les plus significatifs de p et qui sont identiques à ceux du pixel de référence.
- Un suffixe, qui est le nombre des bits restants les moins significatifs de p .

1.3.2 Les transformées sans pertes

La plupart des schémas de compression sans perte sont basés sur les techniques de prédiction, ou de variantes de celles-ci. La compression avec pertes fait appel le plus souvent aux transformations générales du type DCT (Discrete Cosine transform) [6], transformée en ondelettes DWT [7] qui appliquent la transformation linéaire sur les données d'entrée et quantifie ensuite les coefficients résultants.

En théorie, ces techniques peuvent être rendues "sans perte" en quantifiant suffisamment finement les coefficients, ou par l'encodage sans perte de l'image d'erreur. Malheureusement, ces stratégies conduisent à un taux de compression négligeable, voire un taux de compression inférieur à 1.

Ainsi, il est difficile de concevoir un codeur sans perte basé sur des transformées non triviales. Cependant, avec l'introduction de la transformée « rounding transform », il est possible de retrouver de façon sans perte une donnée à partir des coefficients de transformation quantifiés.

I.3.3 Codages Sans pertes

a. Codage Move-to-Front

Il est possible de procéder à un pré-calcul lors de la transformation d'un caractère ASCII en sa valeur entre 0 et 255 : en modifiant à la volée l'ordre de la table. Par exemple, dès qu'un caractère apparaît dans la source, il est d'abord codé par sa valeur, puis il passe en début de liste, et sera codé dorénavant par un 0, tous les autres caractères étant décalés d'une unité. Ce «Move-To-Front » [8] permet d'avoir plus de codes proches de 0 que de 255. Ainsi, l'entropie est modifiée. Par exemple, la chaîne « aaaaffff » peut être modélisée par une source d'entropie 1, si la table est (a, b, c, d, e, f,), elle est alors codée par « 00005555 ». L'entropie du code est 1. C'est aussi l'entropie de la source. Mais le code d'un Move-To-Front sera «00005000», d'entropie $H = 7/8 \log_2(8/7) + \log_2(8)/8 = 0,55$.

Le code lui-même est alors compressible, c'est ce qu'on appelle la réduction d'entropie. Le décodage est simple : à partir du même tableau initial, il suffit d'émettre le caractère correspondant à l'indice et de ranger le tableau en passant ce caractère en premier. Le tableau évolue exactement comme pendant la phase de codage.

b. Codage RLE

L'algorithme RLE est un algorithme extrêmement simple permettant de diminuer l'entropie de données [4]. Le principe consiste à détecter les répétitions et à les encoder différemment. En pratique, une chaîne répétée est encodée sur deux octets:

Le premier annonce le nombre de répétitions; le second indique le caractère à répéter. Par exemple, la chaîne "aaaaaaaa" peut être codée "8a".

Le problème est bien évidemment qu'un fichier ne contenant aucune répétition aura une taille deux fois plus importante que l'original. En pratique, on encode les répétitions sur trois caractères: le premier est un caractère spécial indiquant la présence d'une répétition; le second indique le nombre d'occurrences et le troisième la valeur à répéter. Ainsi la chaîne "aiiiiiibcddddde" sera encodée, avec pour caractère spéciale le signe dièse #: "a#6ibc#5de", ce qui représente une compression de 33,3%. Le codage RLE est notamment employé dans les formats d'image PCX ou BMP, ou bien avant un autre algorithme de compression (notamment HUFFMAN dans le cas de JPEG).

c. Codage par modélisation de contexte

c.1 Prédiction par reconnaissance partielle

La prédiction par reconnaissance partielle se base sur une modélisation de contexte pour évaluer la probabilité des différents symboles [9]. En connaissant le contenu d'une partie d'une source de données (fichier, flux...), un PPM est capable de deviner la suite, avec plus ou moins de précision. Un PPM peut être utilisée en entrée d'un codage arithmétique par exemple, il donne en général de meilleurs taux de compression que des algorithmes à base de Lempel-Ziv (voir section 1. e,f,j,h), mais est sensiblement plus lent.

c.2 Pondération de contextes

La pondération de contextes [10] consiste à utiliser plusieurs prédicteurs (par exemple des PPM) pour obtenir l'estimation la plus fiable possible du symbole à venir dans une source de données (fichier, flux...). Elle peut être basiquement réalisée par une moyenne pondérée, mais les meilleurs résultats sont obtenus par des méthodes d'apprentissage automatique comme les réseaux de neurones, elle est très performante en termes de taux de compression, mais est d'autant plus lente que le nombre de contextes est important. Actuellement, les meilleurs taux de compression sont obtenus par des algorithmes liant pondération de contextes et codage arithmétique, comme PAQ.

d. Codages entropiques

Les codages statistiques utilisent la fréquence de chaque caractère de la source pour la compression, et en codant les caractères les plus fréquents par des mots plus courts, se positionnent proches de l'entropie.

d.1 codage de Huffman

Le codage de Huffman permet de créer des codes à longueur variable sur un nombre entier de bits. La procédure pour construire l'arbre est simple et élégante. Les symboles sont disposés individuellement sous forme d'une chaîne de nœuds et de feuilles et sont connectés par un arbre binaire. Chaque nœud a un poids qui est la fréquence ou la probabilité d'apparition des symboles. L'arbre est ensuite créé suivant les étapes suivantes (voir exemple figure 1.1) :

- Les deux nœuds libres de poids les plus faibles sont identifiés.
- Un nœud parent de ces deux nœuds est créé ; il obtient un poids égal à la somme de celui de ces deux fils.

- Le nœud parent est ajouté à la liste des nœuds libres, et les deux fils sont enlevés.
- Un des deux nœuds fils est désigné comme le chemin pris à partir du nœud parent pour décoder un bit 0, l'autre nœud étant alors pris pour décoder un bit 1.
- Les étapes précédentes sont répétées tant qu'il reste plus d'un seul nœud libre. Ce nœud devient alors la racine de l'arbre.

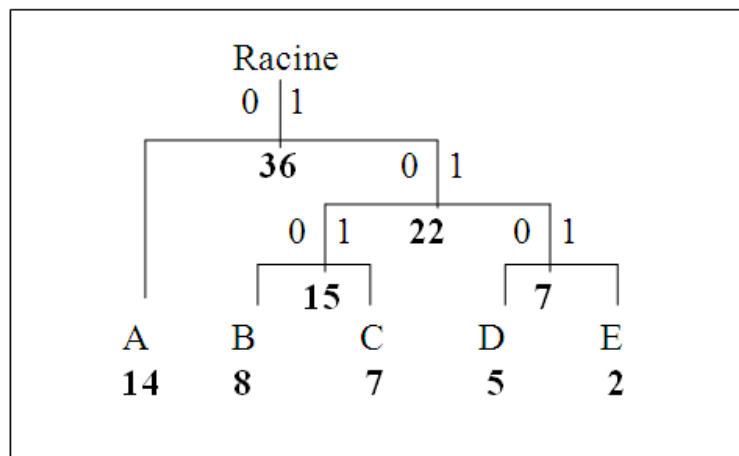


Figure.1.1. Exemple de l'arbre de Huffman

d.2 Codage d'Huffmann adaptatif

L'idée est de reconstruire l'arbre de codage à chaque nouveau caractère reçu. L'inconvénient est que l'on ne peut pas savoir au préalable de quels symboles va se composer le message, et qu'il est donc nécessaire de pouvoir ajouter en cours de compression de nouveaux symboles, et donc de nouveaux codes, à l'arbre de codage. On débutera l'algorithme avec une table de symboles pratiquement vide, à l'exception de deux symboles particuliers qui sont <NEW> et <EOF>. Le premier sera utilisé pour la définition dynamique de nouveaux symboles, le second pour marquer la fin du message. Ces deux symboles reçoivent arbitrairement, au départ, un "poids" de 1. A chaque fois que l'on rencontre un symbole non encore présent dans l'arbre, on va le signaler en émettant un symbole <NEW> suivi de la définition du symbole (par exemple, sa représentation non compressée en code ASCII). Ce nouveau symbole sera ensuite inséré dans l'arbre de codage.

d.3 Codage arithmétique

Le codage arithmétique est assez similaire au codage de Huffman en ceci qu'il associe aux motifs les plus probables les codes les plus courts (entropie). Contrairement au codage de Huffman qui produit au mieux des codes de 1 bit, le codage arithmétique peut produire des codes inférieurs à 1 bpp (bit par pixel) [11].

Le taux de compression obtenu est par conséquent meilleur. Nous pouvons regarder les codes arithmétiques comme une alternative au codage de Huffman. Le principe de base du codage est similaire : coder les messages plus probables avec des mots de code plus courts et utiliser les mots de code plus longs pour des messages moins probables. Il est à noter que le codage arithmétique associe un mot de code à chaque message, c'est-à-dire à une suite de symboles. Grâce à cette propriété il peut surpasser très légèrement le taux de compression des codes de Huffman, en associant un nombre fraction de bits à chaque symbole.

Le codage et le décodage nécessite de connaître les probabilités d'occurrence des symboles différents. De même que pour le codage de Huffman, on peut l'utiliser comme un code statique, semi-statique ou dynamique avec les mêmes avantages et inconvénients. Sa complexité est plus faible dans la configuration dynamique que pour le codage de Huffman dynamique.

e. LZ77 (Lempel-Zif 1977)

La compression LZ77 remplace des motifs récurrents par des références à leur première apparition. Elle donne de moins bons taux de compression que d'autres algorithmes (PPM, CM), mais a le double avantage d'être rapide et asymétrique (c'est-à-dire que l'algorithme de décompression est différent de celui de compression, ce qui peut être exploité pour avoir un algorithme de compression performant et un algorithme de décompression rapide). LZ77 est notamment la base d'algorithmes répandus comme (ZIP, gzip) ou LZMA (7-Zip) [3].

f. LZ78 (Lempel-Zif 1978)

Elle construit son dictionnaire comme un «arbre de phrases» ou chaînes auquel un caractère spécial est ajouté pour chaque séquence rencontrée en cours de compression. Cette dernière se fonde donc sur le remplacement d'une chaîne de texte longue par une référence chiffrée plus courte. Il faut partir d'un dictionnaire vide puis, pour compacter une séquence, rechercher la phrase la plus longue trouvée dans le dictionnaire, encoder en sortie la référence chiffrée trouvée dans le dictionnaire ainsi que le caractère qui la suit, ajouter au dictionnaire une

nouvelle phrase constituée de la chaîne de caractère vérifiée combinée avec le caractère et continuer jusqu'à la fin du fichier. [3].

j. LZW (Lempel -Zif -Welch)

LZW est basée sur la même méthode, mais Welch a constaté que, en créant un dictionnaire initial de tous les symboles possibles, la compression était améliorée puisque le décompresseur peut recréer le dictionnaire initial et ne doit donc pas le transmettre ni envoyer les premiers symboles. Elle a été brevetée par UNISYS et ne pouvait donc être utilisée librement dans tous les pays jusqu'à l'expiration du brevet en 2003. Elle sert dans les modems, mais UNISYS s'est engagé à vendre une licence à tout fabricant avant d'être acceptée comme norme de compression internationale pour les modems. La compression Lempel-Ziv-Welch est dite de type dictionnaire. Elle est basée sur le fait que des motifs se retrouvent plus souvent que d'autres et qu'on peut donc les remplacer par un index dans un dictionnaire. Le dictionnaire est construit dynamiquement d'après les motifs rencontrés.

h. LZSS (Lempel-Zif Storer Szymanski)

La méthode LZSS (SS correspond aux inventeurs Storer et Szymanski) est une amélioration sensible, elle simplifie les enregistrements à écrire en sortie en reportant soit une double référence composée de l'adresse et de la longueur de la chaîne, soit un simple caractère. Pour distinguer le type de l'enregistrement, il lui suffit de préfixer une référence par un bit à 1 et un caractère par un bit à 0.

