UNIVERSITÉ DE
SHERBROOKE
Faculté de génie
Génie électrique et génie informatique

Modelage de contexte simplifié pour la compression basée sur la transformée en cosinus discrète

Mémoire de maîtrise ès sciences appliquées
Spécialité: génie électrique

Jean-Sébastien AUCLAIR BEAUDRY

Jury: Dominique Drouin
Ruben Gonzales-Rubio
Chon Tàm Le Dinh
François Michaud

Sherbrooke (Québec) Canada Septembre 2009

RÉSUMÉ

Le manque grandissant de médecins spécialistes à l'extérieur des grands centres influe négativement sur la qualité des soins reçus par les patients. Une solution possible à ce problème est la supervision des médecins généralistes en région par des spécialistes disponibles dans les grands centres. Cette supervision à distance nécessite le développement de technologies répondant aux besoins précis de celle-ci.

Dans le cadre de ce projet de recherche, la transmission de l'image est considérée. En vue de développer un codec vidéo adéquat pour l'application dans le futur, le codec intra-image est étudié. Plus précisément, le but recherché est de simplifier et de rendre parallélisable le codec AGU[1] [Ponomarenko et coll., 2005] sans en réduire les performances en deça des performances de JPEG2000 [Skodras et coll., 2001]. Ces améliorations facilitent la réalisation matérielle du codec en réduisant la latence si critique aux applications de télésupervision.

Pour accomplir ces objectifs, le modelage du contexte du codec AGU doit être modifié. La méthodologie proposée passe par l'implémentation du codec AGU, l'étude de la source de données et la modification du modelage de contexte. La modification en question est le remplacement de l'utilisation d'une méthode adaptative basée sur un arbre de conditions par un réseau de neurones.

Au terme de cette recherche, le réseau de neurones utilisé comme modeleur de contexte s'avère être un succès. Une structure à neuf entrées et aucune couche cachée est utilisée et permet de rendre presque triviale l'opération de modelage du contexte en gardant des performances supérieures à JPEG2000 en moyenne. La performance est inférieure à JPEG2000 pour une seule image de test sur cinq.

Dans le futur, il est possible d'étudier comment améliorer davantage ce codec intra-image à travers l'utilisation d'un meilleur réseau de neurones ou d'une transformée différente. Il est également souhaitable d'étudier comment faire évoluer le codec en un codec inter-image.

Mots-Clefs Intelligence artificielle, traitement des signaux numériques, traitement d'image, compression de données, codage arithmétique.

[1] Aucune signification publiée.

REMERCIEMENTS

Il est important pour moi de souligner le soutien de mes aidants les plus importants. Cette liste débute par mes codirecteurs Chon Tàm Le Dinh et François Michaud. Leurs conseils et leurs encouragements ont permis à ce projet d'être un succès.

J'aimerais aussi mentionner Dominic Létourneau toujours prêt à répondre à mes nombreuses questions avec une bonne humeur irréprochable. D'autres membres du LABORIUS m'ont également aidés à de nombreuses reprises et ce fut très apprécié.

Je tient absolument à remercier la communauté Lisp pour leur travail et pour l'aide personnalisée que j'ai reçue. Une mention spéciale pour Xach «Xach» Beane, Stas «stassats» Boukarev, Alastair «nyef» Bridgewater, Stelian «fe[nl]ix» Ionescu, Paul «pkhuong» Khuong et Kevin «kpreid» Reid.

Finalement, j'aimerais remercier le Conseil de Recherches en Sciences Naturelles et en Génie ainsi que la Fondation de l'Université de Sherbrooke pour leur soutien financier.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LEXIQUE

Codec COdeur et DECodeur. Ensemble des algorithmes responsables de traiter l'information pour en réduire la taille et pour retrouver l'information originale après la transmission.

Inter-image Qui utilise l'information de plusieurs images. Habituellement utilisé pour qualifier un codage ou un codec.

Intra-image Qui considère l'information d'une seule image. Habituellement utilisé pour qualifier un codage ou un codec.

Latence Typiquement, le temps entre l'entrée et la sortie d'un signal dans un système.

Pré-traitement Dans le contexte d'un codec, traitement qui est effectué avant l'encodage.

Post-traitement Dans le contexte d'un codec, traitement qui est effectué après le décodage.

Surdébit *Overhead*. Données ne faisant pas partie du message, mais devant être transmises pour l'acheminement du message ou son traitement.

Télécoaching Voir télésupervision.

Téléprésence Transmission des perceptions sensorielles, principalement visuelles et tactiles, d'un manipulateur à distance à un opérateur humain, lui donnant ainsi l'impression d'être sur place.

Télésupervision Supervision effectuée à distance. Syn. : Télécoaching.

Temps différé, en Par opposition à *en temps réel*. Mode de traitement où les données sont d'abord enregistrées puis traitées.

Temps réel, en Mode de traitement selon lequel les données sont traitées immédiatement après leur acquisition et selon lequel le temps de réponse qui sépare l'entrée des données de l'émission des résultats est réduit au minimum.

LISTE DES ACRONYMES

BAC *Binary Arithmetic Coding.* Codage arithmétique binaire. Voir section 2.3.3.

bpp Bits par pixel. Unité utilisée dans la mesure de la compression d'images.

BTC *Block Truncation Coding.* Voir section 2.2.3.

CABAC *Context-based Adaptative Binary Arithmetic Coding.* Codage arithmétique binaire adapté au contexte. Voir section 2.3.3.

DCT *Discrete Cosine Transform.* Transformée en cosinus discrète. Voir section 2.1.2.

DWT *Discrete Wavelet Transform.* Transformée par ondelettes discrète. Voir section 2.1.3.

FFNN *FeedForward Neural Network. Réseau de neurones à réaction positive.*

GLBT *Generalized Lapped Biorthogonal Transform.* Transformée biortogonale à recouvrement généralisée. Voir LOT.

GLOT *Generalized Lapped Orthogonal Transform.* Transformée orthogonale à recouvrement généralisée. Voir LOT.

LABORIUS Laboratoire de robotique mobile et de systèmes intelligents de l'Université de Sherbrooke.

LOT *Lapped Orthogonal Transform.* Transformée orthogonale à recouvrement. Voir aussi GLOT et GLBT. Voir section 2.1.4.

PSNR *Peak Signal to Noise Ratio.* Mesure de distorsion d'une image reconstruite par rapport à l'originale.

SQ *Scalar Quantization.* Quantification Scalaire. Voir section 2.2.1.

TCQ *Trellis Coded Quantization.* Quantification codée en treillis.

TCSFQ *Trellis Coded Space-Frequency Quantization.* Quantification espace-fréquence codée en treillis.

VOIP *Voice Over Internet Protocol.* Transmission de la voix par Internet (protocole IP).

YUV Format d'image couleur ou les trois canaux Y, U et V correspondent respectivement à la luminance, la chrominance reliée au bleu et la chrominance reliée au rouge.

CHAPITRE 1

INTRODUCTION

Les défis à surmonter par le système de santé canadien sont de plus en plus grands. D'une part, le vieillissement de la population fera passer la proportion de gens de 65 ans et plus de 13 % en 2004 à 24 % en 2029 [PRECARN, 2005]. D'autre part, les médecins spécialistes se font de plus en plus rares, situation qui est d'ailleurs accrue lorsqu'on s'éloigne des grands centres. Cette conjoncture force des médecins généralistes à soigner des patients qui auraient avantage à recevoir les soins d'un spécialiste. Cela a pour conséquence de réduire la qualité des soins reçus par les patients.

Une solution très prometteuse à ce problème est la supervision de médecins généralistes en région par les spécialistes des grand centres. En effet, en permettant à un spécialiste de suivre et de conseiller un médecin de moindre expérience (pour un type d'opération), il est possible d'améliorer la qualité des soins reçus par les patients [Lemieux et coll., 2007]. Grâce aux moyens de communication modernes, cette supervision peut être effectuée à distance : il s'agit alors de télésupervision.

Pour maximiser la qualité de la télésupervision, et donc la qualité des soins, il est souhaitable de reproduire les éléments essentiels d'une supervision classique :

1. Le superviseur doit observer ce qu'il désire (vision) ;
2. Le superviseur doit entendre ce qui se passe dans la salle d'opération (audition) ;
3. Le superviseur doit avoir accès aux informations vitales du patient ;
4. Le superviseur doit communiquer oralement avec le supervisé.

Ces éléments doivent être naturels, faciles d'utilisation et sans délais bien sûr étant donné la condition parfois critique du patient. De cette façon, le superviseur a accès à l'information qu'il désire comme s'il était présent dans la salle d'opération.

1.1 Situation actuelle

Des systèmes de télésupervision sont déjà utilisés dans plusieurs branches de la médecine [Lemieux et coll., 2007] (par exemple, la radiologie et la psychiatrie). La méde-

cine d'urgence reste toutefois un territoire à conquérir. Les travaux de LEMIEUX et coll. [LEMIEUX et coll., 2007] traitent précisement de cette nouvelle avenue. Ces travaux s'intéressent plus précisément à la vision du superviseur. En simplifiant le système de vision en télésupervision, il est possible de le schématiser tel qu'à la figure 1.1. La contribution de LEMIEUX et coll. est bien identifiée et correspond principalement à la configuration mécanique du système portant les caméras qui capteront les images vues par le télésuperviseur. Grâce à leurs systèmes, les caméras peuvent être déplacées et orientées à la guise de celui-ci. Leurs systèmes ont été complétés et ont subi une première phase de tests.

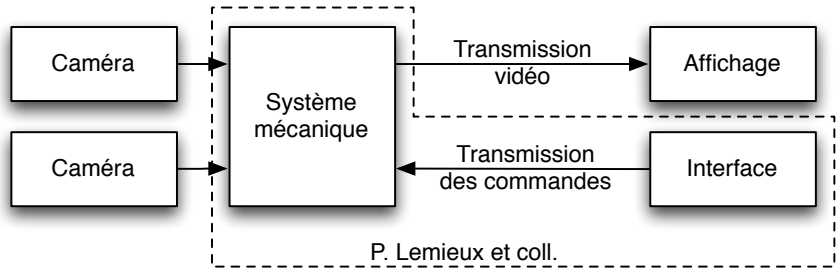


Figure 1.1 – Système développé par P. Lemieux et coll.

LEMIEUX et coll. ont choisi des produits commerciaux pour tous les modules qu'ils n'ont pas réalisés eux-mêmes (voir figure 1.1). Les caméras sont de type «Pan-Tilt-Zoom». La transmission vidéo est effectuée grâce au codec Tanberg T3000. L'affichage est réalisé sur des écrans cathodiques ou à cristaux liquides conventionnels.

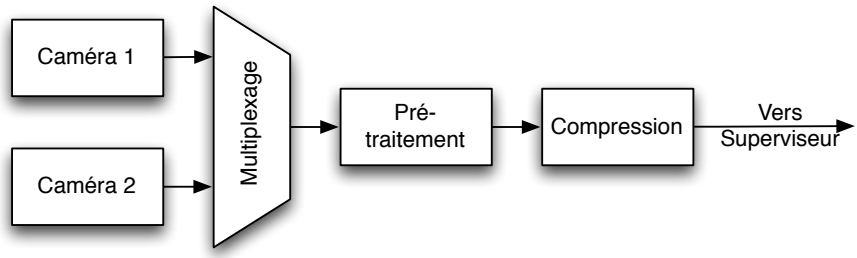
Bien que les caméras et les écrans répondent bien aux besoins de l'application, une amélioration est souhaitée en ce qui a trait à la transmission vidéo. En premier lieu, un plus grand contrôle sur le codec est souhaité. Cela permettrait de réduire certains types de bruit plus critiques pour l'application considérée. Le jugement de la qualité de la vidéo est très subjectif et varie d'une application à l'autre[2]. En second lieu, des capacités avancées de compensation d'éclairage, de réduction du flou dû au mouvement et de synchronisation avec l'information auditive de la salle d'opération sont désirables. Finalement, la considération de l'aspect commercial de l'application envisagée rend désavantageux l'utilisation du codec Tanberg en raison de son coût.

[2]L'application visée par le codec Tanberg T3000 est la téléconférence, permettant la transmission de deux canaux vidéo, un compressé à 25% et l'autre à 75%. Les critères de qualité sont différents pour l'application de télésupervision médicale considérée.

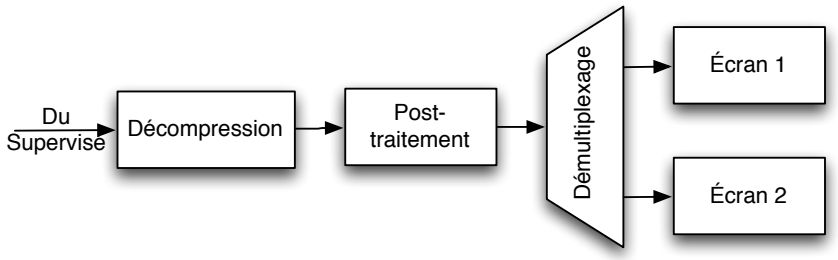
1.2 Solution proposée

En considérant les facteurs énoncés à la section 1.1, plusieurs solutions sont possibles. À court terme, il est avantageux de conserver le codec actuel (Tanberg T3000) en ajoutant des blocs de pré-traitement et de post-traitement pour le rendre compatible au projet. Cette solution implique un temps de conception et de réalisation réduit pour arriver à un résultat satisfaisant rapidement. À long terme, il est avantageux de créer un codec conçu spécifiquement pour les besoins du projet afin d'avoir un plus grand contrôle sur les performances du système de transmission vidéo. Cela a également pour avantage d'éviter les coûts des licenses d'utilisation d'un codec disponible commercialement.

La solution privilégiée est la conception d'un nouveau codec sur mesure pour le système de Lemieux et coll.. Ainsi, en ayant le plein contrôle sur le codec, il peut être plus facilement adapté à d'autres projets en télésanté en cours au LABORIUS. À son expression la plus simple, le codec peut être schématisé tel qu'à la figure 1.2.



(a) Encodeur



(b) Décodeur

Figure 1.2 – Vue d'ensemble du codec vidéo proposé pour l'application de P. Lemieux et coll.

Les étapes de multiplexage et de démultiplexage de la figure 1.2 consistent en le regroupement et la séparation des images. Cela permet de n'utiliser qu'un seul canal et donc de réduire la quantité de matériel requis, en plus d'éliminer la nécessité de synchroniser les deux vidéos à l'affichage puisqu'elles ne feront qu'une durant la transmission. Cette syn-

chronisation est d'autant plus importante, car les deux vidéos partagent potentiellement une partie commune de la scène observée.

Le bloc de pré-traitement est responsable de plusieurs tâches et pourrait d'ailleurs être séparé en plusieurs blocs. L'une de ses tâches est la compensation du flou dû au mouvement. Étant donné la mobilité des caméras, un flou sera introduit en raison du temps d'exposition encore assez loin de l'infiniment court des caméras modernes. Puisque le mouvement des caméras est connu, cela facilite l'annulation numérique du flou introduit. Le bloc de pré-traitement est également responsable de compenser les conditions d'éclairage sous-optimales afin que des images transmises soient les plus claires possibles. De plus, il est souhaitable que le pré-traitement comprenne une forme de compensation des défauts du codec à réaliser. Il s'agit de modifier légèrement les images afin de faciliter le travail des blocs de compression et de décompression.

Le bloc de compression est responsable de réduire au maximum la quantité d'information transmise en tolérant une certaine perte d'information afin d'obtenir une compression plus importante. Cette réduction est réalisée en représentant l'information sous une forme plus efficace. Le bloc de décompression effectue l'opération contraire, c'est-à-dire de retransformer l'information de la forme efficace vers la forme visible.

Puisqu'une certaine perte d'information est tolérée à l'étape de compression, des défauts sont introduits dans la vidéo. Le bloc de post-traitement est responsable de réduire ces défauts et donc d'améliorer la qualité subjective de l'image observée.

1.3 Spécifications globales

L'ensemble de la solution proposée à la section 1.2 doit être conçue et réalisée de telle façon que les besoins du système de Lemieux et coll. soient satisfaits. Ces besoins peuvent être regroupés selon trois thèmes : latence, débit et qualité.

Latence Puisque le système doit permettre au superviseur d'observer et de fournir des commentaires et suggestions en temps réel, il est nécessaire de minimiser la latence. En se fiant au travaux de Na et Yoo [Na et Yoo, 2002] sur la satisfaction des utilisateurs selon la latence pour le VOIP[3], la satisfaction des utilisateurs décroît considérablement à partir de 200 ms, comme le montre la figure 1.3. Une latence de plus de 200 ms est donc jugée inacceptable. Toujours selon les mêmes travaux,

[3]Puisque la vidéo est synchronisée avec la communication vocale, la transmission vidéo est soumise aux mêmes requis de latence.

les utilisateurs ne remarquent pas le délai si la latence est de moins de 100 ms. Cette latence est considérée comme idéale. Une latence entre 100 et 200 ms est jugée acceptable. Il ne faut pas oublier que l'évaluation de la latence doit être de la caméra à l'écran et inclut donc non seulement les délais induits par le codec mais également par le lien de communication sous-jacent. Par ailleurs, pour des raisons de confort visuel, il est nécessaire que la latence soit la même pour les deux vidéos. Cette condition est automatiquement remplie en adoptant la solution proposée à la section 1.2.

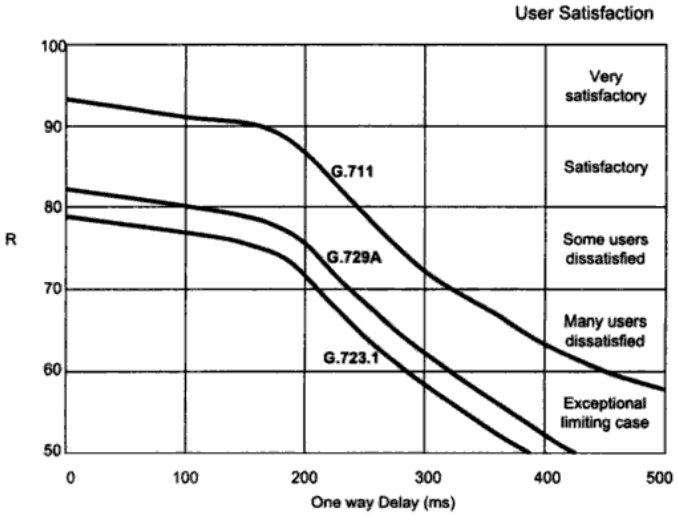


Figure 1.3 – Satisfaction des utilisateurs selon la latence [Na et Yoo, 2002]

Débit La bande passante disponible pour la transmission vidéo (pour la téléconférence sur le réseau hospitalier) est de 384 kbps (kilobits par seconde). La résolution native des caméras est de 720×480 en YUV `4:2:0` pour une moyenne de 12 bits par pixel[4]. Le tableau 1.1 montre la bande passante selon la cadence des images (en images par seconde) et selon la résolution choisie. Le tableau 1.2 montre cette même information mais exprimée en termes de taux de compression. Il est pour l'instant absolument impensable d'atteindre des taux de compression de plus de `400:1` à une qualité raisonnable. Il est donc exclus de transmettre les images en résolution native à pleine cadence. Il faut déterminer quel est le compromis entre la cadence et la résolution des images qui satisfait le mieux les besoins de l'application considérée. Il est difficile d'établir un objectif précis de taux de compression parce que celui-ci varie énormément selon la quantité de mouvement dans les vidéos. Ce mouvement est influencé principalement par les déplacements des caméras et des sujets filmés.

[4]Huit bits de Y, deux bits de U et deux bits de V.

TABLEAU 1.1 – Bande passante (bits par pixel) selon la cadence et la résolution des images

	Résolution des images transmises				
Cadence	Native	/2	/4	/8	/16
10	0.0370	0.0741	0.1481	0.2963	0.5926
15	0.0247	0.0494	0.0988	0.1975	0.3951
20	0.0185	0.0370	0.0741	0.1481	0.2963
25	0.0148	0.0296	0.0593	0.1185	0.2370
30	0.0123	0.0247	0.0494	0.0988	0.1975

TABLEAU 1.2 – Taux de compression requis selon la cadence et la résolution des images

	Résolution des images transmises				
Cadence	Native	/2	/4	/8	/16
10	216	108	54	27	14
15	324	162	81	41	20
20	432	216	108	54	27
25	540	270	135	68	34
30	648	324	162	81	41

En l'absence d'information sur les habitudes des télésuperviseurs et de la quantité de mouvements des opérations où le système sera utilisé, il est possible d'espérer un taux de compression entre `75:1` et `250:1`.

Qualité La qualité d'image des vidéos transmis est très importante dans l'application considérée. Cette qualité doit être assez élevée pour que le superviseur puisse poser un diagnostic et qu'il puisse observer les détails de l'opération en cours. Les dégradations subies par les images doivent affecter le moins possible les tâches du superviseur. Or, l'évaluation objective de la qualité de la vidéo est difficile et plusieurs chercheurs s'y intéressent toujours [Pinson et Wolf, 2004; Wang et coll., 2004; Watson et coll., 2001]. Il faut donc choisir ou concevoir une métrique adaptée à l'application considérée et optimiser le codec de façon à maximiser la mesure de qualité choisie. Il est également possible que la meilleure métrique soit de demander à des télésuperviseurs de choisir, entre deux configurations de codec, la vidéo qu'ils percoivent de meilleure qualité. Par une série de tests avec un bon nombre de télésuperviseurs, il est possible d'optimiser la qualité vidéo du codec. À ce stade, en l'absence de métrique, il n'est pas possible d'établir une spécification objective en ce qui a trait à la qualité de la vidéo.

1.4 Identification du projet

La solution proposée à la section 1.2, soit le développement complet d'un codec vidéo de bonne qualité, est un projet de grande envergure. La compensation du flou relié au mouvement, la compensation de l'éclairage, le post-traitement et la compression sont tous des sujets susceptibles de justifier un ou plusieurs projets de recherche différents. La réalisation complète du projet en question est donc inatteignable dans le cadre d'un seul projet de maîtrise.

Puisque que le projet en est à ses débuts et qu'un codec vidéo commence d'abord et avant tout par un codec d'images, c'est celui-là qui est le centre de cette recherche. Il est donc question de la compression d'une seule image sans considérer les images précédentes et suivantes d'une vidéo. Le fait de ne pas connaître les images précédentes et suivantes réduit de beaucoup la facilité à réduire la taille de l'information. En effet, puisque la redondance dans le temps est très importante (une grande zone des images qui se suivent est identique, et ce particulièrement si la cadence augmente), cela permet d'augmenter de beaucoup le taux de compression. Il ne faut donc pas confondre les spécifications globales du système présenté à la section 1.3 avec les spécifications qui sont définies pour la compression d'images.

Latence. La minimisation de la latence du système global implique une minimisation de la latence du codec d'images. Il est donc nécessaire de limiter la complexité temporelle de l'algorithme. Cela implique aussi que l'algorithme ne pourra pas utiliser d'itérations pour améliorer la compression. Idéalement, la décompression doit pouvoir débuter son travail avant que la compression ait terminé le sien. Cela implique que l'information transmise soit sérialisée ou que le codec opère sur des morceaux d'images. Le traitement de l'image en morceaux (ou en blocs) est d'autant plus souhaitable, car il facilite souvent la parallélisation de l'algorithme.

Débit. La notion de débit à atteindre est très difficile à établir. Celui-ci dépend énormément de la complexité des images. Il est tout à fait possible d'atteindre un débit de 0,07 bpp pour une image très simple, mais plutôt difficile d'atteindre 0,15 bpp pour une image très complexe en gardant une qualité de l'image suffisante. Le débit disponible pour la compression de l'image dépend également de la compression qui est possible au niveau du codec vidéo. Il est tout à fait envisageable que la redondance temporelle dans les vidéos permette un facteur de compression de deux à elle seule. Cela implique que les débits présentés au tableau 1.3 sont disponibles.

TABLEAU 1.3 – Bande passante pour une image (bits par pixel) selon la cadence et la résolution

	Résolution des images transmises				
Cadence	Native	/2	/4	/8	/16
10	0.0741	0.1481	0.2963	0.5926	1.1852
15	0.0494	0.0988	0.1975	0.3951	0.7901
20	0.0370	0.0741	0.1481	0.2963	0.5926
25	0.0296	0.0593	0.1185	0.2370	0.4741
30	0.0247	0.0494	0.0988	0.1975	0.3951

Plus le taux de compression atteint est grand, plus la qualité de l'image transmise peut être grande. Il n'est donc pas utile de fixer un objectif en termes absolus. Il est plus logique de fixer un objectif qualité/débit cohérent avec le domaine.

Qualité. Dans le contexte de l'application envisagée, la qualité d'image est une notion intrinsèquement reliée au débit et les objectifs doivent être établis de pair.

Il est imposé, pour ce projet de recherche, de se baser sur les travaux de Ponomarenko et coll. [Ponomarenko et coll., 2005, 2007] parce que ces travaux ont été identifiés comme à la fine pointe de la science et parce qu'ils présentent beaucoup de potentiel. Leurs approches sont présentées au chapitre 2.4.1.

La structure adaptative du codec AGU implique que les blocs soient traités en série. Cette contrainte est contraire aux requis de latence du système. Il est donc requis de modifier le codec afin de le rendre parallélisable. En utilisation des blocs de 32×32 et la résolution native des images, cela représente une réduction de latence d'un facteur de plus de 300. Toutefois, cette amélioration ne doit pas se faire au détriment de la complexité du codec. Sa complexité doit rester semblable ou être réduite.

La performance qualité/débit du nouveau codec créé est susceptible d'être réduite par la suppression de la structure du codec AGU. Cette réduction ne doit pas amener la performance du codec sous la performance de la figure établie JPEG2000 abordé à la section 2.4.4. Plus précisément, la performance du nouveau codec doit être supérieure à JPEG2000 à un débit de 0,25 bpp pour les images de test classiques Lenna, Barbara, Baboon, Goldhill et Peppers.

L'accomplissement des objectifs ci-haut mène à un codec simple, parallélisable et supérieur a JPEG2000. Cela constitue une bonne pierre d'assise pour l'évolution de celui-ci en un codec vidéo. Il est donc nécessaire qu'une implémentation des éléments important non-triviaux du codec soient livrés de façon à ce qu'ils soient facilement réutilisables pour la suite du projet global.

1.5 Organisation du document

Au chapitre 2, une revue de l'état de l'art est présentée sur le codage d'images en lien avec les travaux de PONOMARENKO et coll.. Par la suite, les différentes approches existantes en ce qui a trait à la transformation, la quantification et le codage sont abordées. Les principaux codecs du domaine sont aussi comparés.

Le chapitre 3 présente une description de la contribution du présent mémoire, soit les stratégies adoptées pour améliorer les codecs de PONOMARENKO et coll., suivie d'une description des systèmes réalisés accompagnée des explications des choix effectués.

Les résultats sont présentés et mis en perspective au chapitre 4 sous la forme d'un article scientifique soumis le 16 avril 2009 à *IEEE Signal Processing Letters*. Quelques pistes d'amélioration sont énoncées et le codec réalisé est évalué face aux requis du projet global.

CHAPITRE 2

COMPRESSION D'IMAGE AVEC PERTES PAR TRANSFORMÉE

Ce chapitre s'intéresse aux trois opérations fondamentales de la compression d'image par transformée, soient la transformation, la quantification et le codage. Les techniques considérées comme la fine pointe de la technologie sont abordées. En fin de chapitre, les codecs les plus performants sont présentés et comparés.

2.1 Transformées pour les images

La très grande corrélation des pixels entre eux implique une représentation de l'information peu efficace. Transformer les images de façon à modifier leur représentation est une stratégie éprouvée de compression. Qui plus est, la quantification de l'information dans le domaine des transformées habituelles affecte moins la qualité de l'image que dans le domaine d'affichage[5]. Ce chapitre présente les transformées pertinentes au domaine.

2.1.1 Transformée Karhunen-Loève

La transformée Karhunen-Loève (KLT) est une transformée linéaire dont les fonctions sont choisies selon la matrice de covariance de la source. En terme de compaction d'énergie, la KLT est optimale et fournit donc des coefficients complètement décorellés. Elle peut mener, entre autres, à l'analyse en composantes principales.

Dans un contexte de compression, il s'agit, en quelque sorte, de la transformée idéale puisqu'elle décorrelle complètement les entrées. Toutefois, la transformée doit être recalculée chaque fois que la source change et puisque les fonctions propres de la KLT dépendent de la source, celles-ci doivent être transmises. Dans un contexte de compression d'image ou de vidéo, cela signifie à chaque image ou même à chaque bloc traité. Cette caractéristique la rend difficilement utilisable.

[5]On fait référence ici au domaine où l'image est un tableau de valeurs représentant la luminance ou la chrominance.

2.1.2 DCT

La transformée en cosinus discrète (DCT) a été proposée à l'origine dans [AHMED et coll., 1974] et est l'une des plus utilisées en compression d'image. Elle s'apparente à la transformée discrète de Fourier, mais n'utilise que des nombres réels. Pour des images naturelles, elle se rapproche de la KLT en termes de compaction d'énergie et de décorellation des entrées. Les codecs d'images utilisent habituellement la version bidimentionnelle de la DCT présentée à l'équation 2.1 à titre indicatif. Les coefficients de la DCT correspondent aux fonctions montrées à la figure 2.1.

$$
\begin{aligned}
C_u &= \begin{cases} \frac{1}{\sqrt{2}} & \text{si } u = 0 \\ 1 & \text{sinon} \end{cases} \\
C_v &= \begin{cases} \frac{1}{\sqrt{2}} & \text{si } v = 0 \\ 1 & \text{sinon} \end{cases} \\
F_{vu} &= \tfrac{1}{4} C_v C_u \sum_{y=0}^{N-1} \sum_{x=0}^{N-1} S_{yx} cos\left(v\pi \frac{2y+1}{2N}\right) cos\left(u\pi \frac{2x+1}{2N}\right)
\end{aligned}
\tag{2.1}
$$

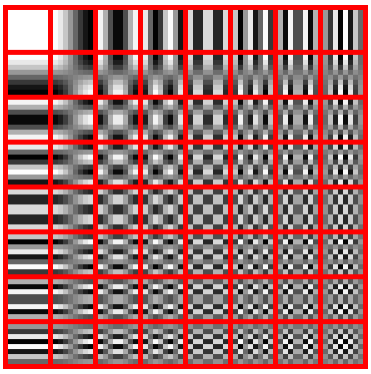


Figure 2.1 – Fonctions de la DCT pour un bloc 8×8

La DCT a l'avantage de rassembler l'information en peu de coefficients peu corrélés et présente donc une compaction semblable à la KLT pour les images naturelles. Toutefois, le bruit de quantification fait apparaître une ou plusieurs fonctions de la DCT et cela brise le caractère naturel de l'image. Puisque la DCT est souvent utilisée en blocs, un effet de bloc apparaît à l'image reconstituée et cela rend nécessaire un filtrage. Finalement, elle génère fréquemment un bruit moustique. La figure 2.2 illustre ces types de bruit (l'image originale est présentée à la figure A.1a).

Figure 2.2 – Lenna à très bas débit par un codec utilisant la DCT

2.1.3 DWT

La transformée par ondelettes discrète (DWT) consiste à décomposer l'image en plusieurs sous-bandes (images de résolution inférieure). Un exemple d'une telle décomposition est présenté à la figure 2.3. Plusieurs sortes d'ondelettes peuvent être utilisées pour effectuer la décomposition.

L'utilisation de la DWT donne d'excellents résultats pour les images comportant des contours marqués comme les bandes dessinées, mais propose des performance habituellement inférieures à la DCT pour les images comportant beaucoup de textures. Il reste que ce type de transformée est plus jeune que la DCT et des améliorations apparaissent encore possibles.

2.1.4 Transformée Orthogonale à recouvrement

Une transformée orthogonale à recouvrement (LOT pour «Lapped Othogonal Transform»[6]) n'est pas une transformée en soi, mais une façon d'appliquer une transformée. Certaines transformées, la DCT par exemple, est habituellement utilisée sur des blocs disjoints d'une image. Après la quantification, des artefacts apparaîssent aux bordures des

[6]Dans la même catégorie, se trouvent les transformées orthogonales à recouverment généralisées (GenLOT) et les transformées biorthogonales à recouvrement généralisées (GLBT).

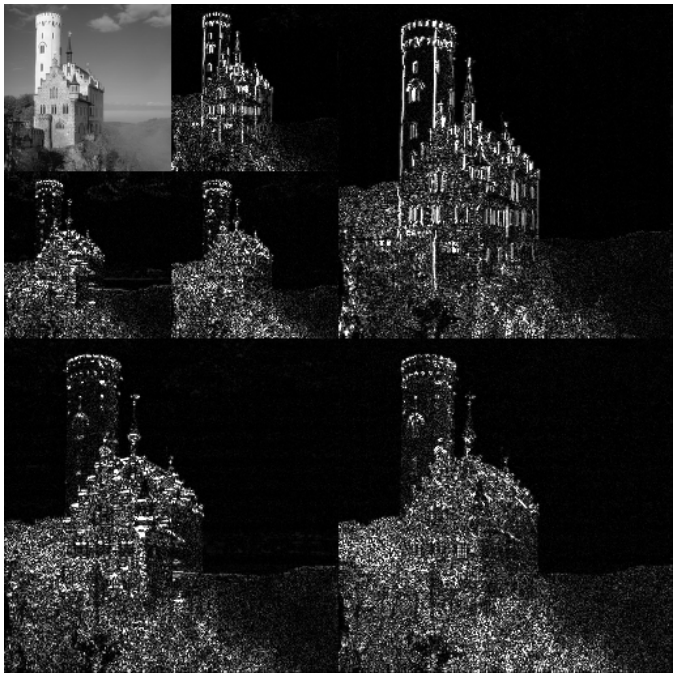


Figure 2.3 – Exemple de décomposition en sous-bandes JPEG 2000

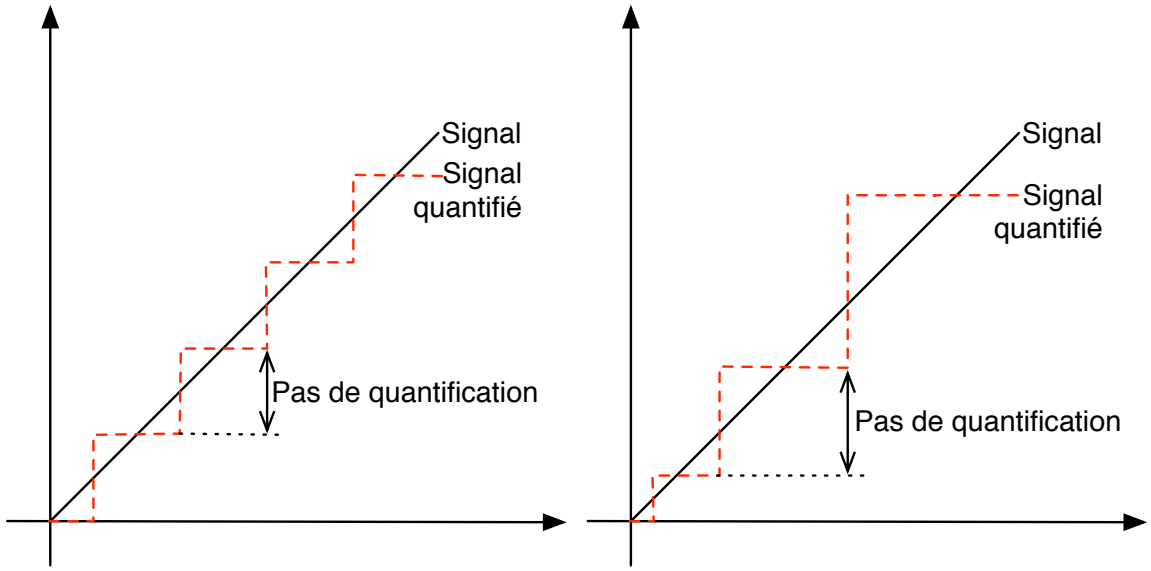
blocs, particulièrement à bas débit. Une des stratégies pour combattre ce problème est de considérer des points à l'extérieur du bloc lors de la transformée, mais sans obtenir plus de coefficients que la transformée normale. Il s'agit alors d'une LOT. Des paramètres sont à déterminer avant chaque transformées et plusieurs approches existent : [Cassereau et coll., 1989] propose une méthode itérative, [Malvar et Staelin, 1989] propose une méthode basée sur les vecteurs propres, etc. Ce type de transformée offre l'avantage très intéressant de réduire de beaucoup l'effet de bloc de la DCT à un coût très raisonnable en termes de performance.