I.4 Compression presque sans pertes

Les méthodes de compression sans perte significative sont un sous-ensemble des méthodes de compression avec pertes, parfois distinguées de ces dernières. La compression sans perte significative peut être vue comme un intermédiaire entre la compression conservative et la compression non conservative, dans le sens où elle permet de conserver toute la signification des données d'origine, tout en éliminant une partie de leur information. Dans le domaine de la compression d'image, la distinction est faite entre la compression sans perte (parfaite au bit près ou bit-perfect) et la compression sans perte significative (parfaite au pixel près ou pixel-perfect).

Une image compressée presque sans perte (à ne pas confondre avec une image compressée avec peu de pertes) peut être décompressée pour obtenir les pixels de sa version non-compressée à l'identique. Elle ne peut en revanche pas être décompressée pour obtenir sa version non compressée intégralement à l'identique (les métadonnées peuvent être différentes).

I.5 La Norme JPEG

"Joint Photographic Expert Group", voté comme norme internationale en 1992 [12]. Travail avec la couleur et niveaux de gris, par exemple, par satellite, médical,... JPEG est une méthode de compression sophistiquée avec ou sans perte pour les images en niveaux de gris et en couleur. Elle ne gère pas bien la compression monochromatique. Elle marche aussi très bien pour des images en tons continus.

Un avantage du JPEG est qu'il utilise beaucoup de paramètres[13], laissant l'utilisateur ajuster le nombre de données perdues (et donc le niveau de compression). L'encodeur JPEG produit en sortie un fichier compressé qui inclut les paramètres, les marqueurs et les données compressées. Les paramètres sont codés sur 4 bits. Les marqueurs servent à identifier les différentes parties du fichier. Les données compressées sont combinées dans les MCU (Minimal Coded Unit), où un MCU est une simple donnée ou trois données provenant des trois composantes de l'image. Entre les marqueurs, l'image est organisée en cadres. En mode hiérarchique, il y a beaucoup de cadres et, dans tous les autres modes, il n'y a qu'un seul un cadre. Il y a deux modes principaux : le mode avec perte et le mode sans perte. La plupart des implémentations ne supporte que le mode avec perte. Ce mode inclut un codage progressif et hiérarchique. Les principaux avantages de la compression JPEG sont :

- Haut niveau de compression, spécialement dans les cas où la qualité de l'image est jugée de très bonne à excellente.
- Utilisation de beaucoup de paramètres, ce qui permet aux utilisateurs avancés d'expérimenter la compression/qualité désirée.
- Obtenir de bons résultats avec n'importe quelle image en tons continus, quelque soit la dimension, la couleur, la résolution ou autres caractéristiques de l'image.
- Une méthode de compression sophistiquée mais pas trop complexe, ce qui autorise des implémentations.

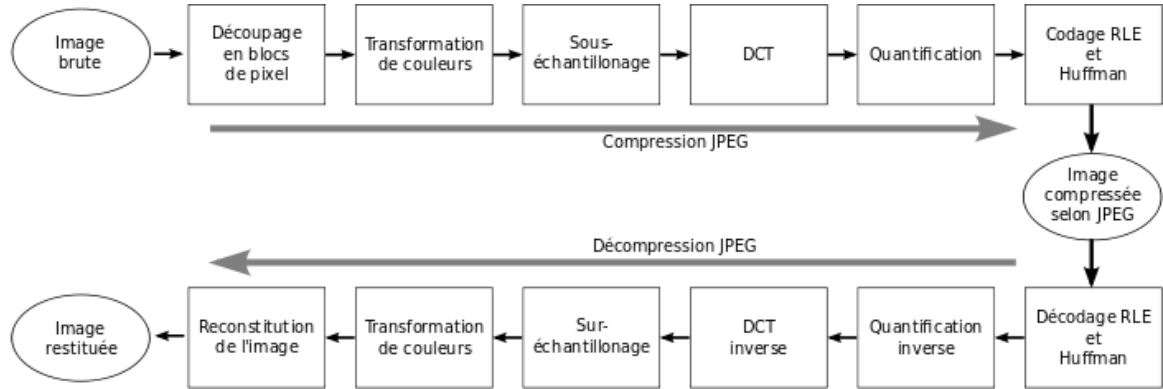


Figure 1.2: La compression JPEG

1.6 Le Mode sans perte :

En plus d’être responsable de la création d’un des formats de compression d’images avec pertes les plus populaires, le JPEG a aussi mis au point quelques standards de compression sans perte, dont le Lossless JPEG (JPEG-LS).

Lorsque l’utilisateur décide qu’aucun pixel ne doit être perdu, la méthode sans perte du JPEG utilise la différenciation pour réduire la valeur des pixels avant qu’ils ne soient compressés, cette forme particulière de différenciation est appelée la prédiction. La valeur des quelques proches pixels voisins est soustraite du pixel pour obtenir un petit nombre, qui est ensuite compressé en utilisant l’algorithme de Huffman (cf. figure1.1).

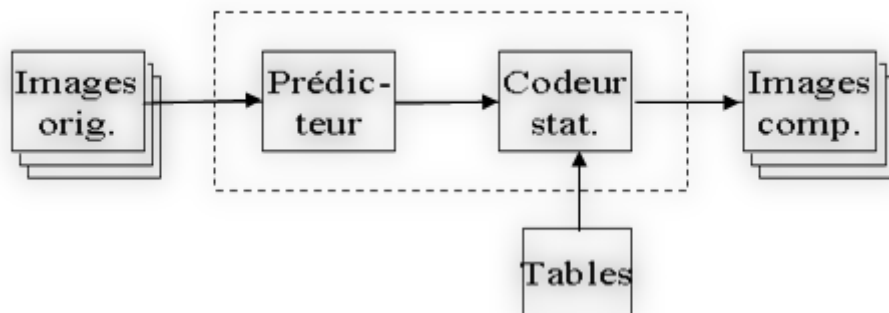


Figure 1.3 : Principe de la Compression JPEG sans pertes

Lossless JPEG a été développé comme un ajout tardif au format JPEG, en 1993, en utilisant une technique complètement différente de la norme JPEG avec perte. Elle utilise un

système de prévision basé sur les plus proches de trois voisins (en haut, à gauche et en haut à gauche), et un codage entropique est utilisé sur l'erreur de prédiction. Lossless JPEG a une certaine popularité dans l'imagerie médicale, et est utilisé dans certains appareils photo numériques pour compresser des images brutes, mais pour le reste n'a jamais été largement adopté [14].

- Un cas particulier de l'effet JPEG où il n'y a pas de perte.

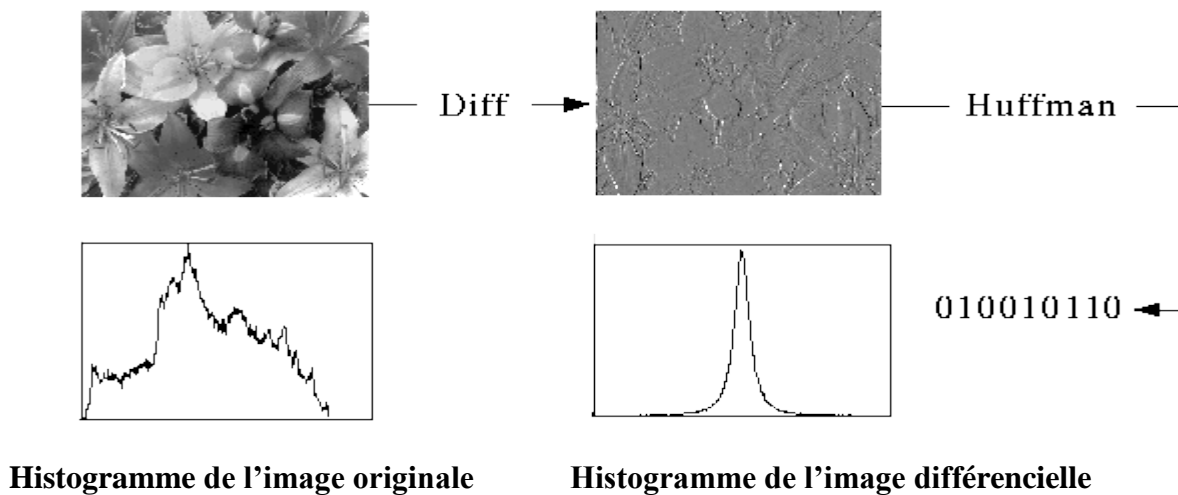


Figure1.4 : JPEG sans perte

1.6.1 Mode de fonctionnement

Contrairement au mode avec perte qui est basée sur la DCT, le procédé de codage sans perte emploie un modèle simple de codage prédictif appelée « modulation par impulsions codées différentielle » (DPCM). Il s'agit d'un modèle dans lequel les prédictions des valeurs d'échantillon sont estimées à partir des échantillons voisins qui sont déjà codés dans l'image. La plupart des prédicteurs prendre la moyenne des échantillons immédiatement au-dessus et à gauche de l'échantillon cible. DPCM code la différence entre les échantillons prédits, au lieu du codage de chaque échantillon de façon indépendante. Les différences d'un échantillon à l'autre sont souvent proches de zéro. Une fois que tous les échantillons sont prédis, les différences entre les échantillons peuvent être obtenus et un codage entropique sans perte en utilisant un codage de Huffman ou codage arithmétique peut être appliqué.

I.6.2 Le prédicteur JPEG

En 1991, Wallace décrit complètement les procédures utilisées par JPEG [14]. Huit prédicteurs (dont un pour une représentation hiérarchique : pas de prédiction) combinent 1, 2 ou 3 valeurs. La solution la moins coûteuse est de ne choisir qu'un seul prédicteur pour l'ensemble de l'image. En 1995, Memon et Sayood [15] démontrent qu'il vaut mieux sélectionner un prédicteur unique pour un bloc de l'image. L'adaptation se fait en calculant le meilleur prédicteur au sens des moindres carrés. Ainsi, si on opère sur des blocs 6×6 , le coût de codage de l'index du prédicteur est de 3/36 bits par pixel (bpp). Pour savoir si cette méthode est avantageuse, le meilleur prédicteur est choisi pour chaque pixel prédit pour le codage de la deuxième diagonale, sans transmettre l'index du prédicteur (seul le coût de transmission minimal nous intéresse). Malheureusement, cette technique nous donne un débit de 5.71 bpp, ce qui se révèle moins bon que la méthode première. Il n'est donc pas nécessaire de poursuivre dans cette voie. Le Lossless JPEG est un format de compression assez rustique basé sur 8 modèles prédictifs simples [16]:

Num	Intensité pixel	Modèle prédictif
00	$\hat{I}(i, j)$	$I(i, j)$
01	$\hat{I}(i, j)$	$I(i - 1, j)$
02	$\hat{I}(i, j)$	$I(i, j - 1)$
03	$\hat{I}(i, j)$	$I(i - 1, j - 1)$
04	$\hat{I}(i, j)$	$I(i, j - 1) + I(i - 1, j) + I(i - 1, j - 1)$
05	$\hat{I}(i, j)$	$I(i, j - 1) + I(i - 1, j) - 2I(i - 1, j - 1)$
06	$\hat{I}(i, j)$	$I(i - 1, j) + I(i, j - 1) - 2I(i - 1, j - 1)$
07	$\hat{I}(i, j)$	$I(i, j - 1) + I(i - 1, j)$

Tableau (1,1) : Modèles prédictifs

I.7 Inconvénients de l'algorithme JPEG

Bien que JPEG est un format de fichier largement utilisé, en particulier dans le domaine de photos en ligne et celles prises par des caméras numériques, il dispose de certains inconvénients majeurs. La première chute au format JPEG, qui a été mentionné est que JPEG est un format «avec perte» et est donc sujette à de nombreux problèmes. Que les plans NASA d'utiliser le fichier compressé à des fins multiples, telles que l'analyse des images de spécifique anomalies, une perte de données de tout type seront causent les images soient inutilisables pour une analyse correcte.