2.2 Quantification

La quantification vise à représenter un signal avec un alphabet réduit tout en minimisant l'erreur. Ce chapitre présente les approches les plus communes pour la quantification dans le contexte de la compression.

2.2.1 Scalaire

La quantification scalaire (SQ) est la plus simple des quantifications. Elle consiste en l'association d'une plage à un niveau de quantification. Si tous ces niveaux sont distribués



(a) Uniforme (b) Non-uniforme

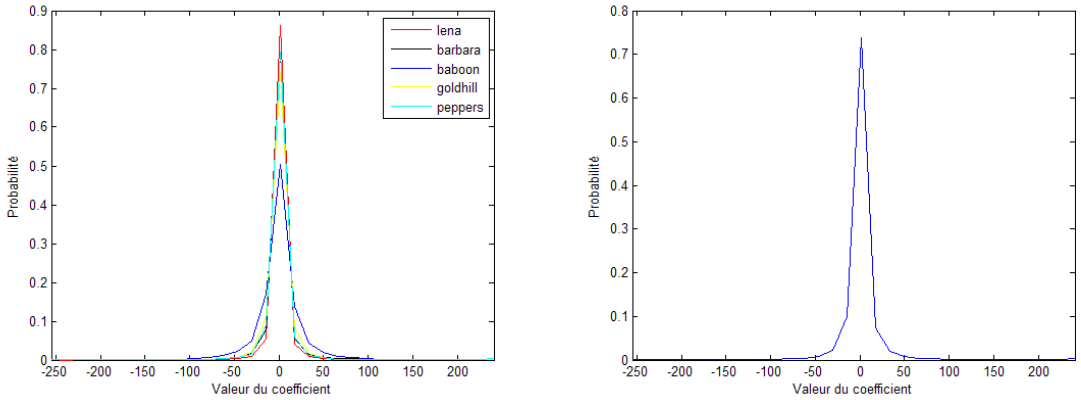
Figure 2.4 – Quantification uniforme et non-uniforme

uniformément, il s'agit alors de quantification scalaire uniforme (pas de quantification constant). La quantification uniforme est illustrée à la figure 2.4a et mise en contraste avec une quantification non-uniforme à la figure 2.4b.

Pour minimiser l'erreur, la quantification scalaire doit être adaptée aux statistiques de la source. Pour une source non-uniforme (gaussienne par exemple), une quantification non-uniforme doit être utilisée[7]. Bien qu'une quantification scalaire uniforme soit habituellement utilisée, l'étude de la densité de probabilité des valeurs des coefficients (DCT réalisée en blocs de 32×32[8]) révèle une probabilité beaucoup plus importante autour de zéro comme le montre la figure 2.5. Il pourrait donc être avantageux d'exploiter cette caractéristique pour réduire l'erreur.

Un élément intéressant à tirer de la figure 2.5 est que l'image dont la courbe de densité de probabilité est la plus applatie est la plus difficile à compresser, et que l'image ayant la courbe la plus concentrée autour de zéro est la plus facile à compresser.

[7]Il serait alors utile d'utiliser l'algorithme de Lloyd-Max pour calculer les niveaux de quantification.
[8]La même forme de densité de probabilité est obtenue pour d'autres tailles de blocs.



(a) Pour chaque image (b) Moyenne

Figure 2.5 – Densité de probabilité des images de test

2.2.2 Matricielle

La quantification matricielle est très utilisée en compression d'image, principalement de pair avec la DCT. Son utilisation est justifiée par le fait que l'oeil humain est plus sensible à certaines fréquences spatiales que d'autres. En effet, l'œil humain est bien meilleur pour voir des légères différences de luminosité sur de grands espaces que les valeurs exactes d'un petit espace où la luminosité varie beaucoup. Il est donc opportun de quantifier davantage l'information haute fréquence par rapport à l'information basse fréquence pour améliorer la qualité subjective de l'image. C'est ce que la quantification matricielle propose. Sachant que pour une DCT bidimentionnelle les coefficients basse fréquence se trouvent dans le coin supérieur gauche, la matrice présentée à la figure 2.6 [Pennebaker et Mitchell, 1992] pourrait être utilisée.

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Figure 2.6 – Matrice répandue de quantification DCT 8×8

$$\begin{bmatrix} 181 & 173 & 156 & 141 \\ 192 & 180 & 138 & 118 \\ 188 & 183 & 141 & 116 \\ 185 & 176 & 157 & 123 \end{bmatrix}$$

(a) Données originales, $m = 159$

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

(b) Classification, $m_1 = 182$, $m_0 = 136$

$$\begin{bmatrix} 182 & 182 & 136 & 136 \\ 182 & 182 & 136 & 136 \\ 182 & 182 & 136 & 136 \\ 182 & 182 & 136 & 136 \end{bmatrix}$$

(c) Données reconstruites

$$\begin{bmatrix} -1 & -9 & 20 & 5 \\ 10 & -2 & 2 & -18 \\ 6 & 1 & 5 & -20 \\ 3 & -6 & 21 & -13 \end{bmatrix}$$

(d) Erreur

Figure 2.7 – Exemple de quantification BTC

2.2.3 Block truncation coding

Le *block truncation coding* (BTC) consiste en une quantification à dictionnaire assez simple et efficace. Pour une quantification à 1 bit, l'algorithme calcule d'abord la moyenne des données et les classifie selon qu'elles soient supérieures ou inférieures à la moyenne. L'appartenance à chacune des classes peut être exprimée par un tableau de bits. Les moyennes de ces deux classes sont calculées et quantifiées. Ainsi, pour un tableau 4×4 et deux moyennes quantifiées à 8 bits chacune, la transmission totalise 32 bits $(16 + 8 + 8)$, soit 2 bits par pixel. La valeur de la moyenne d'une classe est attribuée à toute la classe lors de la déquantification. Un exemple de la situation exprimée ci-haut est présenté à la figure 2.7. Il est possible d'effectuer une quantification à un plus haut nombre de bits en répétant l'algorithme pour chacune des classes. À chaque ajout d'un bit, le nombre de moyennes à transmettre double et les bits de classification doublent également.

2.2.4 Espace-fréquence

La quantification espace-fréquence [Xiong et coll., 1997] (SFQ) est une approche qui se base sur le fait que l'information disponible après la DWT est concentrée dans la bande basse fréquence et concentrée spatialement pour les hautes fréquences (contours). Cet algorithme tente d'identifier le meilleur pas de quantification selon un budget de bande passante. Celui-ci est basé sur un processus d'optimisation itératif et donc incompatible avec les besoins et spécifications du projet.

2.3 Codage

Le codage tel qu'abordé dans ce chapitre est une opération sans perte (dite «lossless») en le sens qu'elle ne modifie pas les données. Le codage cherche à éliminer toute redondance de telle façon que la quantité d'information à transmettre soit minimale. La limite théorique de codage à atteindre est dictée par le théorème de codage des sources de Claude E. Shannon [SHANNON, 1948] et de l'entropie du signal à encoder.

2.3.1 Par plage

Le codage par plage, mieux connu sous l'apellation «run-length encoding» (RLE), est un codage exploitant la répétition de données. Si un pixel noir est représenté par la lettre N et un pixel blanc par la lettre B, la ligne suivante :

```
BBBBBBBBBBBBNBBBBBBBBBBBBNNNBBBBBBBBBBBBBBBBBBBBBBBBNBBBBBBBBBBBBBB
```

est encodée ainsi en RLE :

```
12B1N12B3N24B1N14B
```

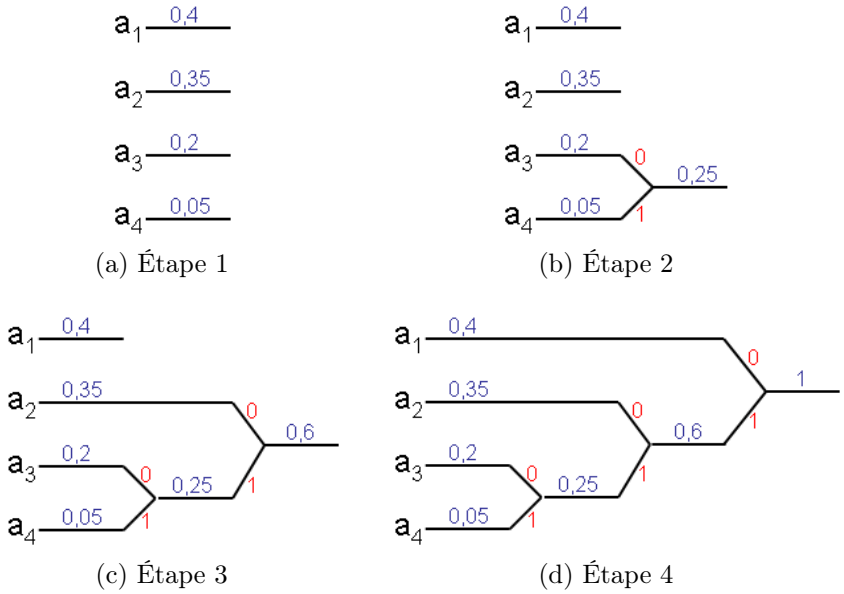
Pour chaque séquence d'un même symbole, l'algorithme indique le nombre de répétitions et le symbole. Le caractère répétitif doit être prédominant, car le surdébit est très important pour les symboles non répétés.

2.3.2 Huffman

L'approche proposée à l'origine par David A. Huffman [HUFFMAN, 1952] est une méthode pour établir les mots de codes associés aux symboles de façon optimale. Le codage Huffman s'effectue en construisant un arbre binaire en partant des feuilles ayant les plus petites probabilités. Ainsi, si une source émet quatre symboles a_1, a_2, a_3 et a_4 et que ces symboles ont respectivement les probabilités $0,4$, $0,35$, $0,2$ et $0,05$, il faut d'abord joindre a_3 et a_4. Le processus complet est illustré à la figure 2.8. En lisant les codes obtenus de la racine vers les feuilles, les codes présentés au tableau 2.1 sont obtenus.

Le codage Huffman n'est généralement pas optimal. En raison de l'attribution de mots de code d'une longueur entière, il ne peut être optimal que si toutes les probabilités des symboles sont des puissances négatives de deux[9].

[9] Par exemple, 1/2, 1/4, 1/8, 1/16, etc.



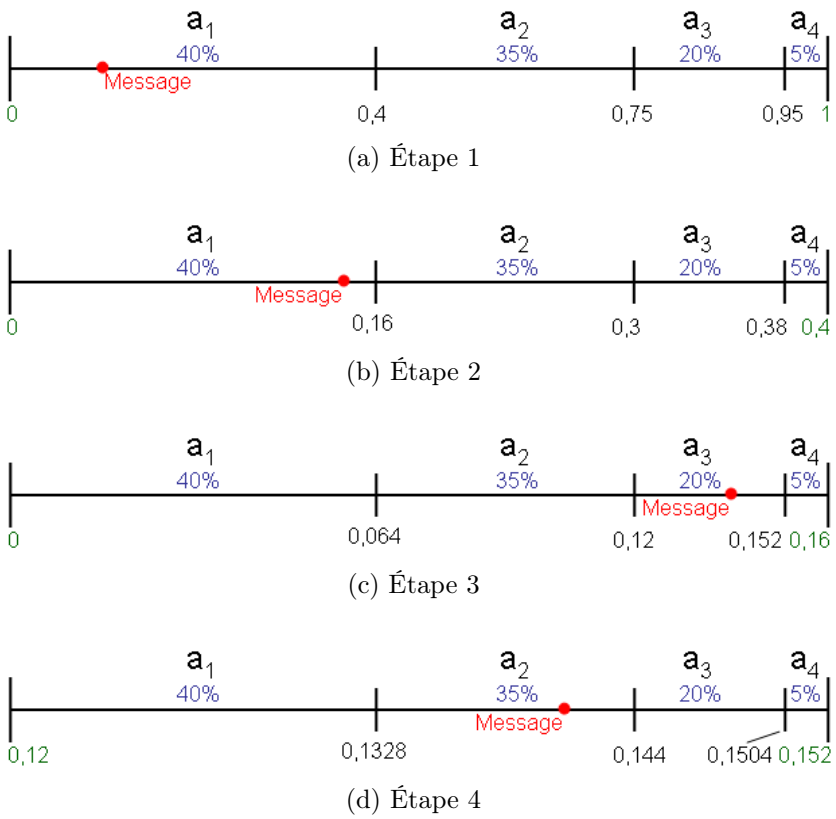
(a) Étape 1 (b) Étape 2

(c) Étape 3 (d) Étape 4

Figure 2.8 – Exemple de construction d'un arbre pour le codage Huffman

TABLEAU 2.1 – Résultat de la construction de l'arbre pour le codage Huffman

Symbole	Code
a1	0
a2	10
a3	110
a4	111



(a) Étape 1

(b) Étape 2

(c) Étape 3

(d) Étape 4

Figure 2.9 – Exemple de décodage arithmétique

2.3.3 Arithmétique

Le codage arithmétique [Rissanen et Langdon, 1979] est une généralisation du codage de Huffman qui élimine la limitation de la longueur entière des mots de code. L'approche utilisée pour y arriver est de représenter un message complet par une fraction entre 0 et 1. En divisant l'intervalle selon les probabilités de chaque symbole et en déterminant dans quel sous-intervalle la fraction se trouve, il est possible de déterminer le message. Le décodage étant beaucoup plus simple que l'encodage, il est utilisé à titre d'exemple pour illustrer le procesus.

Supposons que le message encodé est la fraction 0.1416 et que la source est la même qu'à la section 2.3.2. Supposons également que quatre symboles doivent être décodés de ce message. La première étape est de diviser l'intervalle selon les statistiques de la source comme le montre la figure 2.9a. En réutilisant le sous-intervalle dans lequel le message se trouvait comme nouvel intervalle, le processus se répète comme le montre les figures 2.9b, 2.9c et 2.9d. Le message est donc a_1, a_1, a_3, a_2.

Bien que le codage arithmétique puisse atteindre l'optimalité en théorie, certains facteurs l'en empèchent en pratique. Le facteur le plus important est bien certainement la connaissance des probabilités des symboles de la source. Puisque celles-ci varient et sont souvent imprévisibles, il n'est pas possible d'atteindre l'optimalité. À un degré moindre, la précision de l'implémentation empèche l'optimalité. Puisque les calculs sont faits avec une précision limitée, il n'est pas possible d'utiliser la véritable probabilité d'un symbole si elle est de $10^{-100000}$.

Le codage arithmétique binaire [Langdon et Rissanen, 1981] (BAC) est identique au codage arithmétique sauf que seuls les symboles 0 et 1 sont encodés. Cela a pour avantage majeur de simplifier beaucoup l'algorithme. En contrepartie, le débit de l'algortihme est réduit parce que ses calculs sont effectués pour un seul bit à la fois.

Le codage arithmétique binaire adapté au contexte [Pennebaker et coll., 1988] (CABAC) est une évolution du BAC qui utilise le contexte de chaque symbole (bit) pour déterminer sa probabilité et ainsi l'encoder efficacement. Cette approche débute par la création de plusieurs contextes. La classification d'un symbole dans un de ces contextes dépend d'un certain nombre de conditions qui examinent les symboles encodés précédemment. Une fois le symbole classifié dans un contexte, il est possible d'examiner les probabilités de ce symbole dans son contexte en se basant sur tous les symboles précédents classés dans le contexte en question.

Par exemple, supposons que le bit A de la figure 2.10 soit à coder et que par des recherches antérieures il ait été déterminé que trois contextes sont pertinents et qu'ils dépendent des bits G et H. Plus précisément, le système de conditions présenté à la figure 2.11 est utilisé pour déterminer le contexte probabiliste approprié du bit. Selon tous les bits qui ont été classifiés depuis le début du codage dans chacun de ces contextes, les probabilités présentées au tableau 2.2 ont été calculées. Ainsi, si $H = G = A = 1$ le modèle P_1 sera utilisé et la probabilité d'un bit 1 étant élevée la compression sera efficace. L'objectif est d'obtenir des contextes ne contenant qu'un seul type de bit pour maximiser sa probabilité et du fait même la compression obtenue. Ce nouveau bit s'ajoutant au contexte viendra renforcer la probabilité de $A = 1$ et améliorer la compression des prochains bits qui sont dans la même situation.