Un autre problème avec le formatage JPEG est que si les pixels sont liés, comme un grand nombre de pixels sont tous de la même couleur, puis un flou peuvent se produire, ou plus la perte est en cours de prévu à l'origine. Dans certaines nouvelles versions de l'algorithme JPEG ces questions ont été compensées, mais il est encore un problème inhérent.

I.8 Lossless versus Lossy

La forme la plus courante de l'algorithme aujourd'hui, en raison de son utilisation abondante et de facilité, est le "Lossy" algorithme JPEG. Le principal avantage d'utiliser un algorithme sans perte est que l'utilisateur peut faire la conversion entre les formats non compressés, comme bitmap et TIFF, le format et le retour sans aucune donnée d'être jetés. Un algorithme «avec perte» continuerait à perdre la qualité au cours de chaque compression et de décompression.

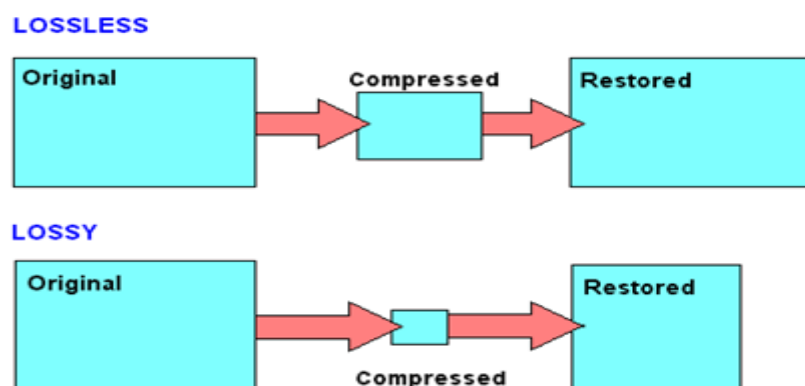


Figure 1.5 : Lossless versus Lossy

1.9 JPEG 2000 sans pertes

JPEG 2000 sans perte comprend un mode spécial sur la base d'un nombre entier d'ondelettes filtre (biorthogonal 3/5). Le mode sans perte JPEG 2000 s'exécute plus lentement et a souvent pires taux de compression que JPEG-LS sur des images artificielles et composé. Le JPEG 2000 réussit mieux que la mise en œuvre de JPEG-LS sur les photos numériques. Il est également évolutif, progressif, et plus largement pris en charge (voir section 2.11).

1.10 Conclusion

Comme de nombreuses méthodes de compression, le JPEG est basé sur des principes mathématiques très compréhensibles. Mais la difficulté intervient lorsque l'on entre en détail dans les démonstrations de l'algorithme. Les théorèmes et principes fondamentaux doivent alors être démontrés, pour expliquer les fondements de la méthode. L'utilisation de telles méthodes de compression des informations est très répandue et utilisées dans de nombreux domaines informatique, téléphonie, vidéo,... etc.

A l'heure actuelle la méthode de compression JPEG est parmi les plus utilisées parce qu'elle atteint des taux de compression très élevés sans que les modifications de l'image ne puissent être décelées par l'œil humain. De plus, beaucoup d'implémentations permettent de choisir la qualité de l'image comprimée grâce à l'utilisation de matrices de quantification paramétrables.

La réputation et donc le nombre d'utilisateurs d'un algorithme de compression dépendent de différents facteurs le rapport taille/qualité, la vitesse de compression et de décompression .Il est donc intéressant de comparer les différentes méthodes, pour pouvoir choisir la plus adaptées à nos besoin et à son utilisation.

Chapitre II

2.1 Introduction

Comme la méthode JPEG n'est pas très efficace pour la compression sans perte, l'ISO, en coopération avec l'IEC, a décidé de développer un nouveau standard simple et rapide pour les images en tons continus. La méthode n'utilise pas DCT, ni de codage arithmétique mais avec une quantification, de façon limitée.

JPEG-LS est basé sur la méthode de compression LOCO-I [17], qui examine beaucoup des voisins déjà vus du pixel courant, les utilise dans le contexte pour prédire le pixel et ensuite calcule une probabilité selon une certaine distribution, cette distribution détermine la prédiction de l'erreur avec un code spécial de Golomb [18].

JPEG-LS permet d'obtenir des taux de compression acceptables à très faible complexité de calcul et de mémoire.

2.2 Algorithme LOCO-I

L'algorithme de compression LOCO-I pour les images fixes est la référence pour la compression sans pertes. En général les méthodes de compression sans pertes 'lossless' sont basées à la fois sur la modélisation du contexte et le codage simple. La modélisation sert à assigner les probabilités optimale et le codage se fait sur la base d'une certaine historique recueilli des pixels précédents.

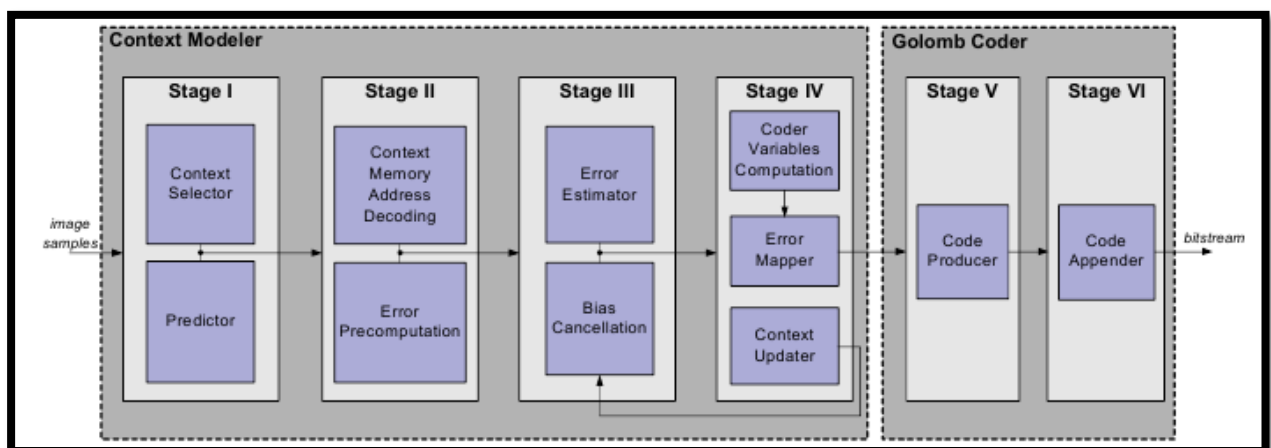


Figure 2.1: Diagramme en bloc pour LOCO-I

2.2.1 Décorrélation

Dans la LOCO-I algorithme, une primitive de détection des contours des bords horizontaux ou verticaux est réalisée en examinant les pixels voisins du pixel courant x comme illustré dans la figure 2.2 .

Le pixel marqué par b est utilisé dans le cas d'un bord vertical tandis que le pixel situé à a est utilisé dans le cas d'un bord horizontal [19].

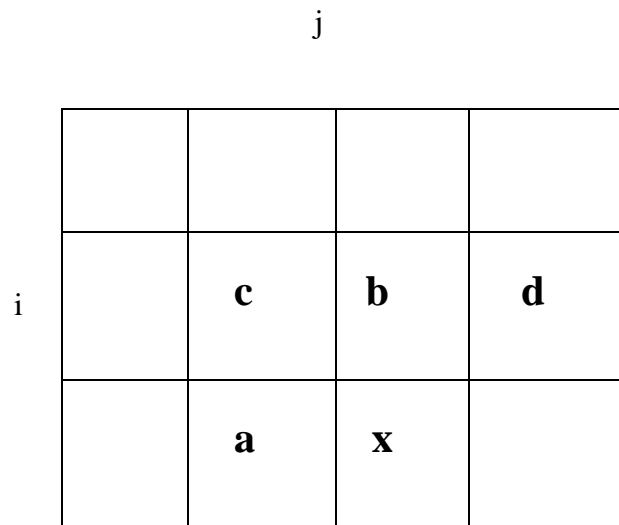


Figure 2.2 : Model causal du JPEG-LS

2.2.2 Le prédicteur médian

Ce prédicteur simple est appelé la détection de bord médian (MED) [20].

Le pixel x est prédit par le prédicteur LOCO-I selon les suppositions suivantes:

$$x_{MED} = \begin{cases} \min(a, b) & \text{if } c \geq \max(a, b) \\ \max(a, b) & \text{if } c \leq \min(a, b) \\ a + b - c & \text{otherwise} \end{cases} \quad (2.1)$$

Il peut utiliser l'un des sept prédicteurs quelconque [21]:

Predictor	Prédiction
1	a
2	b
3	c
4	$a + b - c$
5	$a + (b - c) / 2$
6	$b + (a + c) / 2$
7	$(a + b) / 2$

Tableau 2.1: Utilisation des 7 prédicteurs

Le prédicteur LOCO-I est "causal", il utilise les pixels adjacents qui ont déjà été numérisés, comme il utilise seulement les voisins précédemment codés, le premier pixel I (0, 0) devrait utiliser lui-même. Les autres pixels de la première ligne utilisent toujours P1 (a), lors de la première colonne toujours utiliser P2 (b). Chaque pixel est associé à un contexte sur la base des paramètres calculés localement, dans l'algorithme LOCO-I ya 728 contextes créés [22].

2.3 Description détaillée de l'algorithme JPEG-LS

Le schéma de base de JPEG-LS est donné dans la figure 2.3, nous décrivons brièvement les principales composantes de JPEG-LS.

Dans un balayage de trame, après avoir numérisé les données antérieures, on déduit la valeur du pixel suivant en lui attribuant la distribution de probabilité conditionnelle.

Les unités de modélisation de prédiction en JPEG-LS sont basées sur le modèle causal représenté sur la figure 2.2, où x représente l'échantillon courant, et a , b , c , et d , sont des échantillons voisins dans les positions relatives représentées dans la figure 2.3

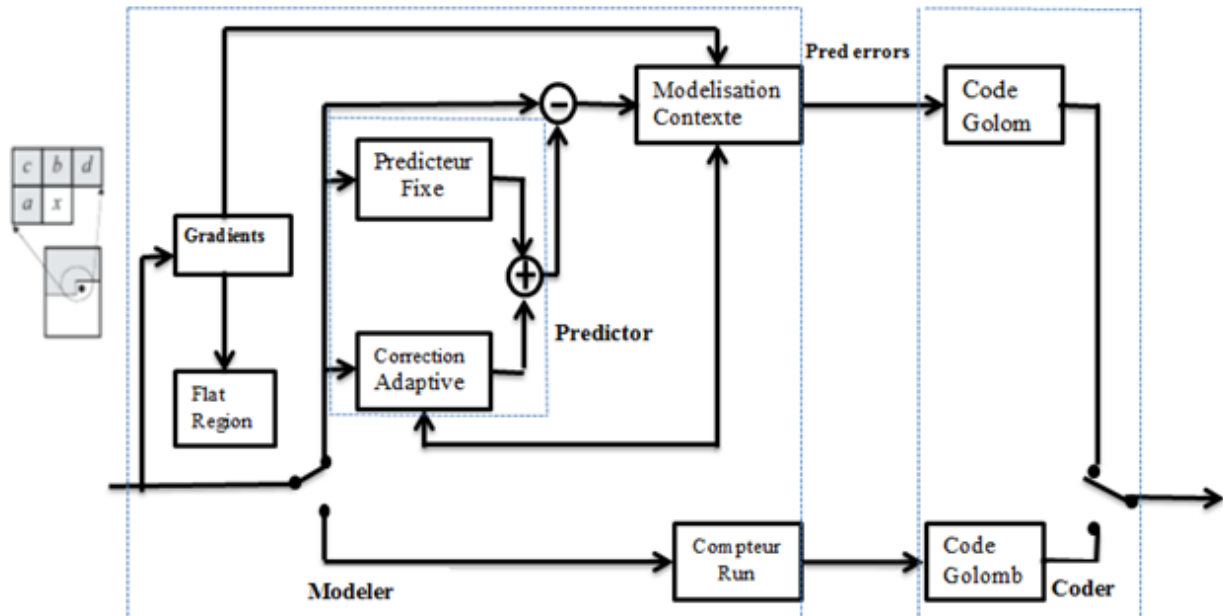


Figure 2.3 : diagramme en bloc de JPEG-LS

2.3.1 Détermination contexte

Une étape de prédiction, dans le quel une valeur déterministe est devinée pour le prochain échantillon basé sur un sous-ensemble fini (Matrice de causalité) de la séquence passé disponible.