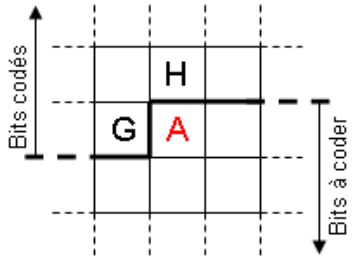


Figure 2.10 – Exemple de bit à coder

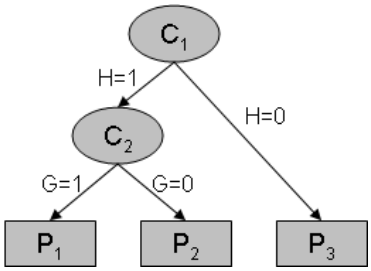


Figure 2.11 – Exemple d'arbre de conditions permettant la classification

TABLEAU 2.2 – Exemples de probabilités associées aux contextes

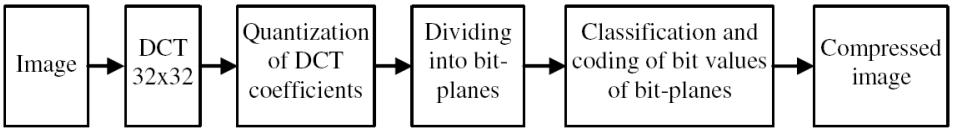
	H	G	P(A=0)	P(A=1)
P_1	1	1	10%	90%
P_2	1	0	60%	40%
P_3	0	-	90%	10%

2.4 Approches de compression

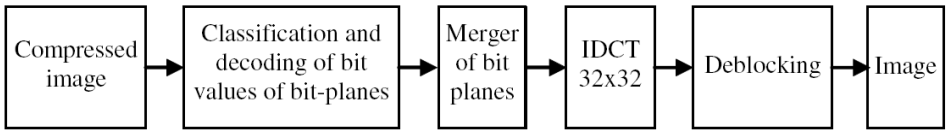
Dans ce chapitre, les principaux compétiteurs dans le domaine de la compression d'image avec perte sont explorés ainsi que quelques figures notables, soit historiquement, soit pour une idée particulièrement intéressante. Les résultats sont réservés à la fin du chapitre (section 2.4.10) où un comparatif des performances est effectué.

2.4.1 AGU – DCT based high quality image compression

La première approche de Ponomarenko et coll. [Ponomarenko et coll., 2005], nommée AGU, traite l'image en bloc de 32×32. Les blocs subissent d'abord une transformée en cosinus discrète (DCT) telle que décrite à la section 2.1.2. Ensuite, les coefficients obtenus par la transformée sont quantifiés uniformément. Le facteur de quantification est une entrée du codec. La compression se termine par un codage arithmétique binaire adapté au contexte (CABAC), abordé plus en détail à la section 2.3.3. La décompression est composée des opérations inverses auxquelles vient s'ajouter un post-traitement visant à éliminer les artefacts créés par le traitement en bloc. La compression et la décompression sont représentées en schémas-blocs à la figure 2.12.



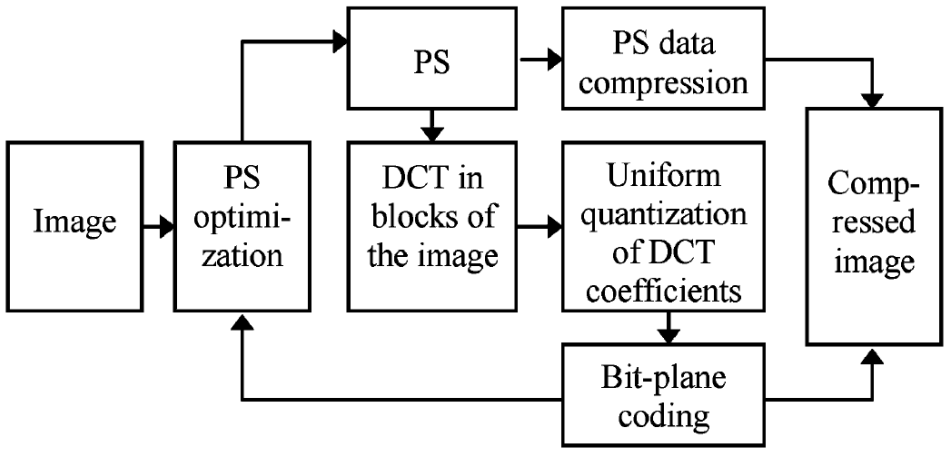
(a) Compression



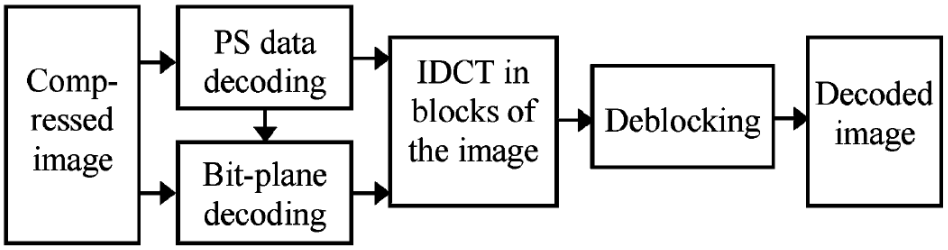
(b) Décompression

Figure 2.12 – Schémas-blocs du codec de l'approche AGU de Ponomarenko et coll. [Ponomarenko et coll., 2005]

La contribution principale de cette approche se situe au niveau de l'étape du codage, plus spécifiquement en ce qui a trait au modèle de contexte. Celui-ci est basé sur une classification des bits dans 15 catégories possibles. La classification vise à obtenir des classes les plus uniformes possibles. Cette uniformité permet au codage arithmétique d'atteindre un taux de compression élevé.



(a) Compression



(b) Décompression

Figure 2.13 – Schémas-blocs du codec de l'approche AGU-MHV de Ponomarenko et coll. [Ponomarenko et coll., 2007]

2.4.2 AGU-MHV – High-quality DCT-based image compression using partition schemes

Leur seconde approche [Ponomarenko et coll., 2007], nommée AGU-MHV (*AGU Modified Horizontal Vertical*), est très sembable à la première à l'exception de la taille des blocs. En effet, plutôt que d'utiliser des blocs de 32×32, cette seconde approche utilise des blocs de différentes tailles allant de 64×64 à 8×8. Comme le montre la figure 2.13, à l'exception de cette taille de blocs adaptative, l'approche reste sensiblement la même.

Pour choisir la taille optimale d'un bloc, le codec utilise un algorithme récursif qui débute avec un bloc de 64×64. À partir de ce bloc, l'algorithme effectue une compression et une décompression pour chacune des huit possibilités de division de bloc représentés à la figure 2.14. Celle qui obtient la meilleure qualité au meilleur taux de compression est choisie. Pour toutes les zones carrés (il y en a deux pour le type 6 par exemple), le processus

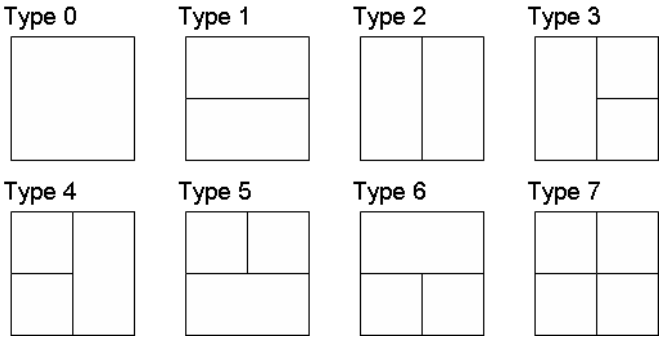


Figure 2.14 – Possibilités de division d'un bloc

est répété et s'arrête lorsqu'il n'y a plus de zones carrées ou lorsque la taille de celles-ci atteint 8×8.

Cette approche de taille adaptative de bloc permet d'améliorer en moyenne la qualité d'image (mesurée grâce au PSNR (*Peak Signal-to-Noise Ratio*)) de 0,5 dB par rapport à l'approche 1. Cette amélioration est toutefois très loin d'être gratuite. La nature récursive de l'algorithme le rend très lourd et nuit à l'implémentation matérielle pour l'application envisagée à la section 1.2.

Contrairement à l'approche AGU qui utilise l'information des blocs adjacents pour le CABAC, l'approche AGU-MHV ne considère que le bloc courant.

2.4.3 JPEG

La compression communément appelée JPEG [Pennebaker et Mitchell, 1992] (*Joint Photographic Experts Group*), le nom du comité l'ayant créée vers la fin des années 1980, est plus rigoureusement désignée ISO/IEC IS 10918-1 ou ITU-T Recommendation T.81. Cette spécification décrit un codec d'image qui fonctionne selon quatre modes d'opération : de base, sans-perte («lossless»), progressif, et hiérarchique. Les modes progressif et hiérarchique sont des variations du mode de base qui permettent une décompression partielle de l'image encodée pour obtenir une image basse résolution. Le mode sans-perte, quant à lui, est un algorithme complètement différent basé sur un codage prédictif.

L'algorithme de base décompose l'image en blocs 8×8 qui sont transformés grâce à la DCT et subissent ensuite une quantification matricielle. Le coefficient continu (DC) de chaque bloc est codé en utilisant une approche prédictive, et les coefficients alternatifs (AC) de chaque bloc sont traités en zigzag tel qu'indiqué à la figure 2.15. Ils sont codés à l'aide du codage Huffman.

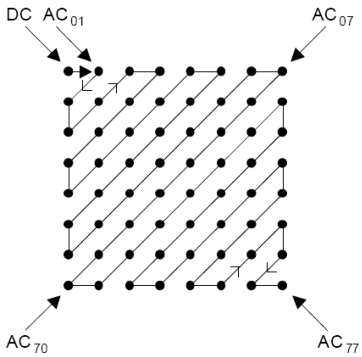


Figure 2.15 – Parcours zigzag d'un bloc

2.4.4 JPEG 2000

JPEG 2000 est un autre standard (ayant vu le jour en l'an 2000) produit par le même comité. En pratique, ce nouveau standard effectue d'abord un pré-traitement qui comprend, entre autres, une division en blocs optionnelle. Alors que la taille des blocs est fixée à 8×8 pour JPEG, JPEG 2000 fonctionne typiquement sans division en blocs ou avec une division en blocs de 64×64 ou 128×128. Comme le montre [Skodras et coll., 2001], l'utilisation de la séparation en blocs affecte considérablement à la baisse la qualité de l'image.

Par la suite, une DWT est utilisée, ce qui diffère considérablement de son prédécesseur qui utilisait la DCT. Deux types d'ondellettes peuvent être utilisées : celles de Daubechies [Antonini et coll., 1992] ou celles de Le Gall [Le Gall et Tabatabai, 1988]. Ces premières offrent de meilleures performances si une certaine dégradation de l'image est tolérée, alors que ces dernières sont utilisées pour une compression sans-perte. La quantification est soit uniforme, soit TCQ (*Treillis Coded Quantization*). Le codage est basé sur l'algoritme *Embedded Block Coding with Optimal Truncation* (EBCOT) [Taubman, 2000]. Donc, à l'exception de la transformée utilisée, il existe en fait plusieurs similarités avec les approches de Ponomarenko et coll., présentés à la section 2.4.1.

2.4.5 EZW

Le codage progressif par arbres de zéros, mieux connu en tant que «Embedded Zerotree Wavelet encoder» (EZW), a été proposé par Shapiro [Shapiro, 1993] lors de ses expérimentations avec la DWT. C'est un codage progressif au sens où l'information transmise à chaque itération raffine progressivement l'image lorsque décompressée.

L'approche considère d'abord que les images naturelles contiennent un contenu à basse fréquence très important, et que le contenu à haute fréquence est habituellement moindre. Pour la DWT, cela signifie que les coefficients reliés spatialement (exemples à la figure 2.16) sont habituellement plus petits dans les bandes haute fréquence. Ainsi, si une valeur est trouvée à la racine de l'arbre, les valeurs dans les branches sont habituellement plus petites.

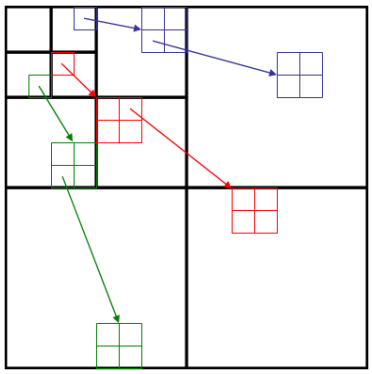


Figure 2.16 – Trois exemples d'arbres liés spatialement pour la DWT

Pour exploiter au maximum cet état de fait, Shapiro utilise des seuils et vérifie si le nombre est supérieur ou inférieur au seuil. S'il est inférieur et que toutes ses feuilles et ses sous-feuilles le sont aussi, il utilise un code spécial qui indique qu'il s'agit d'un arbre de zéros. Cela permet de réduire énormément la quantité d'information transmise pour les images naturelles. Il s'agit également d'une solution élégante pour le codage progressif. Bien que cette technique ne puisse s'appliquer directement pour d'autres transformées, le principe d'arborification de l'information conformément à une caractéristique statistique comme celle observée par Shapiro présente un bon potentiel de codage.

2.4.6 SPIHT

L'approche basée sur le partitionnement d'ensembles dans des arbres hiérarchiques (Set Partionning In Hierarchical Trees) a été proposée par A. Said et W.A. Pearlman [Said et Pearlman, 1996]. Les auteurs proposent une amélioration significative au EZW qui est très performante tant en compression qu'en complexité temporelle. L'amélioration consiste en un traitement plus sophistiqué des arbres tels que représentés à la figure 2.16. Ce traitement amélioré, effectué à l'aide d'un algorithme conçu de façon à ne requérir aucune nouvelle information transmise, permet d'obtenir un meilleur codage sans codage entropique que la compression EZW avec le codage entropique. Cela représente une avancée très particulière car la compression a été améliorée en réduisant la complexité, ce qui va à l'en-

contre des tenets de la théorie de l'information. Toutefois, la pleine performance de SPIHT est atteinte en utilisant un codage entropique de type CABAC avant la transmission.

2.4.7 ADL-SPIHT

Ding et coll. proposent dans [Ding et coll., 2007] une nouvelle façon d'appliquer la DWT dans le contexte du codage SPIHT. Il s'agit de faire la transformée par ondelettes en utilisant une configuration de type *lifting* adaptée à la direction du contenu de l'image. Cette avancée permet d'améliorer légèrement le SPIHT comme le montre le comparatif de la section 2.4.10.

2.4.8 X-W 1999

Xiong et Wu proposent dans [Xiong et Wu, 1999] un nouveau codec basé sur ce qu'ils identifient de meilleur dans le domaine. Leur approche utilise la DWT, la SFQ combinée à la TCQ (TCSFQ) et une combinaison de EZW et de CABAC proposée dans [Wu, 1997]. Cette approche est davantage une intégration judicieuse de technologies existantes que l'apport d'une nouveauté. L'utilisation de la SFQ rend cette approche inutilisable dans le contexte du projet présenté à la section 1.2. Toutefois, elle est un exemple réussi de l'application de la TCQ.

2.4.9 GLBT 16×32

Tran et Nguyen introduisent dans [Tran et Nguyen, 1998] une approche de codage pleinement progressive basée sur l'utilisation de GLBT de la DCT spécialement conçues à cet effet. En définissant un nouveau critère de conception pour les GBLT et en les utilisant avec un codage progressif par arbres de zéros, les résultats obtenus sont supérieurs à SPIHT, parfois jusqu'à 2–3 dB.

2.4.10 Comparatif

Les codecs dont il a été question depuis le début du chapitre forment la fine pointe de la technologie du domaine. Les approches de Ponomarenko et coll. présentées au chapitre 2.4.1 se classent très bien, comme le montre la figure 2.17 et le tableau 2.3. Les résultats sont tirés des publications respectives des codecs à l'exception de JPEG2000 dont la performance a été tirée de [Ponomarenko et coll., 2005]. Puisque les performance sont tirées des publications respectives et que les auteurs n'ont pas tous utilisé

les mêmes images, certaines données sont absentes. La seconde approche de Ponomarenko et coll. (AGU-MHV) obtient les meilleures performances, suivie de leur première approche (AGU) qui compétitionne avec XW-1999 pour la deuxième position en termes de performance moyenne sur l'ensemble des images. La quatrième position appartient à GLBT 16×32. Finalement, JPEG2000, ADL-SPIHT et SPIHT ont des performances assez proches pour la majorité des images.

Il est intéressant de voir que les meilleures performances se partagent entre des codecs qui utilisent la DCT (AGU, AGU-MHV et GLBT 16×32) et des codecs qui utilisent la DWT (JPEG2000, SPIHT, ADL-SPIHT et XW-1999). Aucune des deux transformées ne semble avoir un avantage significatif sur l'autre. Certes, une tendance est observable où la performance des codec utilisant la DWT est à la hausse, c'est-à-dire dans une image avec beaucoup de contrastes (Peppers) : JPEG2000 et SPIHT dépassent AGU. Leur performance est également beaucoup plus basse lorsqu'il s'agit d'une image avec beaucoup de motifs (Barbara). XW-1999 déroge à cette dernière tendance et ce, probablement grâce à la SFQ qui distingue beaucoup le codec des deux autres.

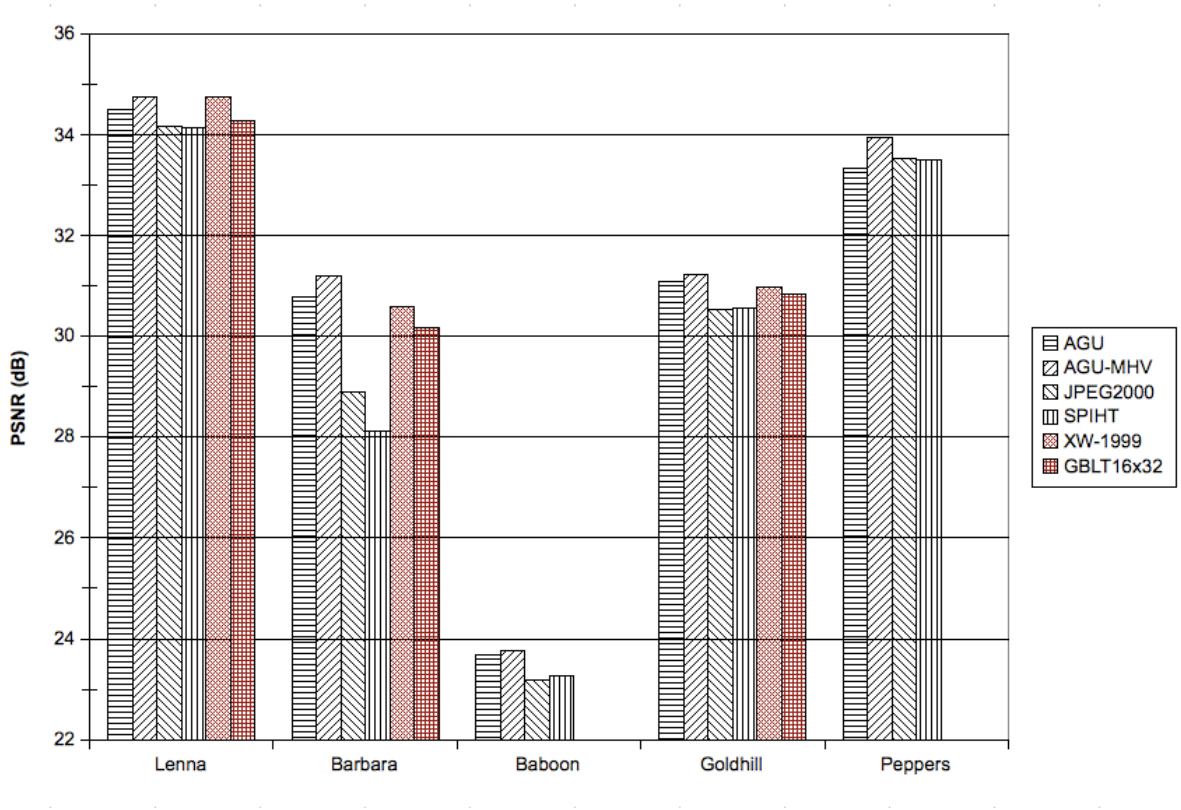


Figure 2.17 – Comparatif de la performance des codec à 0.25 bpp

TABLEAU 2.3 – Comparatif de la performance (PSNR) des codec à 0.25 bpp

	Lenna	Barbara	Baboon	Goldhill	Peppers
AGU	34.50	30.77	23.69	31.09	33.32
AGU-PS	34.75	31.21	23.77	31.22	33.95
JPEG2000	34.15	28.89	23.18	30.53	33.54
SPIHT	34.14	28.13	23.26	30.56	33.51
ADL-SPIHT	34.18	29.28	23.37		
XW-1999	34.76	30.60		30.98	
GBLT16x32	34.27	30.18		30.84	

CHAPITRE 3

AMÉLIORATIONS AU CODEC AGU

La transformation du codec AGU en un codec plus simple et parallélisable passe par la restructuration du codage. En effet, les opérations de transformation et de quantification sont parallélisables puisqu'elles ne s'intéressent qu'à un bloc, les blocs peuvent donc être traités en parallèle. Toutefois, le codage, par sa nature adaptative, ne peut être utilisé. Il existe plusieurs options qui rendraient possible l'exécution en parallèle :

Changement du type de codage. En changeant le type de codage pour une combinaison de codages intrinsèquement non adaptatifs comme le RLE abordé à la section 2.3.1, il est possible de traiter chacun de blocs de façon indépendante. Cette option ne permet toutefois pas de profiter de la grande efficacité du codage arithmétique.

Augmentation de la vitesse d'adaptation. Quelques blocs sont requis pour une adaptation suffisante du modeleur de contexte utilisé dans le codec AGU. Cela le rend difficilement utilisable de façon indépendante sur chacun des blocs. En augmentant la vitesse d'adaptation significativement, il est possible que le modeleur de contexte atteigne une bonne performance sur chacun des blocs. La faible quantité de données pour réaliser l'adaptation pose toutefois problème.

Pré-adaptation du modeleur de contexte. Il est possible de réaliser un modeleur de contexte général qui fonctionne correctement sur chacune des images, quoique moins bien qu'une adaptation spécifique à chacune de celles-ci. Cette stratégie repose toutefois sur l'hypothèse que les images naturelles ont beaucoup de caractéristiques communes dans le domaine de la DCT.

Parmi ces options, la pré-adaptation du modeleur de contexte semble la plus prometteuse. Puisque l'opération de modelage de contexte peut être considérée comme une opération de classification et qu'un entraînement est requis, il semble tout naturel d'étudier comment utiliser un réseau de neurones pour réaliser le modeleur de contexte.

L'implémentation du codec AGU est présentée à la section 3.1. Ensuite, pour arriver à réaliser le réseau de neurones, les outils développés sont présentés à la section 3.2. De

plus, afin de mieux comprendre les résultats observés sur les images de tests (présentées à l'annexe A), la section 3.3 présente des observations statistiques sur celles-ci.

3.1 Implémentation du codec AGU

L'implémentation du codec original est, d'une part, une bonne façon de comprendre les techniques utilisées et, d'autre part, la première étape en vue de son amélioration. Cette implémentation est réalisée à partir de [Ponomarenko et coll., 2005]. Ponomarenko et coll. fournissent l'exécutable du codec AGU gratuitement à des fins de recherche. Cet exécutable est particulièrement utile à l'implémentation du codec car il permet de comparer les performances avec ou sans l'étape finale de filtrage, à des niveaux de qualité plus nombreux que les résultats de l'article. Les modules réalisés correspondent aux blocs présentés à la figures 2.12, c'est à dire la division en blocs de 32×32, la DCT, la quantification, le traitement en plans de bits, la classification, le codage arithmétique binaire et le filtrage visant à éliminer les artefacts de bloc.

Les modalités d'implémentations sont :

Division en blocs. La division en bloc pour AGU est une division régulière en blocs de 32×32. Il a été choisi de conserver les données dans un tableau de la taille de l'image mais de l'accompagner d'une classe identifiant le type de division en bloc et ses caractéristiques. La classe identifiant le type de division doit hériter de la classe *block-scheme*, dans le cas de la division en bloc régulière. La classe se nomme *block-scheme-regular*. Chaque *block-scheme* définit des fonctions de transformations de coordonnées pour passer du domaine en blocs au domaine absolu du tableau de données. Le tableau de données et le type de division sont enveloppés dans une classe nommée *b-array* (pour block-array) pour en faciliter la manipulation.

DCT. La DCT et son inverse sont réalisées par la bibliothèque "Fastest Fourier Transform in the West" (www.fftw.org). L'utilisation d'une bibliothèque pour la DCT permet de réduire considérablement le temps d'implémentation d'AGU. Cette bibliothèque est testée et mature et son utilisation permet donc d'éliminer une source de problèmes.

La bibliothèque FFTW est d'abord conçue pour calculer des transformées de Fourier. Celles-ci sont par la suite converties en DCT (toujours par la FFTW). Toutefois, il existe plusieurs versions de la DCT. Qui plus est, le calcul d'une DCT suivie de la IDCT correspondante à l'aide de FFTW ne permet pas de revenir au signal original. Un certain schémas de facteur doit être ajouté pour que l'opération soit réversible.

En l'absence d'information sur le type de DCT utilisé dans AGU et le schémas des facteurs, la reproduction des résultats est difficile.

Un grand nombre de tests révèlent que la DCT-II est utilisée pour la transformation et la DCT-III pour l'inverse et ce, dans un schémas demi-normé (*half-scaled*). Cela se traduit, dans le cas de blocs de 32×32, par un facteur de 128 pour la composante DC, un facteur de $64\sqrt{2}$ pour le reste de la première ligne et de la première colonne et un facteur de 64 pour le reste du bloc (voir figure 3.1).

	0	1	2	3		31
0	128	90,5	90,5	90,5	...	90,5
1	90,5	64	64	64	...	64
2	90,5	64	64	64	...	64
3	90,5	64	64	64	...	64
	...	...	...	...	...	64
31	90,5	64	64	64	64	64

Figure 3.1 – Schémas demi-normé pour un bloc 32×32

Traitement en plans de bits. Puisque le codec AGU est basé sur un codage entropique arithmétique binaire, les données sont traitées un bit à la fois. L'article mentionne également que les données sont traités du plan de bits le plus significatif au plan de bits le moins significatif. Cela pourrait être le plan de bits global ou le plan de bits d'un bloc. Puisque la transformée est effectuée un bloc à la fois, il est facile de croire que l'encodage est également effectué un bloc à la fois. Toutefois, les tests adéquats révèlent que bien que la DCT soit effectué par bloc, le codage ne l'est pas. Le parcours des bits doit se faire un plan de bits global à la fois.

Classification. L'article se concentre davantage sur la classification des bits. L'implémentation s'en trouve donc facilitée pour cette section. En contrepartie, une erreur est présente dans l'arbre de classification présenté à la figure 3 de [Ponomarenko et coll., 2005]. En effet, la condition C13 est évaluée alors que la condition C7 est

fausse. Or, la condition C7 est fausse uniquement si les 4 bits voisins immédiats de même importance sont tous zéro et la condition C13 s'intéresse à la somme de ces même 4 voisins. L'inversion du vrai et du faux pour la condition C7 permet d'atteindre les performance de l'exécutable fourni.

Codage arithmétique binaire. Le codage arithmétique binaire est une technique complexe mais qui comporte peu de configuration. L'unique configuration se situe au niveau de la précision pour les codeurs à précision finie. Le codeur utilisé pour la reproduction des performances d'AGU utilise une précision de 22 bits.

La somme des inconnus, des ambiguïtés et des erreurs de [Ponomarenko et coll., 2005] a rendu l'implémentation beaucoup plus difficile que prévu. Chaque facteur est assez simple lorsqu'il est indépendant des autres. Leur combinaison synergique rend la tâche beaucoup plus complexe. Les résultats obtenus sont les suivants :

- L'aspect visuel de l'image permet de vérifier que les dégradations sont les mêmes pour les deux implémentations. L'image de test *Lenna* couplée à un pas de quantification (QS) de 40 ainsi que l'image *Baboon* couplée à un pas de quantification de 80 servent de validation. Cette validation est sommaire mais adéquate en considérant le fait qu'il ne s'agit pas du sujet principal du projet de recherche. Les deux paires d'images présentées aux figures 3.2 et 3.3 sont identiques au premier coup d'œil. Toutefois, en les examinant avec beaucoup de minutie, il est possible de remarquer certains pixels qui diffèrent. Cela s'explique par une différence au niveau de la transformée pour le filtrage. Le codec original utilise une version entière de la DCT alors que la réimplémentation utilise une version en virgule flottante.

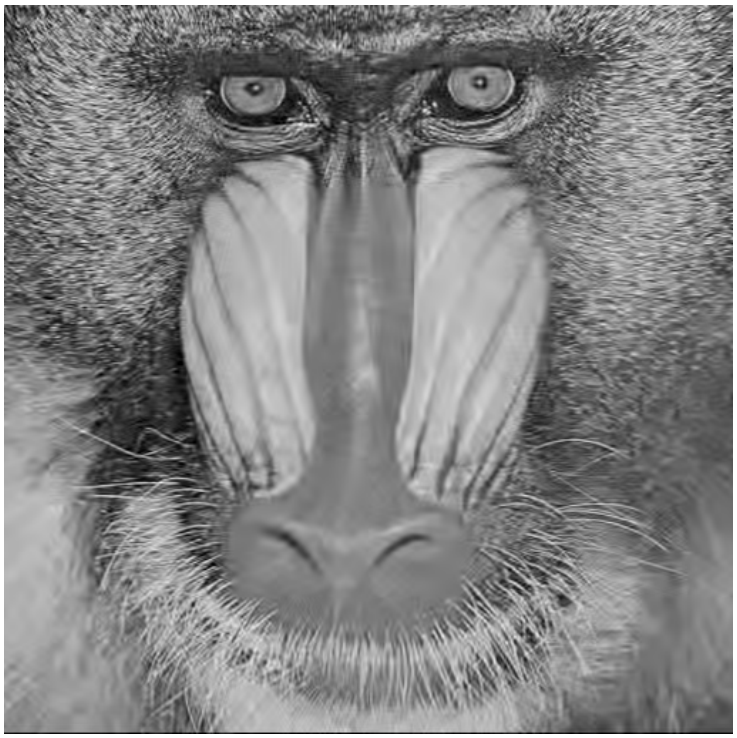
- Comme le montre la figure 3.4, la performance de la réimplémentation est presque identique à la performance du codec original. Les sources précises des différences sont difficiles à identifier. Les plus probables sont des différences en ce qui a trait au surdébit à propos duquel aucun détail n'est donné dans l'article, et des différences d'implémentation du codage arithmétiques. Dans tous les cas, les différences sont considérées comme mineures et ayant peu d'importance.

La réimplémentation du codec AGU est donc réussie. Les quelques différences obtenues ne sont pas significatives.

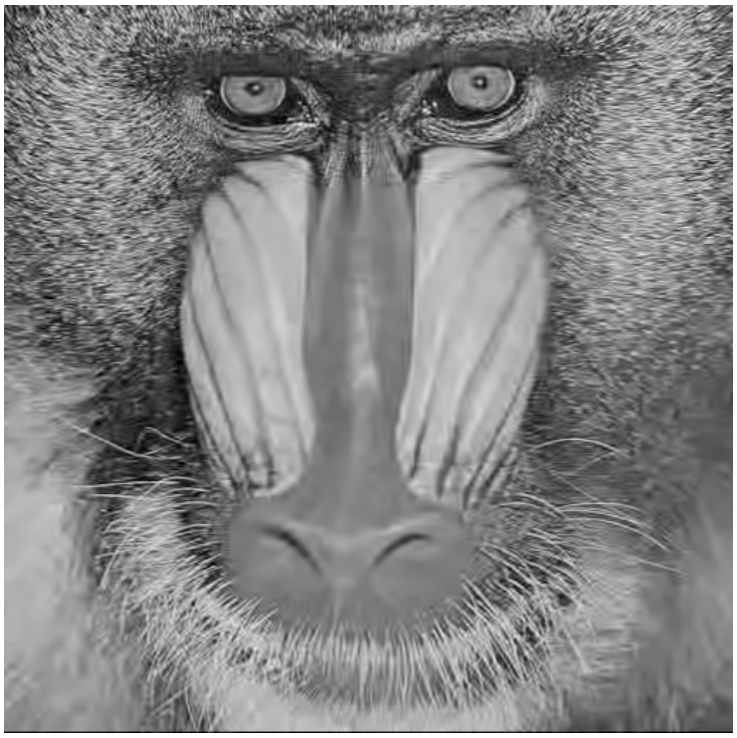
(a) AGU - Original

(b) Réimplémentation

Figure 3.2 – Comparaison des implémentations pour Lenna (QS=40)

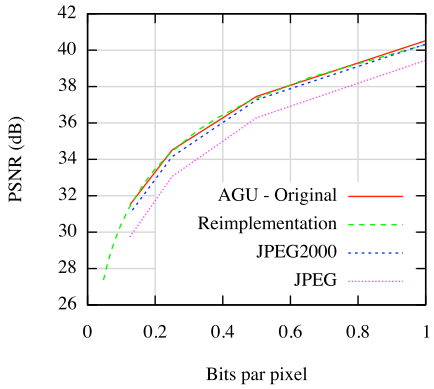


(a) AGU - Original

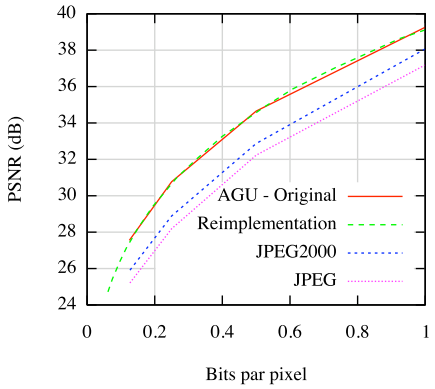


(b) Réimplémentation

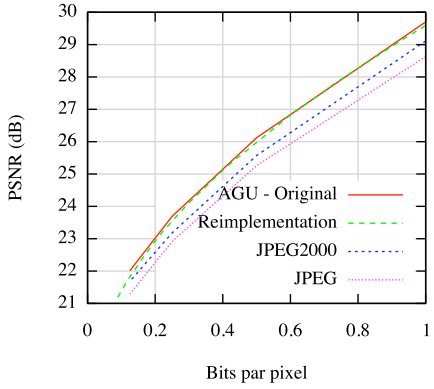
Figure 3.3 – Comparaison des implémentations pour Baboon (QS=80)



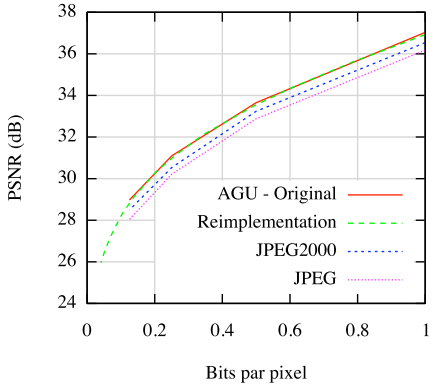
(a) Lenna



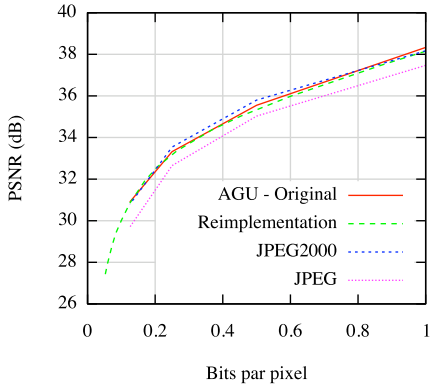
(b) Barbara



(c) Baboon



(d) Goldhill



(e) Peppers

Figure 3.4 – Comparaison des implémentations pour les cinq images

3.2 Outils développés

En vue de l'amélioration du codec, il est préférable de se doter d'outils qui permettront d'identifier les pistes d'amélioration et de les tester rapidement. Plusieurs avenues sont envisageables : applications graphiques, outils console et environnements de script.