JPEG-LS doit alors répondre à la détermination de contexte, et en garantissant un nombre relativement faible, même si suffisamment grand pour capturer les différentes statistiques locales de l'image. Le contexte qui conditionne le codage du courant de prédiction résiduelle en JPEG-LS est construit sur les différences. Ces différences représentent le gradient local, capturant ainsi le niveau d'activité (lissé, nervosité) entourant un échantillon, qui régit la statistique comportement des erreurs de prédiction [23].

Un modèle probabiliste de prédiction résiduel (ou erreur de signal), conditionnée par le contexte de modélisation. Ce modèle détermine la façon dont le résidu est enfoncé. Les unités de prédiction et de modélisation en JPEG-LS sont basés sur le modèle causal représenté sur la

figure 2.2, où x désigne l'échantillon courant, et a, b, c, d sont des échantillons voisins dont les positions relatives illustrées dans la figure 2.2.

2.3.2 Calcul du nombre contexte

Afin de réduire davantage le nombre de contextes, les contextes symétriques sont fusionnés. Le total nombre de contextes devient alors 365 contextes. JPEG-LS fournit les seuils de défaut $T1, T2, T3$ à définir les limites entre les régions de quantification. Celles-ci dépendent la taille de l'alphabet, et peut également être modifié par l'utilisateur. Un choix approprié s'effondre régions de quantification, ce qui entraîne un plus petit nombre effectif de contextes, avec des applications à la compression et de l'estimation d'erreur correct ainsi que les paramètres.

$$N_{\text{contextes}} = ((2T + 1)^3 + 1) / 2|_{T=4} = 365 \quad (2,2)$$

$$((2 \times 4 + 1)^3 + 1) / 2 = 365$$

2.3.3 Gradients locaux

Le gradient local reflète le niveau d'activités telles que la douceur et la nervosité des échantillons voisins. Notez que ces différences sont étroitement liées au comportement statistique des erreurs de prédiction. Chacun des différences constatées dans les équations ci-dessous est ensuite quantifiée dans des régions peu près équiprobables et connectée [24].

$$G_1 = d - b \quad (2.4)$$

$$G_2 = b - c \quad (2.5)$$

$$G_3 = c - a \quad (2.6)$$

Pour JPEG-LS, les différences $G1, G2, G3$ sont quantifiées en 9 régions et sont indexés entre -4 à 4.

Le but de la quantification est de maximiser l'information mutuelle entre la valeur d'échantillonnage actuelle et son contexte tels que les dépendances d'ordre supérieur peuvent être capturés.

On peut obtenir les contextes basés sur l'hypothèse que les contextes après la fusion de signes positifs et négatifs. Une estimation du biais peut être obtenue en divisant les erreurs de

prédiction cumulatifs dans chaque contexte par un nombre d'occurrences du contexte. Dans LOCO-I algorithme, cette procédure est modifiée et améliorée de telle sorte que le nombre d'additions et de soustractions sont réduites [25].

2.3.4 Modélisation du contexte

Le contexte du pixel à coder est souvent utilisé dans le calcul d'optimisation des coefficients du prédicteur. Réduire le nombre de paramètres est un objectif clé de cette modélisation du contexte. Ce nombre est déterminé par le nombre de paramètres libres qui définissent la distribution de codage à chaque contexte et par le nombre de contextes [26]. Le but de la modélisation de contexte, c'est que les structures d'ordre supérieur comme motifs de texture et de l'activité locale de l'image peut être exploitée par modélisation contexte de l'erreur de prédiction. Les contextes sont déterminées en obtenant des différences entre les échantillons voisins qui représente la locale gradient.

C'est pourquoi, certains systèmes sont amenés à mettre en place un prédicteur du second ordre, basé sur la modélisation du contexte. CALIC et LOCO-I proposent ainsi cette solution dans leur schéma de codage.

2.4 Codage différentiel

La méthode de compression DPCM est un membre de la famille des encodages différentiels, qui est lui-même une généralisation du concept d'encodage relatif. Il est basé sur le fait que les pixels voisins d'une image sont en corrélation. Les méthodes d'encodage différentielles calculent les différences $d_i = a_i - a_{i-1}$ entre les données a_i consécutives, et encode les d_i . La première donnée, a_0 , est soit encodée séparément, soit écrite dans le flux compressé dans un format brut [27].

En principe, n'importe quelle méthode, avec ou sans perte, peut être utilisée pour encoder les différences. En pratique, la quantification est souvent utilisée, résultant en une compression avec perte. La quantité encodée n'est pas la différence d_i mais un nombre similaire, quantifié que l'on note par \hat{d}_i . La différence entre d_i et \hat{d}_i est l'erreur de quantification q_i .

Donc $\hat{d}_i = d_i + q_i$ (cf. figure 2.4).

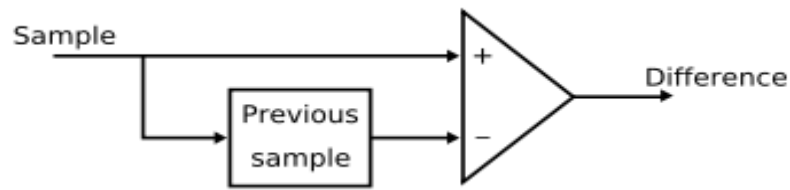


Figure 2.4 : Modèle de codeur DPCM

2.5 Codage prédictif

Un des systèmes les plus simples de compression des images repose sur la prédiction de la valeur d'un pixel en fonction d'un certain nombre de pixels voisins. On peut alors supprimer l'information redondante du pixel et ne coder que l'erreur de prédiction, déterminée par la différence entre la valeur du pixel à coder et sa prédiction. Cette erreur est par la suite encodée. La Modulation par Impulsion et Codage Différentiel est la méthode la plus utilisée dans la compression d'image sans perte et consiste à suivre une ou plusieurs étapes suivantes [28] :

- Parcours de l'image en "raster scan" (ligne par ligne)
- Détermination du contexte du pixel courant - utilisation des pixels avoisinants
- Prédiction localement adaptée du pixel courant
- Boucle de retour de l'erreur permettant la correction de la prédiction.

Ainsi, connaissant l'erreur de prédiction (aussi dénommée prédiction résiduelle) et le système de prédiction, le décodeur peut retrouver la valeur originale du pixel.

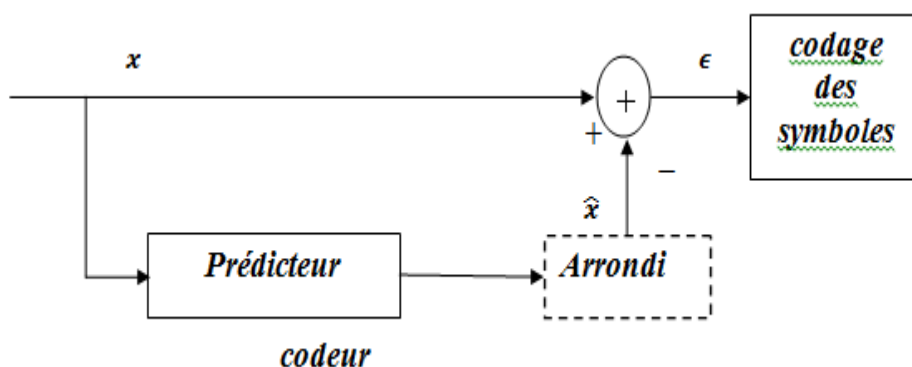


Figure 2.5 : Codage prédictif sans perte

2.6 Calcul du résiduel

Le résiduel du bloc courant, appelé aussi erreur de prédiction, est la différence entre un prédicteur et ce bloc courant. L'expression du résiduel e est donnée dans l'équation suivante :

$$e(x, y) = p(x, y) - \hat{p}(x, y) \quad (2,10)$$

Où : $p(x, y)$ est le pixel du bloc courant à la position (x, y) et \hat{p} est le prédicteur :

L'opération de prédiction inverse est donnée dans l'équation suivante :

$$p(x, y) = \hat{p}(x, y) + e(x, y)$$

Le prédicteur \hat{p} est ajouté au résiduel pour retrouver les pixels $p(x, y)$ du bloc courant. Le décodeur calcule le prédicteur \hat{p} à partir des informations déjà décodées et extraites du train binaire.

2.7 paramétrage

Une connaissance préalable de la structure de compresser les images peuvent être ensuite utilisé en adaptant distributions paramétriques avec peu de paramètres par un contexte aux données. Cette approche permet à un plus grand nombre de contextes pour capturer les dépendances d'ordre supérieur sans pénalité au coût modèle global. Bien qu'il y ait de la place pour des combinaisons créatives, l'observation largement acceptée que des résidus de prédiction en images à tons continus sont bien modélisées par une distribution à deux faces géométrique (TSGD) rend ce modèle très attrayant pour le codage d'images. Il ne nécessite que deux paramètres (représentant le taux de décroissance et le passage de zéro) par le contexte[29] .

Quatre paramètres statistiques sont conservés pour chaque contexte. Il s'agit de l'accumulateur de l'ampleur des erreurs de prédiction précédents $A [Q]$, le biais variable $B [Q]$, qui accumule les valeurs des erreurs, le paramètre de correction d'erreur de prédiction $C [Q]$ et de l'occurrences compteur $N [Q]$.

2.8 Algorithme de codage

Les distributions géométriques des deux côtés (TSGDs) sont utilisées pour aider à décrire le survenance d'erreurs résiduelles de prédiction dans les calculs de probabilité initiales. Il est utilisé avec ce qu'on appelle un codeur Golomb-Rice pour décrire élégamment les distributions d'erreurs.

2.8.1 Le Model TSGD

Il s'agit d'une observation largement acceptée que les statistiques globales de résidus d'une prédiction fixe images à tons continus sont bien modélisé par un TSGD centrée à zéro. Coder dans un cadre de faible complexité, le choix d'un modèle TSGD est d'une importance capitale du fait qu'elle peut être efficacement codé avec une famille élargie de type codes de Golomb, qui sont choisis de manière adaptative [30].