Les applications graphiques ont l'avantage d'être facile à utiliser, particulièrement pour les non-experts. Cette facilité d'utilisation et d'apprentissage s'accompagne d'un temps de développement de beaucoup supérieur. Il est également important de noter qu'en vertu du nombre restreint de chercheurs ayant à interagir avec les outils, il n'est pas rentable, dans le cadre du projet courant, d'investir une somme importante de temps à rendre les outils très conviviaux.

Les outils console permettent de réduire de beaucoup le temps de développement par rapport aux applications graphiques. Pour les experts, ils sont habituellement aussi faciles à utiliser avec l'avantage majeur de faciliter le traitement en lot (*batch processing*). Chaque modification à l'outil nécessite toutefois de compiler à nouveau et de recharger les données. Dans le contexte présent de modifications et de tests nombreux, cela ralentit considérablement le développement.

La troisième option considérée est l'utilisation d'un environnement de scripts. Celui-ci à l'avantage de ne pas nécessiter de compilation ou de chargement de données. Le traitement en lot est facilité par rapport aux outils console. Il s'agit d'un environnement idéal pour le développement de prototypes. Toutefois, l'exécution est habituellement plus lente en raison de la nature interprétée (non-compilée) de la majorité des environnement de script.

Ayant accès à une équipe très réduite de développement et à bon nombre d'ordinateurs grâce aux grappes de calcul de Mammouth, la troisième option est celle qui convient le plus au projet en cours.

3.2.1 Choix de l'environnement de scripts

Les environnement Matlab et SLIME ont été considérés. Bien que d'autres environnements de scripts existent, ils ont été exclus en raison du manque de familiarité afin de réduire le temps passé à apprendre l'environnement.

Matlab est un language et un environnement de scripts propriétaire développé par MathWorks (www.mathworks.com). Il a l'avantage d'inclure plusieurs outils mathématiques de traitement d'image. Comme pour plusieurs languages de scripts, Matlab est assez limité.

L'absence d'espaces de nommage (*namespaces*), de macros et d'autres fonctionnalités de langages de programmation de bonne qualité rend difficile la production de modules de haute qualité.

SLIME est un environnement de développement pour Lisp basée sur l'éditeur Emacs. Il fournit un environnement légèrement moins convivial que Matlab. En contrepartie, Lisp, ou plus précisément Common Lisp, est un langage beaucoup plus puissant que Matlab car il possède la plupart des caractéristiques des langages de bonne qualité. Puisque la frontière entre le code et les données est très mince en Lisp, cela facilite la création de prototypes par la programmation. SLIME et Common Lisp ont également l'avantage d'être disponibles gratuitement, et ce, pour plusieurs plateformes.

Malgré la plus grande familiarité de l'équipe avec Matlab, SLIME a été choisi pour les avantages fournis par le langage sous-jacent ainsi que pour la facilité avec laquelle le code pourra être exécuté sur Mammouth.

Étant donné le contexte de réalisation de prototype, la rapidité d'implémentation prime sur la performance. Beaucoup de choix de conception sont basés sur cette priorité mais sont réalisés pour la plupart en encapsulant de telle façon qu'un changement d'implémentation soit possible. Cette stratégie permet à la fois d'obtenir des résultats très rapidement et de permettre la postérité des modules.

3.2.2 Modules fondamentaux

L'amélioration du modelage du contexte pour le codec étudié implique un certain nombre de modules fondamentaux. Puisque le codage arithmétique binaire est conservé, les données à encoder demeurent sous la forme d'un volume de bits. Il s'agit donc de construire des outils permettant d'étudier ce volume de bits et de créer un modelage de contexte atteignant les objectifs énoncés à la section 1.4.

La plupart de ces outils sont regroupés dans l'espace de nommage *bitvolume*. Cet espace de nommage est l'espace de développement principal où les outils sont développés et certains sont extraits dans leur propre espace de nommage une fois matures, et importés dans l'espace de nommage principal.

3.2.3 Volume de bits

Un volume de bits est contenu dans un tableau d'entiers. Celui-ci est accompagné d'un descripteur de division en blocs appelé *block-scheme*. Ce couple forme une structure *b-array* tel que représenté à la figure 3.5.

La division en blocs est fondamentalement un changement de domaine. Par exemple, une division en blocs régulière fait passer le système de coordonnées du volume de bits de (x, y, z) à (i, j, x, y, z), en considérant que x est la position horizontale du bit, y sa position verticale et z sa position de profondeur. Dans le deuxième système, i et j sont les coordonnées du bloc, x et y deviennent la position locale du bit dans le bloc et z garde la même signification. Le *block-scheme* doit donc contenir les informations requises pour faire les changement du système absolu dans lequel des données sont stockées, au système de coordonnées du type de division. C'est ce qui est illustré à la figure 3.6 : *block-scheme-nil* représente le cas ou aucune division en bloc n'est réalisée, *block-scheme-regular* représente la division régulière (AGU par exemple) et *block-scheme-mhv* représente le cas d'une division horizontale verticale modifiée (AGU-MHV par exemple).

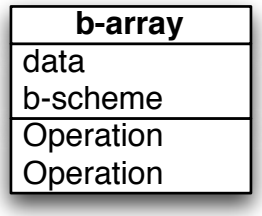


Figure 3.5 – Structure b-array

L'accès aux bits est faite grâces aux méthodes *baref* et *bvref* qui accèdent respectivement à une valeur et un bit ainsi qu'aux méthodes dérivées. Ces méthodes se spécialisent sur la classe du *block-scheme* d'un *b-array* et sur la classe de la coordonnée reçue. L'accès est réalisé par défault en utilisant la méthode *absolute<-custom* définie pour le type de division en blocs, mais il est également possible de spécialiser les méthodes avec une conversion plus efficace pour des questions de rapidité d'exécution, comme c'est le cas pour la division en blocs régulière.

Ces coordonnées sont contenues dans des structures appropriées. Les coordonnées utilisés pour un *block-scheme-regular* sont contenues dans une structure de type *regular-coordinates*, abbréviée *rc*.

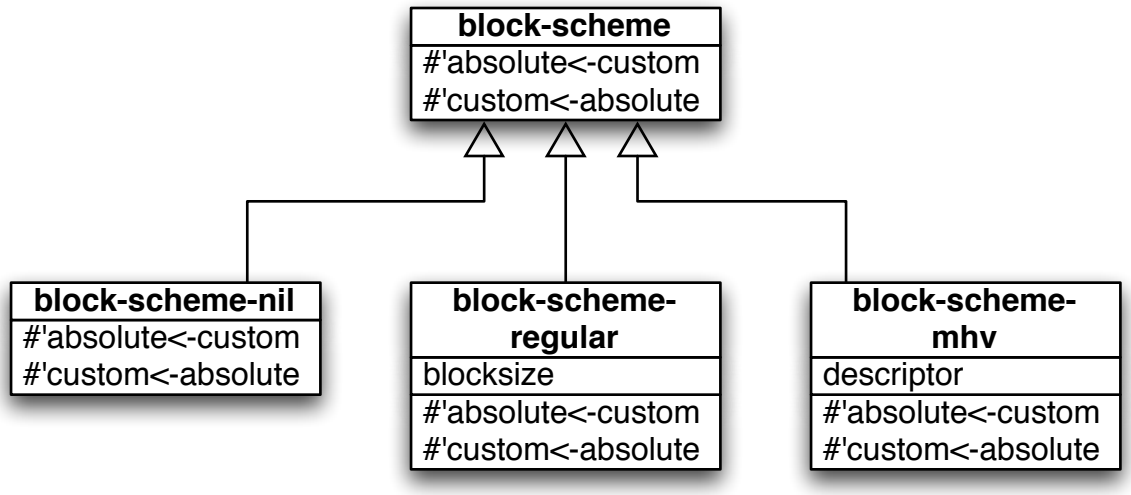


Figure 3.6 – Classes représentant des types de division en blocs

Ces éléments permettent de représenter un volume de bits et d'y accéder facilement dans le système de coordonnées approprié à sa division en blocs ou dans le système de coordonnées absolu *absolute-coordinates*.

3.2.4 Parcours d'un volume de bits

L'ordre dans lequel un volume de bits est parcouru est très important, car seulement les bits encodés précédemment sont disponibles lors de l'encodage d'un bit. Un type de parcours doit connaître les dimensions du volume à parcourir. Il fournit la coordonnée de départ et la fait progresser. Quelques-unes de ces classes sont représentées à la figure 3.7.

La macro *path-loop* permet de faciliter l'utilisation d'un parcours en établissant tout ce qui est requis dans l'environnement lexical. Elle permet également de cacher davantage les détails d'implémentation pour l'utilisateur.

3.2.5 Prédicteurs

L'amélioration du modelage de contexte vise à améliorer la prédiction pour le bit courant. Un prédicteur doit connaître les bits voisins du bit à coder et sa position. Le nombre et la position des voisins dépend du prédicteur, du type de division en bloc et de l'ordre de parcours. Par exemple, deux bits qui sont aux positions horizontales 0 et 32 sont voisins en i pour une division en blocs régulière de taille 32. Un prédicteur reçoit donc un *b-array* contenant les bits déjà encodés et la position du bit pour lequel il doit fournir un

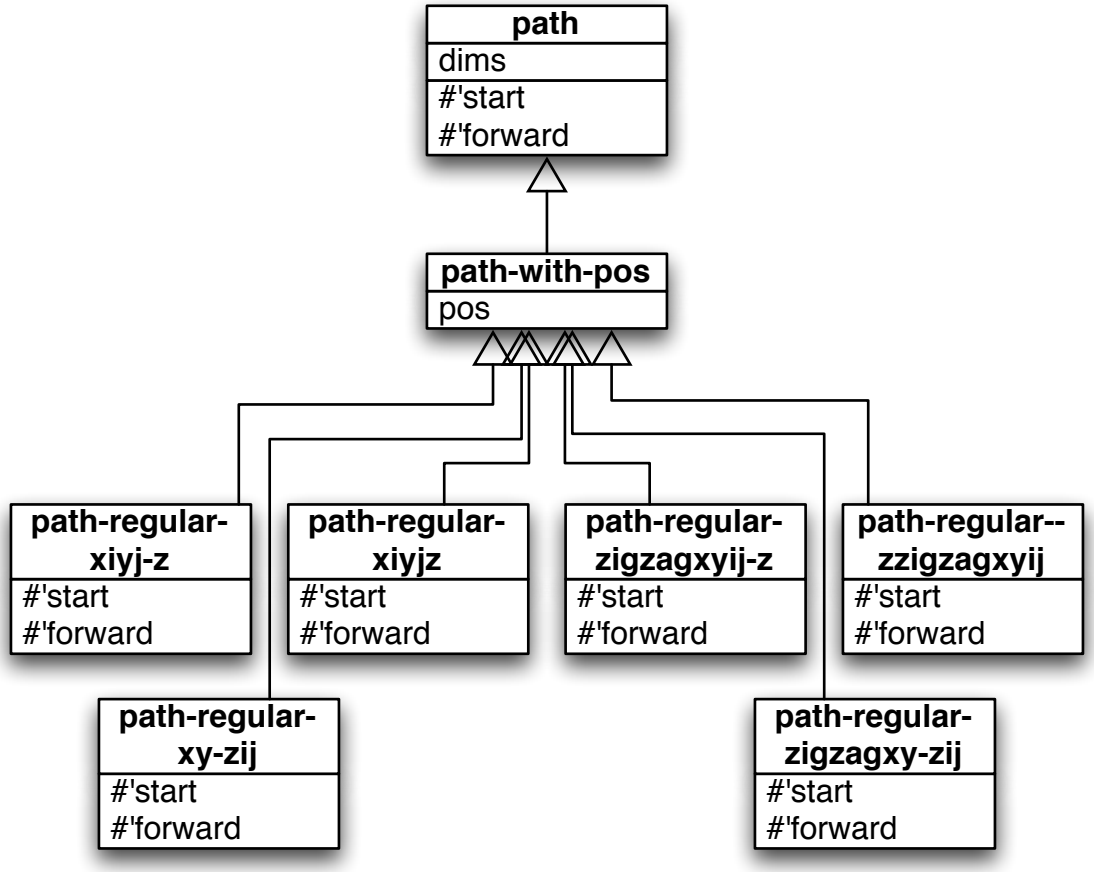


Figure 3.7 – Classes représentant des type de parcours de bits

prédiction, tel que représenté à la figure 3.8. En mode entraînement, le prédicteur reçoit aussi la valeur du bit et ne fournit rien : cette étape est appelée *update*. La figure 3.9 montre quelques classes de prédicteurs.

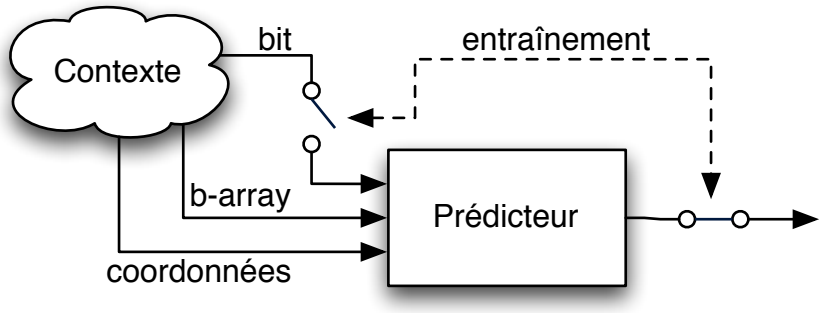


Figure 3.8 – Configuration pour l'utilisation d'un prédicteur

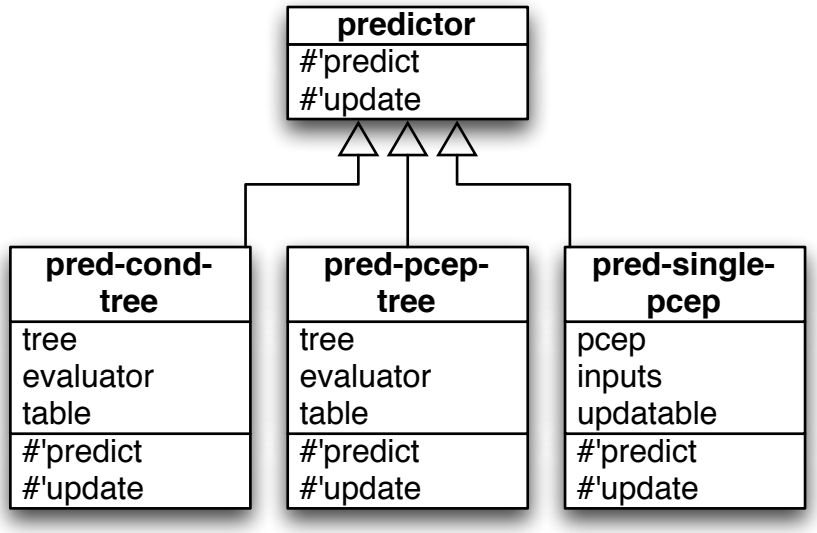


Figure 3.9 – Classes de prédicteurs

Arbre de conditions

Le prédicteur arbre de conditions (*pred-cond-tree*) est un prédicteur comme ceux présents dans [Ponomarenko et coll., 2005, 2007]. Le prédicteur est composé de l'arbre, d'une fonction d'évaluation de l'arbre et une table de hachage. L'arbre est près du code Lisp mais sous une forme plus pratique pour les modifications. La fonction d'évaluation est créée en transformant l'arbre en code Lisp pour ensuite le compiler. La table de hachage fait le lien entre la forme textuelle des catégories et leurs instances.

Puisque l'arbre est sous une forme proche du code Lisp, il est facile pour un algorithme de le modifier. Cela rend possible la création d'un algorithme d'optimisation d'arbre de

conditions. Le contexte de recherche du meilleur arbre de conditions rend difficile le choix d'une heuristique, et le grand nombre de conditions possibles rend les recherches largeur d'abord très lente. Une recherche utilisant un algorithme glouton (*greedy algorithm*) se prête bien à ce type de recherche.

L'algorithme glouton est l'algorithme choisi pour la recherche d'arbre de conditions *greedy-tree*. L'algorithme détermine d'abord les modifications possibles et pertinentes à l'arbre en prenant soin, par exemple, d'éliminer les erreurs de logique comme l'évaluation d'une condition dans une branche dans laquelle elle a déjà été évaluée. Une fois les modifications possibles identifiées, l'algorithme calcule l'amélioration de compression entraînée par chacune d'elles. La meilleure modification est appliquée et soustraite à la liste des modifications possibles ainsi que toutes celles qu'elle invalide. L'algorithme réitère ainsi pour ajouter le nombre de conditions qui lui avait été demandé. La réutilisation des modifications de l'itération précédente permet d'économiser le temps requis pour calculer l'amélioration encourue. Cet algorithme est illustré à la figure 3.10.

Conditions

Une amélioration telle que considérée par l'algorithme *greedy-tree* est le remplacement d'une feuille de l'arbre par une condition et deux nouvelles feuilles. Cela implique que les améliorations possibles dépendent des conditions possibles. Les conditions possibles sont fournies en entrée à l'algorithme. Puisque leur nombre est potentiellement très grand, des outils pour en générer selon certains patrons permettent de faciliter la tâche.

Un outil permettant de générer une condition en utilisant une forme géométrique est *cond-bit-shape*. Cet outil crée des conditions qui effectuent un ou-inclusif sur plusieurs bits qui sont disposés selon une forme géométrique telles que celles présentées à la figure 3.11. Quelques autres formes géométriques dérivées de celles présentés à la figure 3.11 sont également mis en oeuvre.

Un second outil nommé *many-cond-bit-shapes* permet de générer plusieurs conditions en faisant varier les formes et les paramètres.

Ces deux outils combinés à d'autres de moindre importance et à quelques conditions plus spécialisées créées manuellement permettent d'obtenir des banques de conditions très nombreuses pour l'exécution de l'algorithme *greedy-tree*.

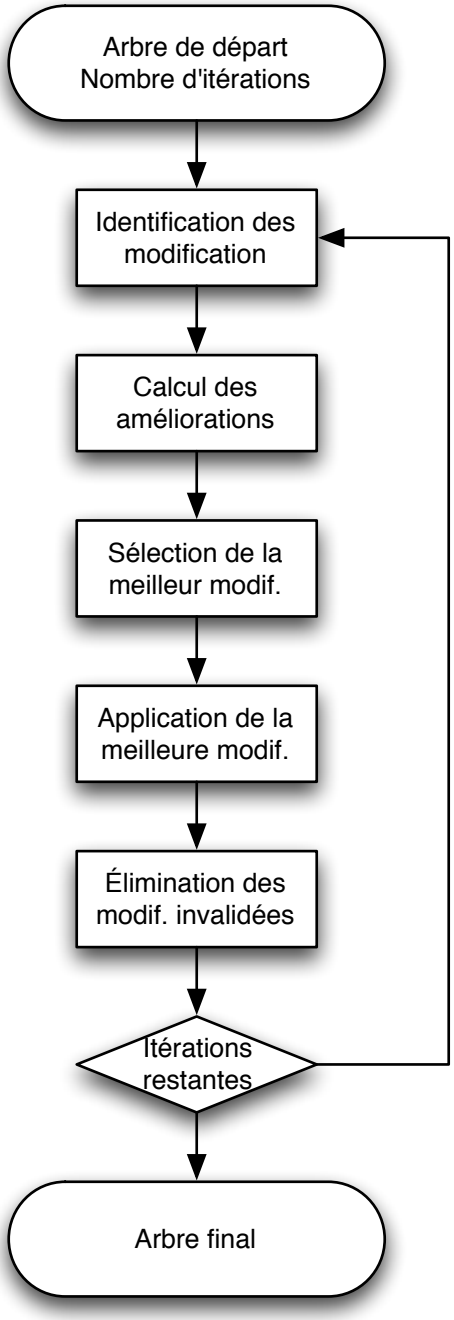
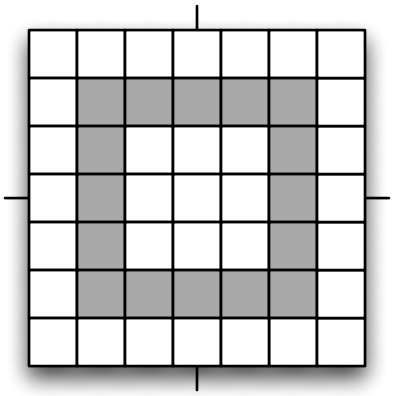
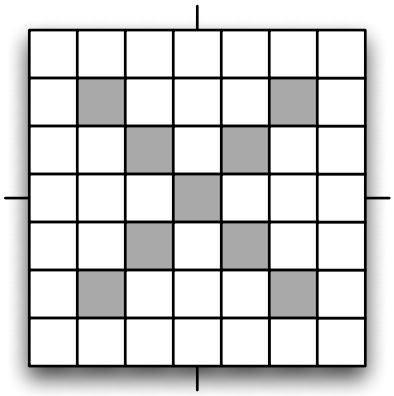


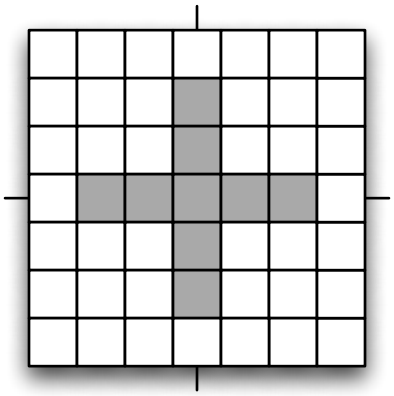
Figure 3.10 – Algorithme de recherche d'arbre



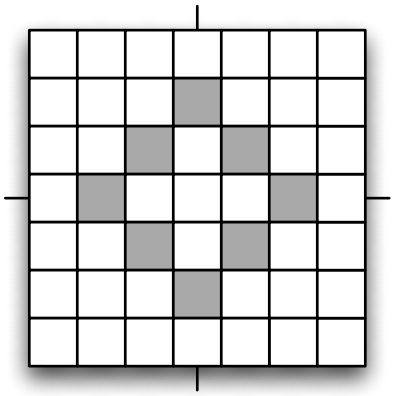
(a) *:square*



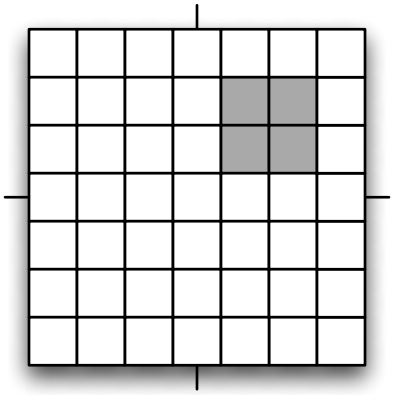
(b) *:x*



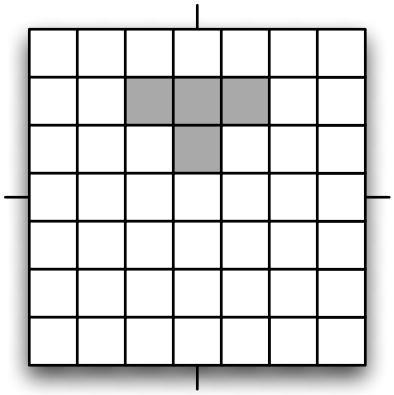
(c) *:cross*



(d) *:diamond*



(e) *:up-right*



(f) *:up-v*

Figure 3.11 – Formes principales pour *cond-bit-shape*

Perceptron

Une seconde stratégie pour le modelage du contexte est l'utilisation d'un perceptron ou d'un réseau de neurones. Les entrées du réseau de neurones peuvent être des conditions basées sur le contexte ou des éléments direct du contexte comme des bits individuels ou les coordonnées du bit à encoder. Un exemple de perceptron à fonction d'activation linéaire bornée utilisant des conditions comme entrées est illustré à la figure 3.12.

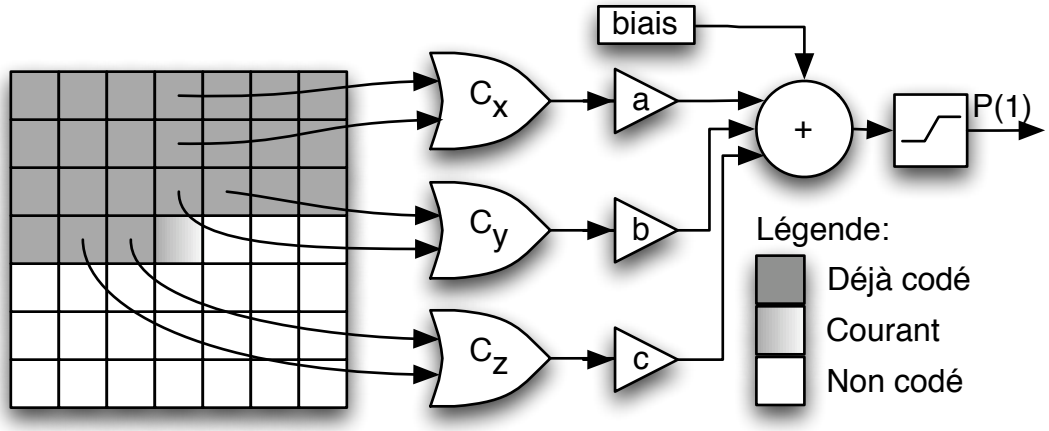


Figure 3.12 – Exemple de configuration d'un prédicteur basé sur un perceptron

La bibliothèque *Fast Artificial Neural Network Library* (leenissen.dk/fann) est utilisée pour toutes les opérations relatives au réseau de neurones. Cette bibliothèque est mature et propose un bon nombre de configurations qui permettent d'obtenir un réseau de neurones adapté aux besoin du projet en très peu de temps.

Les perceptrons et réseau de neurones utilisés pour le modelage du contexte peuvent être opérés avec ou sans entraînement durant le codage. Dans le second cas, la méthode *update* du prédicteur n'effectuera aucun travail. Bien que cela soit illogique dans le cas d'un arbre de conditions, ce n'est pas le cas pour un réseau de neurones qui a été entraîné avant l'encodage.

Le prédicteur de type perceptron (*pred-single-pcep*) contient le perceptron lui-même (*pcep*), la liste de fonctions qui permettent d'obtenir les entrées du perceptron (*inputs*) et un drapeau qui indique si le perceptron doit être entraîné pendant le codage (*updatable*). Ces éléments sont illustrés à la figure 3.9.

Arbre de perceptrons

La combinaison de la structure en arbre et des réseaux de neurones est une troisième avenue pour l'amélioration du modelage de contexte. Celle-ci permet une méthode rigoureuse pour la formation de conditions. En effet, puisque les réseau de neurones viennent remplacer les conditions et que plusieurs méthodes existent pour l'entraînement de ces réseaux, la sélection d'une condition parmi une liste prédéfinie dans *greedy-tree* est remplacée par la formation d'une condition par l'entraînement d'un réseau de neurones. Le prédicteur basé sur ce type d'arbre est nommé *pred-pcep-tree-cat*.

L'algorithme présenté à la figure 3.10 peut être réutilisé pour la formation de l'arbre de perceptron. Une modification est le remplacement d'une feuille par un réseau de neurones et un seuil qui permet de former une condition accompagnés de deux feuilles contenant des catégories. Cet algorithme est contenu dans la fonction *pcep-tree-cat*.

Puisque les réseaux de neurones présents dans l'arbre cherche à distinguer entre les zéros et les uns du volume de bits, ils ont une sortie qui peut être interprétée comme la probabilité que le bit soit 1. Il est donc possible de retirer les feuilles et d'utiliser les réseaux devenus feuilles pour obtenir la prédiction. Ce type de prédicteur est nommé *pred-pcep-tree*.

Il est encore une fois possible de réutiliser l'algorithme de la figure 3.10 pour trouver un arbre de réseaux de neurones. Les modifications sont l'ajout d'un seuil à un réseau de neurones feuille et de deux nouveau réseaux feuilles. Cet algorithme est nommé *pcep-tree*.

3.2.6 Outils auxiliaires

Bien que le modelage du contexte soit étudié, certains autres outils et modules auxiliaires sont nécessaires pour valider son bon fonctionnement et comparer ses performances aux autres méthodes.

Codage

Les outils de la catégorie codage sont des outils qui opèrent dans un contexte d'encodage ou de décodage d'un *b-array*, c'est-à-dire dans un contexte où seuls les bits qui ont déjà été parcourus sont disponibles. Ce groupe inclut la préparation de données d'entraînement pour un réseau de neurones, la mesure de la performance d'un prédicteur sur un groupe d'images, le calcul de la valeur médiane d'une caractéristiques pour un groupe d'images et d'autres opérations de moindre importance.

Codage arithmétique

Deux codeurs arithmétiques sont disponibles. Le premier est un codeur arithmétique binaire à précision finie. Il a l'avantage d'être plus rapide tout en offrant une précision suffisante. Le second est un codeur arithmétique à précision infinie. Il permet d'encoder une source ayant un alphabet de taille arbitraire et possède une précision infinie. Sa vitesse est toutefois inférieure.

3.3 Observations statistiques sur les images de test

Le modelage du contexte en jeu lors du codage entropique requiert une bonne connaissance des statistiques de la source. La source qui nous intéresse est la DCT en blocs pour des images naturelles. Cette source peut être examinée de façon absolue, en regardant par exemple la probabilité de trouver un 1 au premier bit du volume de bits ou la probabilité d'un 1 au plan de bit le moins significatif du volume de bits. Elle peut également être examinée de façon relative en s'intéressant à la probabilité d'un 1 pour un bit qui a un voisin immédiat au plan de bit supérieur. Ce chapitre s'intéresse à ces deux types d'observation.

3.3.1 Statistiques absolues

Les observations absolues s'intéressent aux statistiques qui ne dépendent pas des bits entourant un bit. Parmi les statistiques absolues, on note les probabilités globales de 1 ou de 0, ainsi que leurs probabilités selon leur position dans le blocs. La positition d'un bit dans un bloc est exprimée selon x, y et z. La coordonnée en z correspond au plan de bits, $z = 0$ étant le plan de bits le moins significatif. Les coordonnées x et y correspondent à la position du coefficients de la DCT dans un bloc. La position décrite par $x = y = 0$ correspond à la composante de fréquence 0 du bloc.

Statistiques globales

La statistique la plus simple est la probabilité globale de 1 (et donc implicitement la probabilité de 0 puisqu'il s'agit de codage arithmétique binaire). La figure 3.13 montre la probabilité de 1 pour chacune des images. Cette probabilité se situe entre 1 et 1,5%.

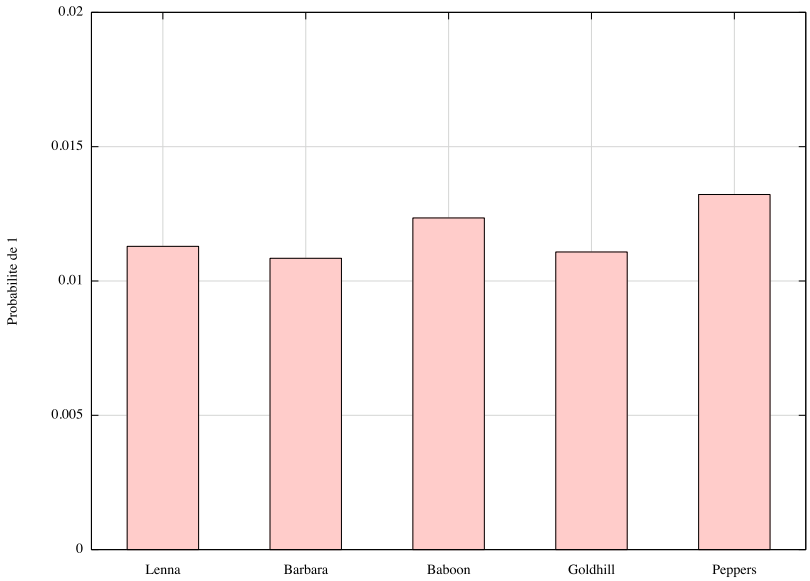


Figure 3.13 – Probabilité globale de 1 pour chacune des images

Statistiques par plan de bits

Les coefficients de grandes valeurs étant moins nombreux que les coefficient de valeur faible, la probabilité de 1 est plus grande en allant vers les plans de bit les plus faibles. Les figures 3.14 et 3.15 montrent cet étant de fait.

La remonté suivie de la chute brusque est due au composantes DC des blocs qui sont typiquement beaucoup plus hautes que le reste des valeurs. Cette constatation est confirmée par les figures 3.16 et 3.17.

Les figures 3.16 et 3.17 montrent une certaine chute de la probabilité de 1 pour le dernier plan de bits pour la plupart des images. Cela est dû au fait que seul un petit nombre de coefficients atteignent une telle amplitude.

Statistiques selon x et y

En considérant tous les plans de bits de tous les blocs d'une image, on obtient les probabilités illustrées à la figure 3.18. Les pics à très haute fréquence pour Peppers rendent le codage de cette image plus difficile car ces bits sont difficiles à prévoir.

Il est également possible de s'intéresser à la probabilité de 1 selon $x + y$ ou selon la norme du vecteur formé par ces coordonnées. C'est ce qui est illustré aux figures 3.19 et 3.20. La forme générale est celle d'une décroissante à l'exception de Peppers qui connaît une remontée significative dans la section des hautes fréquences.