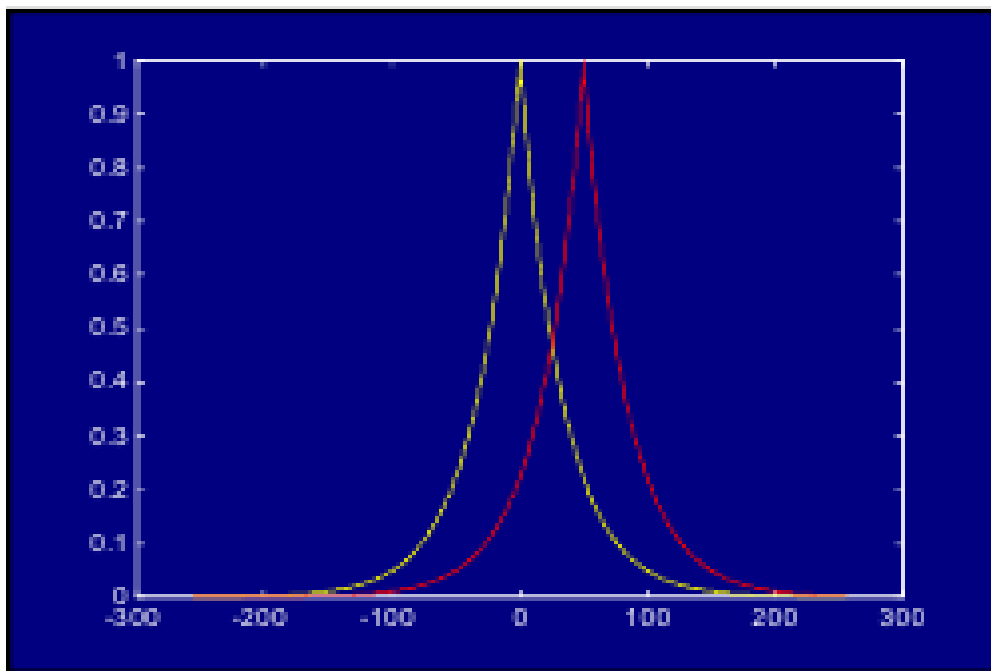


Figure 2.6 : La distrubition géométrique TSGD

2.8.2 Codage des distributions

Il s'agit d'une observation largement acceptée que la distribution des résidus de prédiction en images en tons continus peuvent souvent être estimés par une distribution de Laplace, à savoir une décroissance exponentielle à deux faces centré sur zéro. En principe, un résidu de prédiction peut prendre n'importe quelle valeur dans la gamme $-\alpha \leq x \leq \alpha + 1$, Où $\alpha = 2^{\beta}$ est la taille de l'alphabet d'entrée. En fait, compte tenu de la valeur prédite \hat{X} (connu à la fois l'encodeur et le décodeur), peut prendre une seule valeur différence possible. Une façon d'exploiter cette propriété consiste à prendre la valeur réelle de la prédiction résiduel et le réduire modulo à une valeur $-\alpha / 2$ et $\alpha / 2 - 1$.

2.8.3 Correction adaptative

La partie d'adaptation du prédicteur est basé sur le contexte et il est utilisé pour «annuler» la partie entière du décalage dû au prédicteur fixe (cf. figure 2.2). Par conséquent, le décalage dépendant du contexte dans les distributions est limitée à la plage de $0 < S < 1$. Cela explique comment cette correction adaptative (ou de l'annulation de polarisation) est réalisée à faible complexité.

2.9 Codage de golomb

Le codage de Golomb est un codage entropique inventé par Solomon Wolf Golomb en 1966 [30] et utilisé essentiellement en compression de données. Le code produit est un code préfixe.

2.9.1 Principe

Le codage de Golomb d'un entier naturel N dépend d'un paramètre k et se fait en deux étapes [31]:

- le codage du quotient de la division euclidienne de N par k avec un codage unaire (cf. tableau 2.2).
- le codage du reste r de la même division avec un codage binaire tronqué (voir section 2.8.2).

N (non-négative)	N (strictement)	Code unaire	Alternative
0	1	0	1
1	2	10	01
2	3	110	001
3	4	1110	0001
4	5	11110	00001
5	6	111110	000001
6	7	1111110	0000001
7	8	11111110	00000001
8	9	111111110	000000001
9	10	1111111110	0000000001

Tableau 2.2 : Code unaire

2.9.2 Codage binaire tronqué

Est un codage entropique généralement utilisé pour des distributions de probabilités uniformes avec un alphabet fini. Il est paramétré par un alphabet de taille totale de nombre N . Il s'agit d'une forme un peu plus générale du codage binaire lorsque N n'est pas une puissance de deux. Mathématiquement, pour coder un entier, on code d'abord en unaire, puis en binaire tronqué (cf. tableau 2.3).

Reste (r)	Paramètre Golomb (k)	$\text{Log}_2(k)$	code binaire tronqué
0	2	1	0
1	4	2	01
2	8	3	010
3	16	4	0011

Tableau 2.3 : Code binaire tronqué

Alors on a : $r = N - q \cdot k$ (2,11)

Décimal	Binaire	Code de Golomb k = 1 (unaire)	Code de Golomb k = 2	Code de Golomb k = 3	Code de Golomb k = 4	Code de Golomb k = 5	Code de Golomb k = 10	Code de Golomb k = 16
0	0000	0	0 0	0 0	0 00	0 00	0 000	0 0000
1	0001	10	0 1	0 10	0 01	0 01	0 001	0 0001
2	0010	110	10 0	0 11	0 10	0 10	0 010	0 0010
3	0011	1110	101	100	011	0110	0011	00011
4	0100	11110	1100	1010	1000	0111	0100	00100
5	0101	111110	1101	1011	1001	1000	0101	00101
6	0110	1111110	11100	1100	1010	1001	01100	00110
7	0111	11111110	11101	11010	1011	1010	01101	00111
8	1000	11111110	11110	11011	11000	10110	01110	01000
9	1001	11111110	11111	11100	11001	10111	01111	01001
10	1010	11111110	11110	11100	11010	11000	10000	01010

Tableau 2.4: Codage de Golomb

2.9.3 Optimalité

Le codage de Golomb est adapté pour des données dans les quelles les valeurs les plus faibles sont plus probables que les autres (mais où les autres peuvent malgré tout apparaitre) [31].

Le codage de Golomb est principalement utilisé dans sa variante dite codage de Rice qui peut être implémentée de façon plus efficace. Un codage de Rice est d'ailleurs équivalent à un codage de Golomb dont le paramètre est 2 élevé à la puissance du paramètre de Rice (voir section suivante).

2.9.4 Codage de Rice

Le codage de Rice, codage de Golomb-Rice ou GPO2 (pour *Golomb-power-of-2*) est un codage entropique inventé par Robert F. Rice et James R. Plaunt en 1971 [30] et utilisé essentiellement en compression de données. Le code produit est un code préfixe.

a. Principe

Le codage de Rice d'un entier naturel dépend d'un paramètre et se fait en deux étapes :

- le codage du quotient de la division euclidienne de N par 2^k avec un codage unaire ;
- le codage du reste de la même division avec un codage binaire sur k bits.

Le codage de Rice de paramètre k est strictement équivalent à un codage de Golomb de paramètre 2^k .

Mathématiquement, pour coder un entier $N, N \in \mathbb{N}, N = q \times 2^k + r$, on code d'abord en unaire, puis en binaire.

$$q = \lfloor N/2^k \rfloor \quad (2,12)$$

$$r = N - 2^k \times q \quad (2,13)$$

La division par 2^k peut être implémentée par un décalage de k bits vers la droite, et la seconde étape revient à répliquer les k bits de poids faible de la valeur à coder. Ces opérations simples font que le codage de Rice est particulièrement adapté pour une implémentation rapide.

Valeur	Quotient	Reste	Code
0	0	0	100
1	0	1	101
2	0	2	110
3	0	3	111
4	1	0	0100
5	1	1	0101
6	1	2	0110
7	1	3	0111
8	2	0	00100
9	2	1	00101
10	2	2	00110
11	2	3	00111
12	3	0	000100
13	3	1	000101
14	3	2	000110
15	3	3	000111

Tableau 2.5: Code Golomb-Rice

b. Longueur du code

Le codage de Rice est adapté pour des données dans lesquelles les valeurs les plus faibles sont plus probables que les autres (mais où les autres peuvent malgré tout apparaître). Il est particulièrement apprécié en informatique car son implémentation est simple et rapide.

c. Choix du paramètre

Le choix du paramètre k utilisé lors du codage de Rice détermine le taux de compression qu'il est possible d'obtenir. Le paramètre optimal k_{opt} pour coder V valeurs sur un intervalle de taille I est exprimé par :

$$k_{opt} = - \left[\frac{\log_2\left(2 - \frac{V}{I}\right)}{\log_2\left(1 - \frac{V}{I}\right)} \right] \quad (2,14)$$

2.10 La sélection du mode

Le codeur entre un mode de fonctionnement où une région plate a été détecté dire $a = b = c = d$ ou $[q1, q2, q3] = [0, 0, 0]$. Une fois en mode d'exécution ,le symbole a est attendu et la durée d'exécution est codé.

Une stratégie similaire à celle ci-dessus a été intégré dans le modèle de contexte dans LOCO-I/JPEG-LS. Le codeur entre un mode "RUN" lorsqu'une "zone plate" contexte avec $a = b = c = d$ est détectée. Depuis la région centrale de quantification pour le gradients $G1, G2, G3$ est le singleton $\{0\}$, la condition d'exécution est facilement détectée dans le processus de quantification contexte en vérifiant $[q1, q2, q3] = [0,0,0]$. Une fois en mode course, une course du symbole a est prévu, et la longueur d'exécution (qui peut être zéro) est codé.

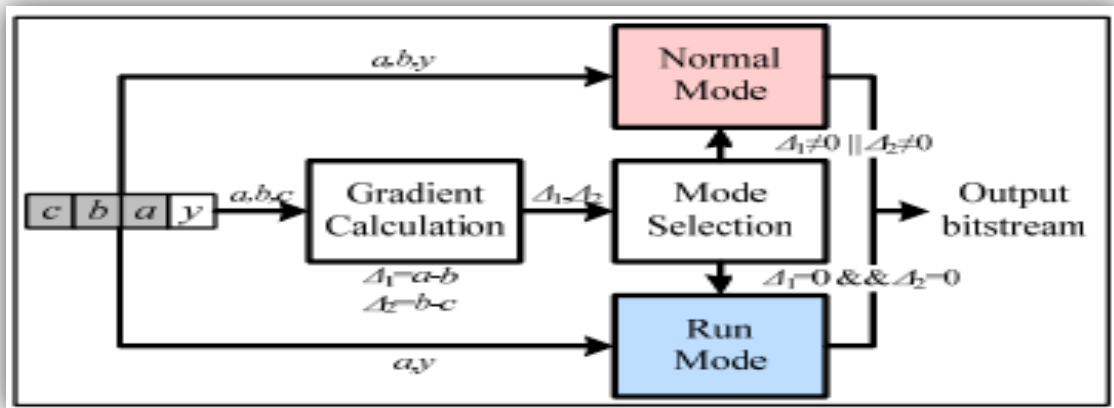


Figure 2.7: sélection du Mode

La volonté de passer en mode "run" se fait lorsque les pixels voisins sont déterminés pour être le même ordre de grandeur. Cela signifie que l'algorithme de prédiction plus ont probablement un résidu de prédiction de zéro, il n'est donc pas nécessaire de poursuivre l'encodage. Les données pour ce domaine sont enregistrées dans l'équation de la probabilité.

2.10.2 Extension d'alphabet

LOCO-I et JPEG-LS, peuvent produire une redondance importante (c.-à-d longueur du code excès par rapport à l'entropie) pour les contextes représentant les régions lisses, dont les distributions sont très culminé en tant que résidu de prédiction de 0 est extrêmement probable. Ce problème est résolu par l'extension alphabet[32]. En codant des blocs de données comme «super-symboles», les distributions ont tendance à être plat (c.-à-d la redondance est réparti sur de nombreux symboles).

2.10.3 L'optimisation des étapes de modélisation

Inspiré des idées de modélisation universel. Dans ce schéma, le contexte de x_{t+1} est déterminée sur des différences $x_{t_i} - x_{t_j}$ où les couples (t_i, t_j) correspondent à des emplacements adjacents dans un modèle de causalité fixe, avec $t_i, t_j \leq t$. Chaque différence quantifiées de façon adaptative basée sur la notion de complexité stochastique, pour atteindre un nombre optimal de contextes [33]. L'étape de prédiction est réalisée avec une adaptative optimisée, en fonction du contexte de fonction de valeurs d'échantillonnage voisines. Les résidus de prédiction, modélisés

par un TSGD, sont codés en arithmétique et la longueur du code qui en résulte est asymptotiquement optimal dans une catégorie générale des processus utilisés pour modéliser les données.

2.11 Le codage RLE

RLE est l'une des méthodes les plus anciennes, les plus simples et la plus utilisée. Tout son secret consiste à identifier et supprimer des redondances d'informations en les codant sous une forme plus compacte. RLE s'emploie à réduire la taille physique d'une répétition de chaîne de caractère. Cette chaîne répétée est appelée un passage (run) et est typiquement codée avec 2 bytes. Le premier byte représente le nombre de caractères dans le passage et est appelé le compteur de passage (run count). Il peut prendre une valeur comprise entre 0h et 128h ou 256h.

Le second byte est la valeur du caractère dans le passage qui peut prendre la valeur 0h à FFh. Ce dernier byte est appelé la valeur du passage (run value).

2.11.1 Le niveau Bit

Le niveau bit s'intéresse à une succession de bits similaires tout en ne tenant absolument pas compte du niveau byte ou du niveau pixel. Seule les images noires et blancs (codée sur 1 bit) contiennent suffisamment de passages de bits pour que cette méthode soit efficace. Un codage typique de RLE au niveau bit code le tout sur un byte ; Soit la valeur du passage sur le MSB (Most Significant Bit) et la valeur du compteur de passage sur les 7 derniers bits. S'il devait s'avérer qu'il y avait des passages de plus de 128 (7 bits) caractères, la chaîne serait simplement coupée en plusieurs paquets[34].

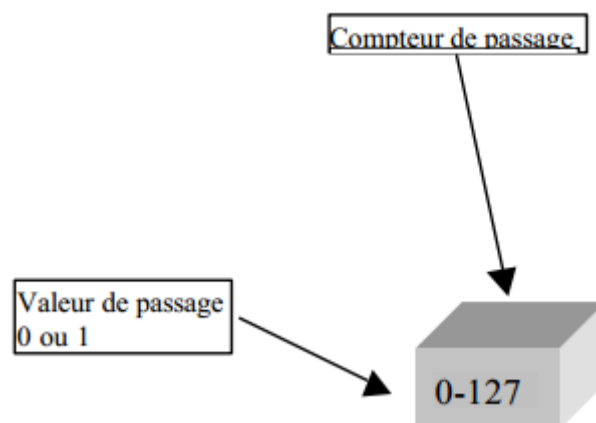


Figure 2.8 : Niveau bit

2.11.2 Codage RLE dans les zones uniformes

Pour éviter d'avoir la longueur du code excès par rapport à l'entropie, on peut utiliser l'extension alphabet blocs de codes de symboles au lieu de coder des symboles individuels ,il évacue l'excès de longueur de codage sur de nombreux symboles. C'est le "run" mode de JPEG-LS et il est exécuté une fois par région contexte plat ou lisse caractérisée par des gradients de zéro est détecté. Une course de l'ouest symbole "a" est attendu et la fin de course se produit lorsque survient un nouveau symbole ou la fin de ligne est atteinte .La course totale de longueur est codée et l'encodeur devrait revenir à la «normale» mode.

2.12 Avantages de l'algorithme JPEG-LS

JPEG-LS a l'avantage de travailler comme un algorithme standard. En outre, contrairement aux algorithmes tels que la norme JPEG avec perte ; Cet algorithme permet la compression sans perte ainsi que d'un mode de pertes contrôlé où une nette supérieure lié à l'erreur d'élévation est choisi par l'utilisateur. Tout cela est réalisé à une complexité de calcul très faible. En plus de ces avantages algorithmiques, nous montrent que JPEG-LS atteint des résultats significativement meilleurs de compression que ceux obtenus avec d'autres algorithmes (non standard) précédemment étudiés pour la compression de données.

Les résultats présentés (voir tableau 2.7 [35]) ici suggèrent que JPEG-LS peut être immédiatement adopté pour la compression de données pour un certain nombre d'applications.

	J2K	JPEG-LS	JPEG	VTC
lossless comp.	+++	++++	<u>+</u> ^a	-
Lossy comp.	+++++	+	+++	++++
progressive	++++	-	<u>+</u> ^b	++
ROI codage	+++	-	-	<u>+</u> ^d
Arbitraire shaped	-	-	-	++
Random accès	++	-	-	-
Faible complexité	++	+++++	+++++	+
Erreur resilience	+++	+	+	+++
Non-itérative	+++	-	-	+
Généricité	+++	+++	++	++

Tableau 2.6 : Comparaison de performances de différents algorithmes

2.13 Conclusion

Le JPEG-LS présente une compression rapide et un taux de compression élevé, semblable au format JPEG 2000. JPEG-LS est plus semblable à l'ancien algorithme JPEG qu'au la norme JPEG2000, mais il donne des résultats intéressantes dans les deux cas de compression sans pertes ou presque sans pertes.

Chapitre III

3.1 Introduction

Dans ce chapitre, nous présentons les résultats de compression obtenus avec la configuration de base de JPEG-LS discuté dans le chapitre 2. Ces résultats sont comparés avec ceux obtenus avec d'autres programmes pertinents rapportés dans la littérature, sur une grande variété d'images.

3.2 Paramètres de validation

En fonction de l'application recherchée, l'algorithme de compression doit pouvoir vérifier un certain nombre de critères de qualité.

3.2.1 Quantité d'information

On définit la notion de quantité d'information (ou incertitude) d'une observation de la variable aléatoire par :

$$I_x = -\log(P(X=x)) \quad (3,1)$$

Avec :

- I_x est l'incertitude (quantité d'information)
- P : probabilité d'un événement

3.2.2 L'entropie

Soit X une variable aléatoire discrète dont les réalisations x prennent leurs valeurs dans X . On définit l'entropie $H(X)$ comme étant une mesure de l'incertitude de X : c'est la quantité moyenne d'information nécessaire à la description de X ; pour une source binaire, elle est représentée comme étant le nombre minimal de bits indispensables à la description de X . Si X suit une densité $p(x)$, alors :

$$H(X) = E_X\{I_X\} = E_X\{-\log(P(X))\} \quad (3,2)$$

Où:

- E est l'espérance mathématique.
- Le log est en base 2 et l'entropie s'exprime en bits.

$$H(S) = -\sum_{k=1}^n p_k \cdot \log(p_k) \quad (3,3)$$

Où :

- S est la source d'information
- P loi de probabilité
- H L'entropie

3.2.3 Taux de compression (RC)

Le taux de compression, souvent utilisé, est l'inverse du quotient de compression, il est habituellement exprimé en pourcentage. Il est mesuré par le rapport entre le volume des données initiales et celui des données codées. Plus le taux de compression est élevée ; plus l'espace nécessaire pour le stockage diminue, ainsi que le temps nécessaire pour la transmission. Il est calculé par la formule suivante :

$$RC(\%) = \frac{\text{nombre de bits codés}}{\text{nombre de bits de l'image originale}} \times 100 \quad (3,4)$$

Dans la pratique, on utilise plutôt le débit pour mesurer le pouvoir de compactage d'une méthode.

Le débit est exprimé en bits par pixel :

$$RC(bpp) = \frac{\text{nombre de bits codés}}{\text{taille de l'image (nombre de pixels)}} \quad (3,5)$$

3.2.4 Temps de calcul

La contrainte du temps est un facteur essentiel dans l'évaluation des performances de toute méthode de compression, elle revient à calculer le temps pris par la compression et la décompression des images. Cette contrainte est plus au moins imposée selon l'application visée par la compression [36] .

3.3.5 Complexité Algorithmique

L'évaluation de la complexité est un problème difficile, sans bien mesure définie. Cela signifie différentes choses pour différentes applications. Il peut s'agir d'une bande passante mémoire, la mémoire de travail totale, nombre de cycles CPU, nombre de portes quincailleri[37].

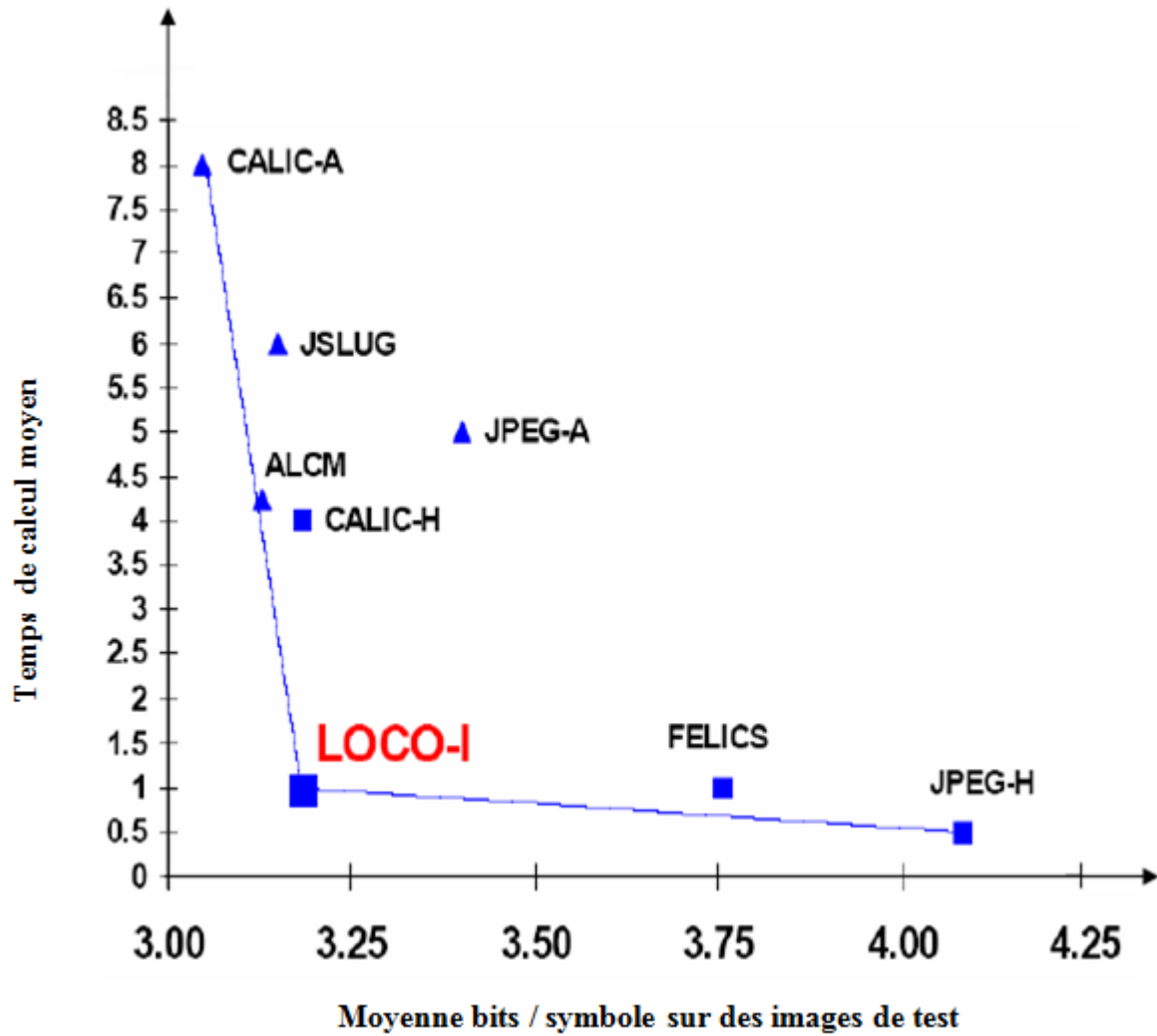


Figure 3.1 :Complexité alorithmique

3.2.6 Fonctionnalités

La plupart des applications nécessitent d'autres fonctionnalités dans un algorithme de codage de simple efficacité de compression. Il s'agit souvent dénommé les fonctionnalités, nous fournissons une matrice de fonctionnalités qui indique l'ensemble d'appui caractéristiques de chaque norme et une appréciation de la façon dont elles sont remplies [38].

3.3 Mise en oeuvre de JPEG-LS

Il a été montré que l'exécution de la JPEG-LS implique plusieurs fonctions et des processus, de la détermination de contexte au codage de l'erreur mappé. Si une tentative de mettre en œuvre l'écoulement de l'algorithme en une seule étape a été faite, un chemin critique de retard inacceptable serait produit [39]. Ce serait de limiter la fréquence maximale de fonctionnement. Ce pendant, la répartition des processus de l'algorithme en plus d'une étape, par pipeline, minimise le chemin critique et réduit considérablement le délai global maximum.

3.4 Organigramme Global de l'algorithme JPEG-LS

On a commencé par la détermination du contexte pour aller au calcul du gradient puis on passe par un test afin de choisir entre l'erreur de prédiction qui sera codé par le codage de golomb par contre le mode run travaille avec le codage de RLE si tout les gradients sont nuls.

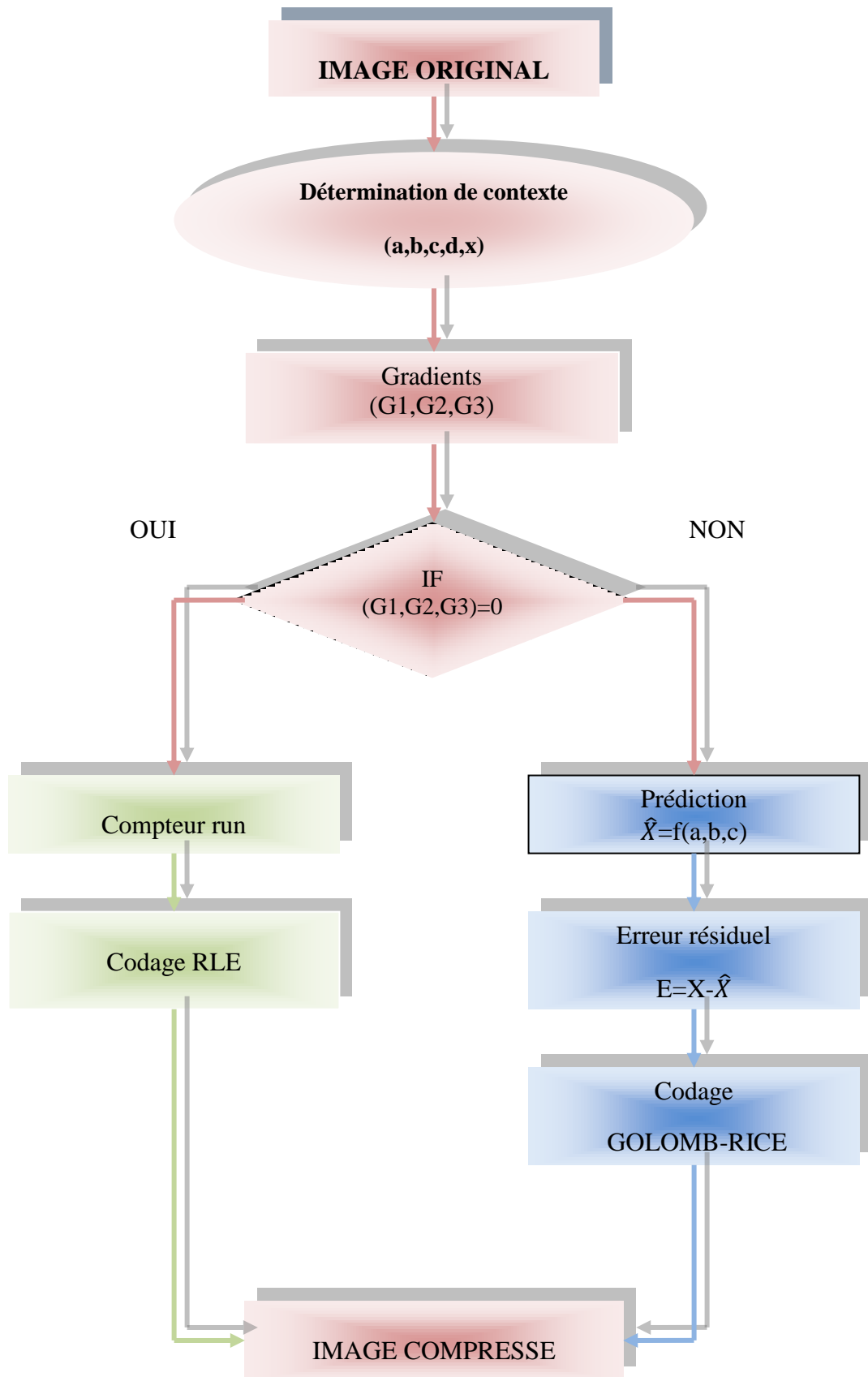


Figure 3.1 : Organigramme global de JPEG-LS

3.4.1 Calcul de l'erreur de prédiction

Pour l'erreur de prédiction sera calculé par la différence entre le pixel courant et le pixel prédit.

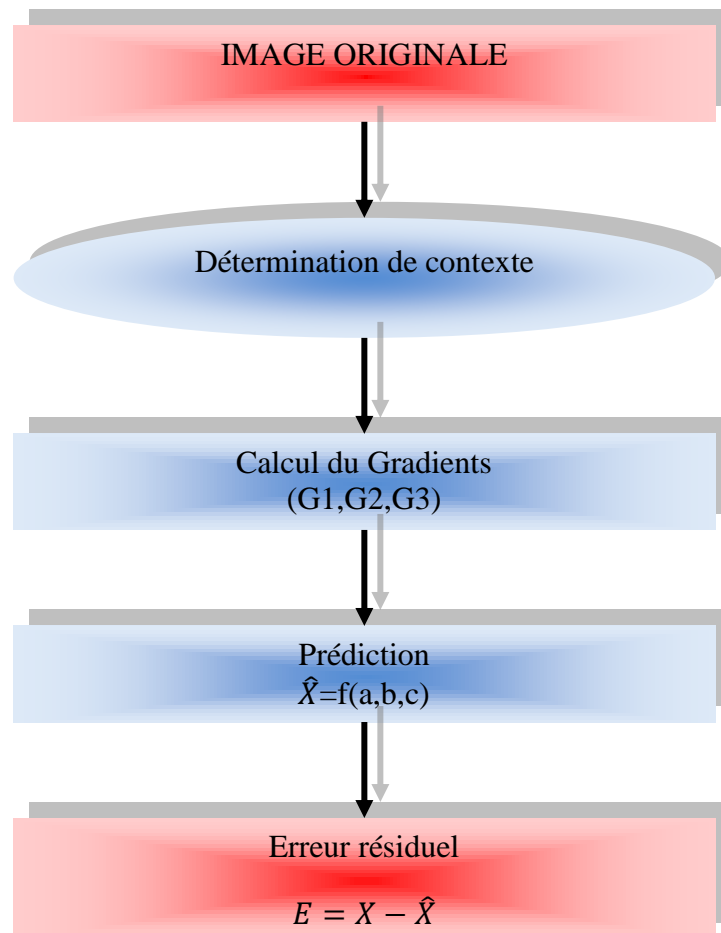


Figure 3.2: Organigramme de calcul de l'erreur de prédiction

3.4.2 Organigramme de codage de golomb

Le codage de golomb se fait en deux partie ,c'est de coder le quotient et le reste séparément mais pour le reste on le code avec le binaire tronqué

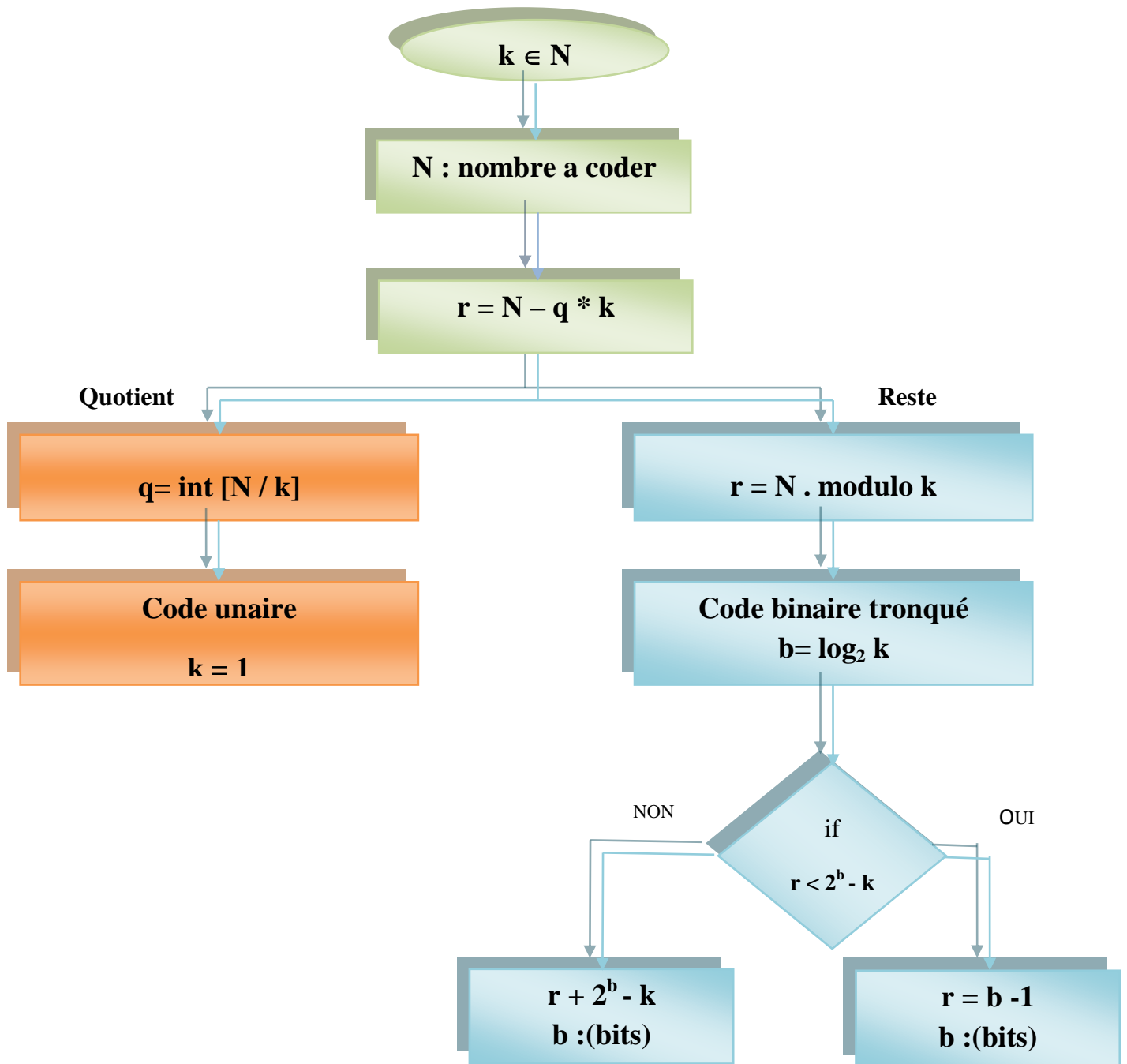


Figure 3.3 : Organigramme de codage de golomb

3.4.3 Mode run

Lorsque le locale gradient est nul on a un compteur run qui calcul la redondance des pixels qui seront réduit par le codage RLE

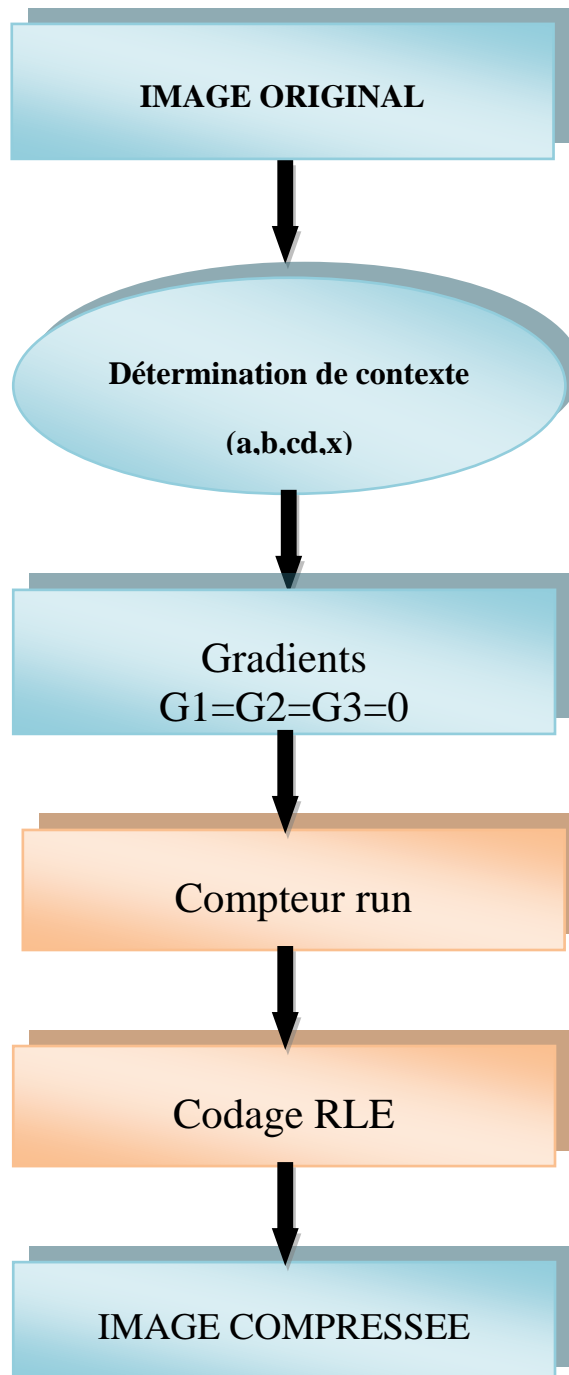


Figure 3.4 : Organigramme du mode run

3.5 Résultats obtenus :



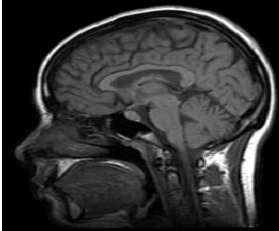



Images	Nom	JPEG-LS	JPEG 2000
	Women	4,43 bpp	4,80 bpp
	Lena	4,43 bpp	4,75 bpp
	Mri	3,32 bpp	3,54 bpp
	Fruits	4,40 bpp	4,30 bpp
	Airplane	4.48 bpp	4.54 bpp
	Cameraman	4,52 bpp	4,58 bpp

Tableau 3.1: Résultats et comparaison avec différents algorithmes

3.6 Discussion

Les résultats obtenus par l'algorithme JPEG-LS sont meilleurs dans toutes les images de test utilisés. La variation du taux de compression dépend d'une image à une autre, et varie entre 0.06 et 0.37 bpp. Cette variation est due à la variation des caractéristiques des images.

3.7 Conclusion

Nous avons présenté un algorithme de compression sans pertes. Les taux de compression sont généralement meilleurs que le standard JPEG-2000 pour toutes les images de test utilisés.

JPEG-LS représente un algorithme de compression beaucoup moins complexe que l'algorithme standard JPEG. Cependant, d'un point de vue fonctionnalité, JPEG 2000 représente une véritable amélioration, en fournissant un flux binaires progressif et analysable.

Conclusion générale

Conclusion générale

Il est important de comprendre que toute compression apportée à une image n'implique pas forcément une diminution de la qualité d'une image. En effet, dans le cas d'une compression sans pertes, aucune information n'a subi de détérioration mais présente simplement une réorganisation des données informatiques composant une image. A la différence d'une compression d'image sans pertes, la compression avec pertes ou destructive, réglera le sort de bon nombre d'informations jugées inutiles car non perceptible par un œil humain.

Parmi les méthodes de compression sans pertes disponibles, JPEG-LS a d'excellentes performances de codage. Cependant, il ne comprime une seule image avec intra codage et n'utilise pas la corrélation entre trames entre images.

Bien JPEG-LS n'est pas un nouvel algorithme, il est encore un moyen très efficace et une norme de compression utile.

JPEG-LS se distingue comme la meilleure option lorsque seulement la compression sans perte est d'un intérêt, fournissant la meilleure efficacité de la compression à une faible complexité. D'autre part JPEG 2000 est la solution la plus flexible, si la complexité ajoutée est acceptable.

Perspectives

Les sources d'image couramment utilisés (Scanners, appareils photo numériques, etc) fournis une représentation à base de pixels, qui est maintenue par les algorithmes de compression. Cependant, il ya un nombre croissant d'ordinateurs génèrent des images qui ne sont pas à l'origine à base de pixels, mais on a converti à une telle représentation avant compression. De nouveaux algorithmes de compression pourraient travailler sur la nature spécifique de ces images et de garder une représentation fondée sur d'autres constructions mathématiques. Alors que la compression des images à base de pixels semble atteindre une limite, cette nouvelle approche pourrait ouvrir la porte à une augmentation de l'efficacité de la compression des images générées par ordinateur tout en ayant l'avantage d'être indépendant de la résolution.

Des travaux sont envisageables pour l'amélioration des divers éléments de la chaîne de traitements, pour l'extension volumique de l'approche, ainsi que pour l'optimisation des prédicteurs qui permettrait d'obtenir de meilleurs taux de compression.

Bibliographies

Bibliographies

[1]:Istok Kespret - *Compresser vos données avec LHA, PKZIP, ARJ ...* - (éd. Micro Application, coll. "Le livre de", 1994) - 348 p. - (ISBN 2-7429-0172-8)

[2]:Jean-Paul Guillois - *Techniques de compression des images* - (éd. Hermès - Lavoisier, coll. "IC2", 1996) - 252 p. - (ISBN 2-86601-536-3)

[3]: Pennebaker, WB & Mitchell, JL (1993). *JPEG standard de compression de données d'image* . New York:. Van Nostrand Reinhold ISBN 0-442-01272-1 .

[4]: UIT-T. ISO DIS 10918-1 Compression numérique et codage des images en tons continus (JPEG). Recommandation T.81.

[5]:S. Dewitte and J. Cornelis. Lossless integer wavelet transform. *SPLetters*, 4(6) :158–160, Juin 1997.

[6]:A. Zandi, J. D. Allen, E. L. Schwartz, and Martin P. Boliek. CREW : Compression with reversible embedded wavelets. In *Data Compression Conference*, pages 212–221, 1995

[7]: <https://code.google.com/p/zopfli/> [archive]

[8] : Douglas A. Kerr, *Chrominance Subsampling in Digital Images [archive]* 2009

[9] :http://fr.wikipedia.org/w/index.php?title=Prédiction_par_reconnaissance_partielle&oldid=89598072 Catégories :

[10] :http://fr.wikipedia.org/w/index.php?title=Pondération_de_contextes&oldid=89597962 ». Catégories :

[11]:Rissanen, Jorma . (mai 1976) "généralisé Kraft inégalité et le codage arithmétique" (PDF). *IBM Journal de la Recherche et Développement* **20** (3):. 198-203 doi : 10.1147/rd.203.0198 . <http://domino.watson.ibm.com/tchjr/journalindex.nsf/4ac37cf0bdc4dd6a85256547004d47e1/53fec2e5af172a3185256bfa0067f7a0?OpenDocument>. Récupérée 21/09/2007

Bibliographies

[12] : Pennebaker, WB & Mitchell, JL (1993). *JPEG standard de compression de données d'image* . New York:. Van Nostrand Reinhold ISBN 0-442-01272-1 .

[13]: G.K. Wallace. The jpeg still picture compression standard. CACM, 34(4) :30–44, April 1991.

[14] : N. D. Memon and K. Sayood. Lossless image compression : a comparative study. in Proc. SPIE (Still-Image Compression), 2418(8–20), Feb. 1995.

[15]: Olivier Godin feat. Sylvain Bérubé, Compression basée sur le contexte ,Université de Sherbrooke,15 février 2011

[16]: A.K. Jain. Advances in mathematical models for images processing. in Proc. IEEE, 69(5) :502–528, 1981.

[17]: W.S. Lee. Edge adaptive prediction for lossless image coding. in Proc. IEEE Data compression conference, Snowbird :483–490, 1999.

[18]: M.J. Weinberger, G. Seroussi, and G. Sapiro. Loco-i : a low complexity, context- based, lossless image compression algorithm. roc. Of the IEEE Data Compression Conference, pages 141–150, 1996.

[19]: X. Wu. Lossless compression of continuous-tone images, via context selection, quantization and modelling. IEEE Trans. on Image Processing, 6(5) :656–664, 1996.

[20]: Matsuda, H. Mori, and S. Itoh. Lossless coding of still images using minimum- rate predictors. Proc. IEEE ICIP, 2001.

[21]: S. Dewitte and J. Cornelis. Lossless integer wavelet transform. SPLetters, 4(6) :158–160, Juin 1997.

[22]: A. Zandi, J. D. Allen, E. L. Schwartz, and Martin P. Boliek. CREW : Compression with reversible embedded wavelets. In Data Compression Conference, pages 212–221, 1995

Bibliographies

- [23]: W. Philips. The lossless dct for combined lossy/lossless image coding. In Proc. International Conference on Image Processing, volume 3. ICIP'98, Oct. 1998.
- [24]: K. Komatsu and K. Sezaki. Reversible discrete cosine transform. In Proc. IEEE. ICSASSP'98, Mai 1998.
- [25]: K. Komatsu and K. Sezaki. 2d lossless
- [26]: G.K. Wallace. The jpeg still picture compression standard. CACM, 34(4) :30–44, April 1991.
- [27]: N. D. Memon and K. Sayood. Lossless image compression : a comparative study. in Proc. SPIE (Still-Image Compression), 2418(8–20), Feb. 1995.
- [28]: Professeur : M. Tri An Banh, Rapport de synthèse sur la compression d'image pour le cours INFO006-0 : Structure de l'information, Campisi Anthony ,18 avril 2008
- [29]: Memon, Nasir D.; Wu, Xiaolin; Sippy, V. & Miller, G. (1997). «Extension Interband de codage de la nouvelle norme JPEG sans perte". *Proc. Int SPIE. Soc. Opt. . Eng* **3024** (47):. 47-58
doi : [10.1117/12.263270](https://doi.org/10.1117/12.263270)
- [30]: M. J. Weinberger, G. Seroussi, and G. Sapiro, “Effects of resets and number of contexts on the baseline.”ISO/IEC JTC1/SC29/WG1 document N386, June 1996.
- [31]: M. J. Weinberger, J. Rissanen, and R. B. Arps, “Applications of universal context modeling to lossless compression of gray-scale images,” IEEE Trans. Image Processing, vol. 5, pp. 575–586, Apr. 1996.
- [32]: M. J. Weinberger and G. Seroussi, “Sequential prediction and ranking in universal context modeling and data compression,” IEEE Trans. Inform. Theory, vol. 43, pp. 1697–1706, Sept. 1997. Preliminary version presented at the 1994 IEEE Intern'l Symp. on Inform. Theory, Trondheim, Norway, July 1994.
- [33]: MM. S.Maadi, Y. Peneveyre, etC. Lambercy, étudiants en télécommunications de dernière année

Bibliographies

[34]: Solomon Wolf Golomb, « Run-length encodings », *IEEE Transactions on Information Theory* IT-12, pp. 399-401, 1966.

[35]: T. Seemann, P.E. Tisher, and B.Meyer. History-based blending of image sub- predictors. Proc. International Picture Coding Symposium PCS97, 1997.

[36]: M.J. Weinberger, G. Seroussi, and G. Sapiro. Loco-i : a low complexity, context- based, lossless image compression algorithm. roc. Of the IEEE Data Compression Conference, pages 141–150, 1996.

[37]: http://en.wikipedia.org/wiki/Truncated_binary_encoding

[38]: ISO/IEC, ISO/IEC 14496-2:1999: Information technology — Coding of audio-visual objects — Part 2: Visual, December 1999.

[39]: ISO/IEC JTC 1/SC 29/WG 1, ISO/IEC FCD 15444-1: Information technology — JPEG 2000 image coding system: Core coding system, WG 1 N 1646, March 2000. <http://www.jpeg.org/FCD15444-1.htm>.