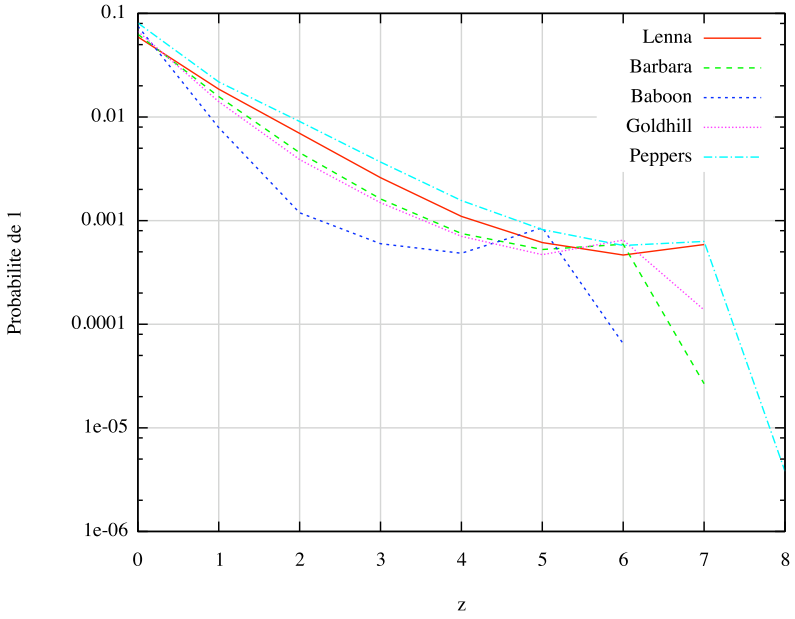


Figure 3.14 – Probabilité de 1 selon z

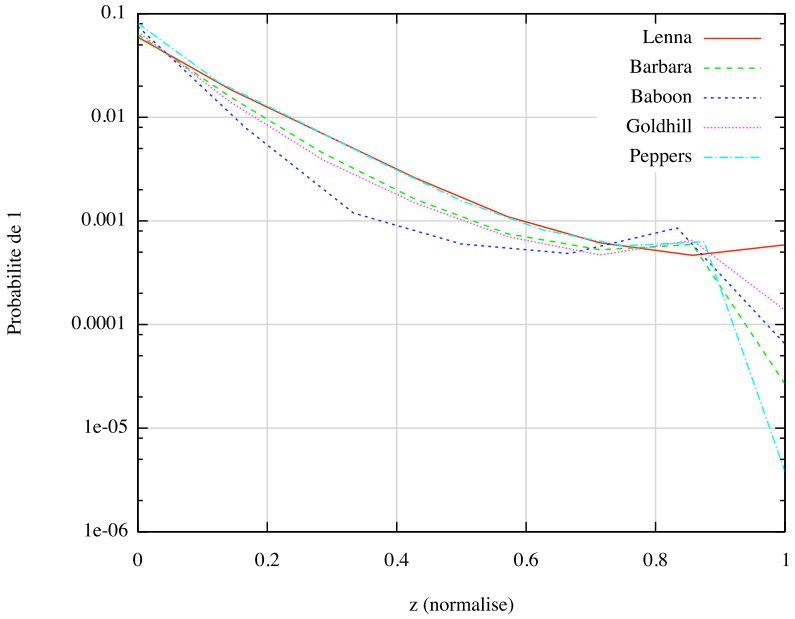


Figure 3.15 – Probabilité de 1 selon z nomalisé

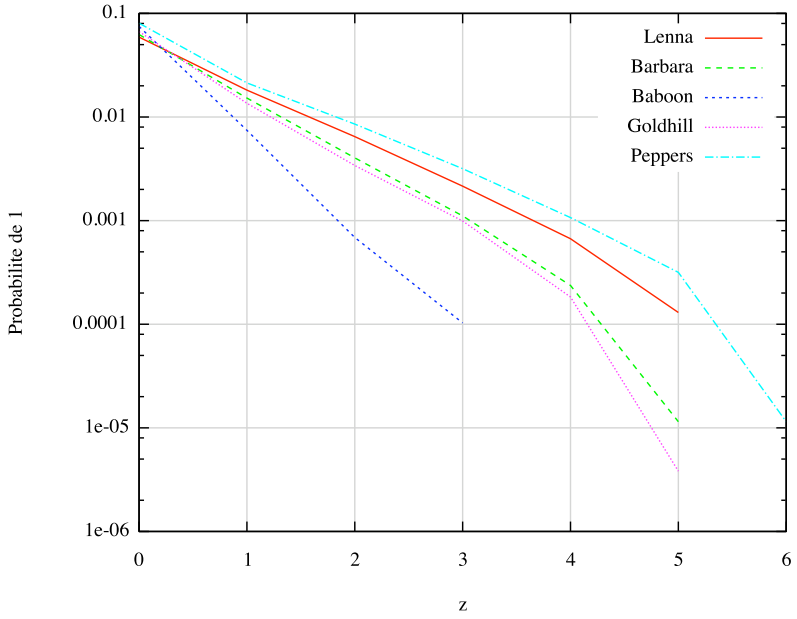


Figure 3.16 – Probabilité de 1 selon z en excluant $x = y = 0$

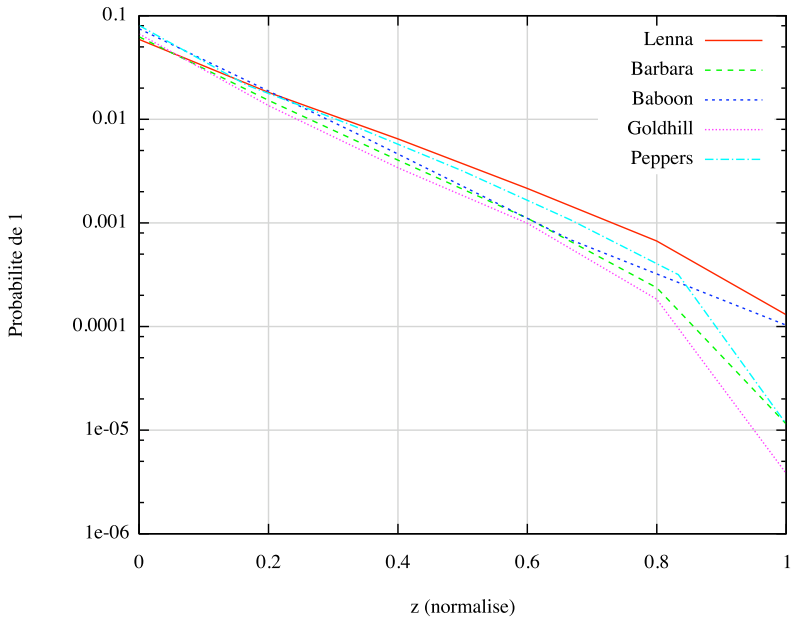
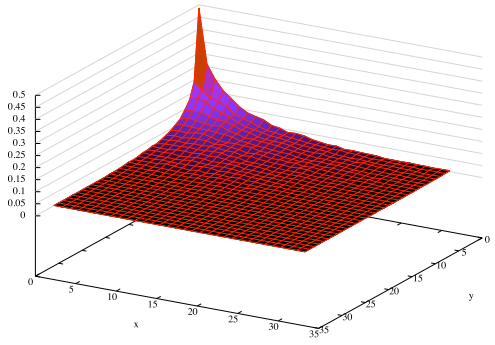
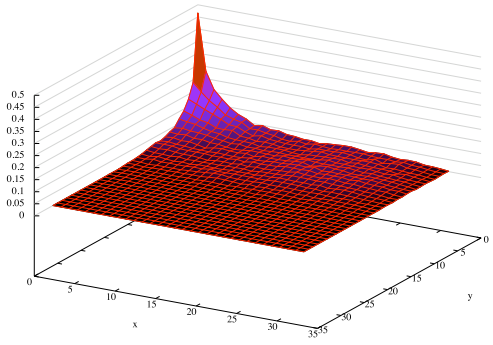


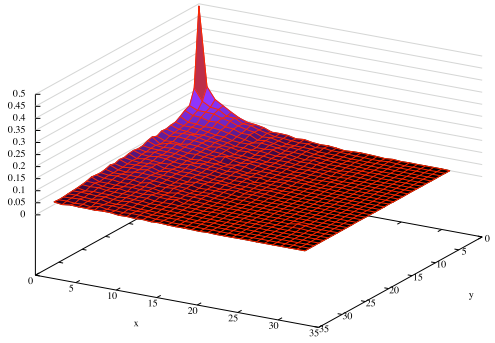
Figure 3.17 – Probabilité de 1 selon z nomalisé en excluant $x = y = 0$



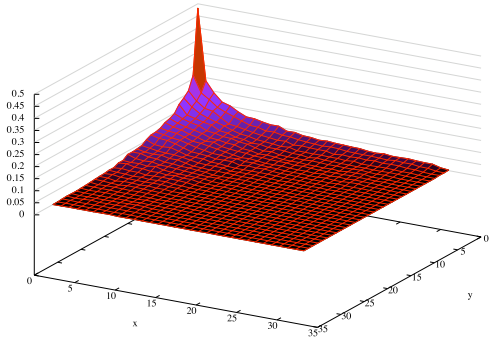
(a) Lenna



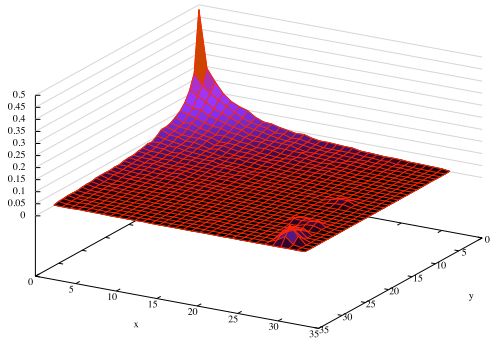
(b) Barbara



(c) Baboon



(d) Goldhill



(e) Peppers

Figure 3.18 – Probabilite de 1 selon la position (x, y)

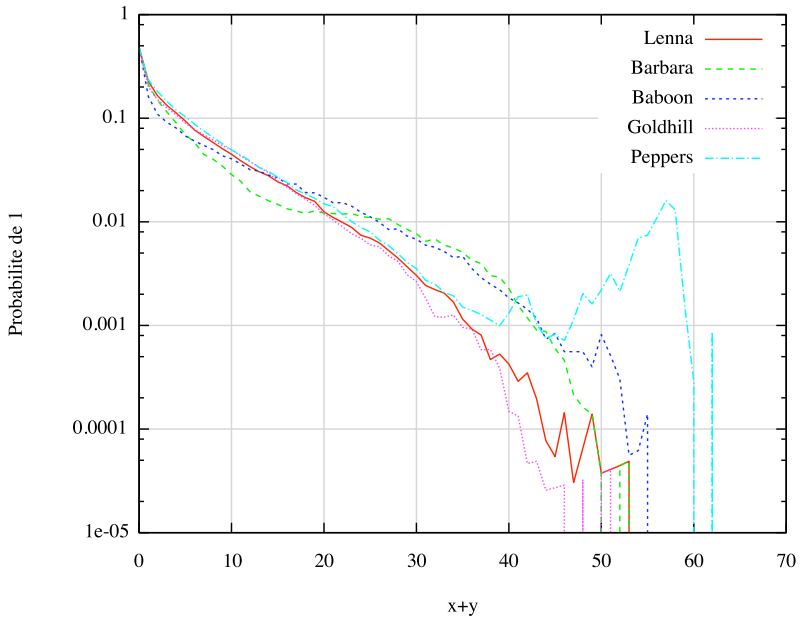


Figure 3.19 – Probabilité de 1 selon $x + y$

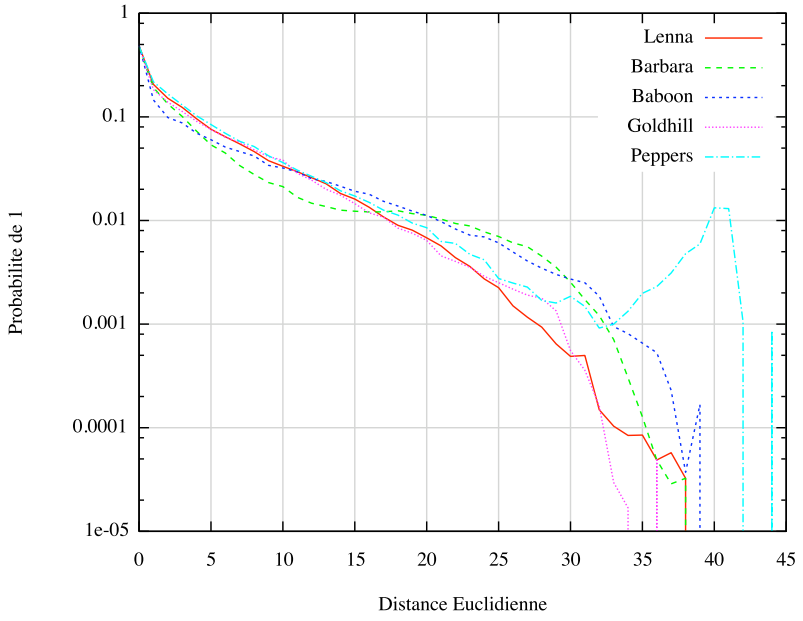


Figure 3.20 – Probabilité de 1 selon $\sqrt{x^2 + y^2}$

Il est vérifié à la figure 3.21 que la forme d'exponentielle décroissante est applicable pour tous les plans de bits. Toutefois, on remarque une grande différence entre les images et une certaine ressemblance dans la forme de la courbe pour les plans de bits d'une même image.

3.3.2 Observations relatives

Les observations relatives considèrent les données déjà codées. Il s'agit d'étudier les effets des bits environnants sur le bit qui est sur le point d'être encodé.

Effet des bits plus significatifs

La présence d'un 1 dans un plan de bit supérieur est très significatif puisque cela signifie que le coefficient est différent de zéro. Cela pourrait nous amener à croire que les bits qui se trouvent sous un 1 ont une probabilité de $0,5$. Or, l'étude de la probalité de 1 selon la valeur au-dessus révèle qu'elle tend effectivement vers $0,5$, mais que ce n'est pas le cas pour des petites valeurs au-dessus au bit. Les figures 3.22 et 3.23 montrent cet état de fait. Il est important de noter qu'en raison du faible nombre de données qui possèdent de hautes valeurs, de grandes variations apparaissent à la figure 3.22.

Effet des bits voisins du même plan de bits

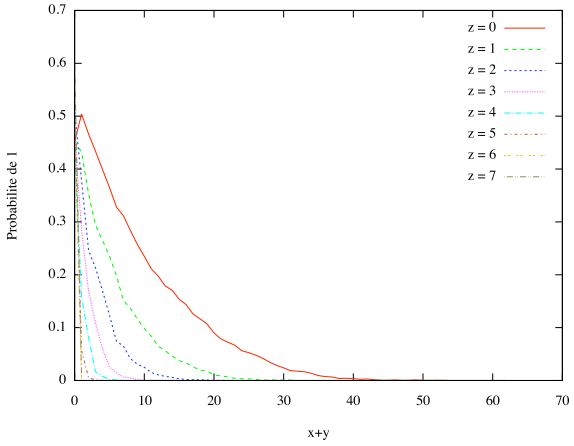
La figure 3.24 montre que la présence d'un bit environnant augmente la probabilité de 1 pour le bit courant. L'interprétation correcte de la figure est que pour tous les bits qui ont un 1 comme voisin $\Delta(x, y) = (-1, 0)$, 35% ont la valeur 1. Il est intéressant de remarquer que les bits n'ont pas une contribution proportionelle à leur distance et que la direction a une grande influence.

Il est également possible de s'intéresser à la somme des voisins du même plan. Les figures 3.25 et 3.26 s'intéressent à cette donnée. On remarque que les courbes tendent vers une probabilité de 1 de 50% et sont tout de même assez semblables.

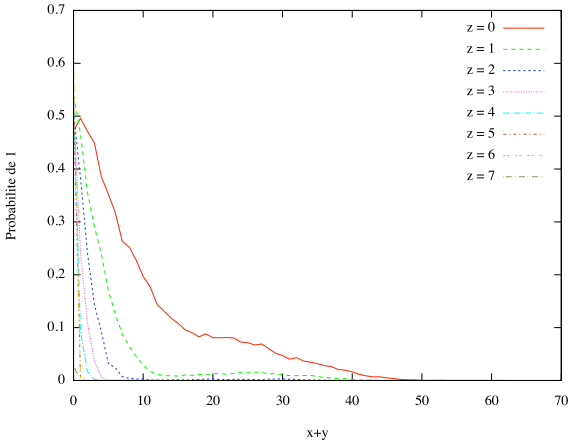
Effet des bits voisins du plan de bits supérieur (z+1)

La figure 3.28 indique que le nombre de bits égaux à 1 en ce qui à trait aux neuf bits les plus près du plan supérieur n'est pas d'une grande importance. La distinction à faire est entre l'absence et la présence d'au moins un 1 parmi ces bits.

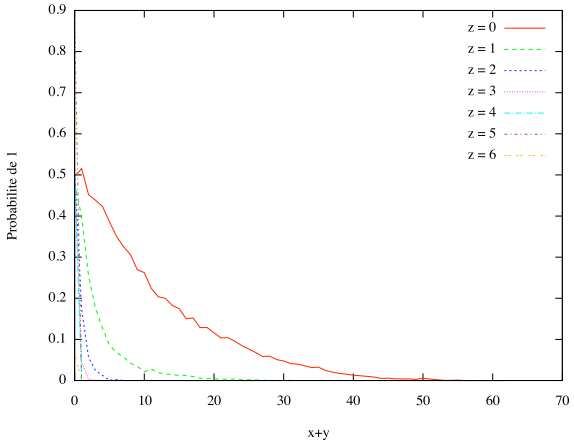
Lorsqu'on agrandit la zone aux 25 bits les plus près du plan de bits supérieur, il apparaît un niveau intermédiaire lorsque la somme est de 1 tel que montré à la figure 3.29.



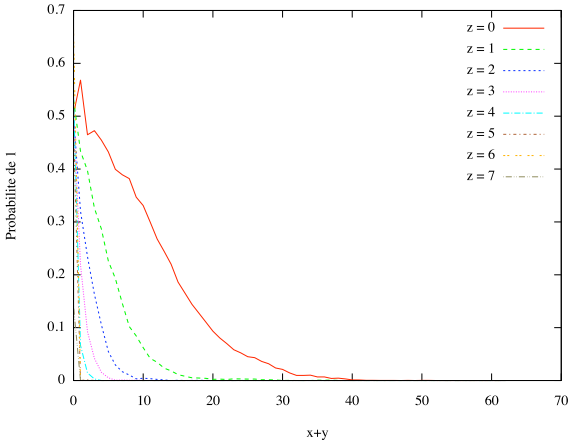
(a) Lenna



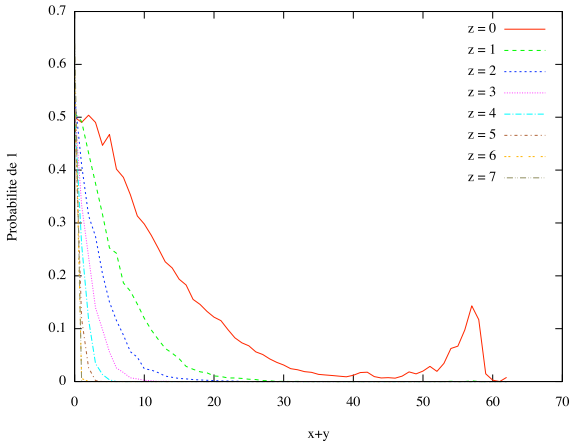
(b) Barbara



(c) Baboon



(d) Goldhill



(e) Peppers

Figure 3.21 – Probabilite de 1 selon la distance et le plan $x + y$

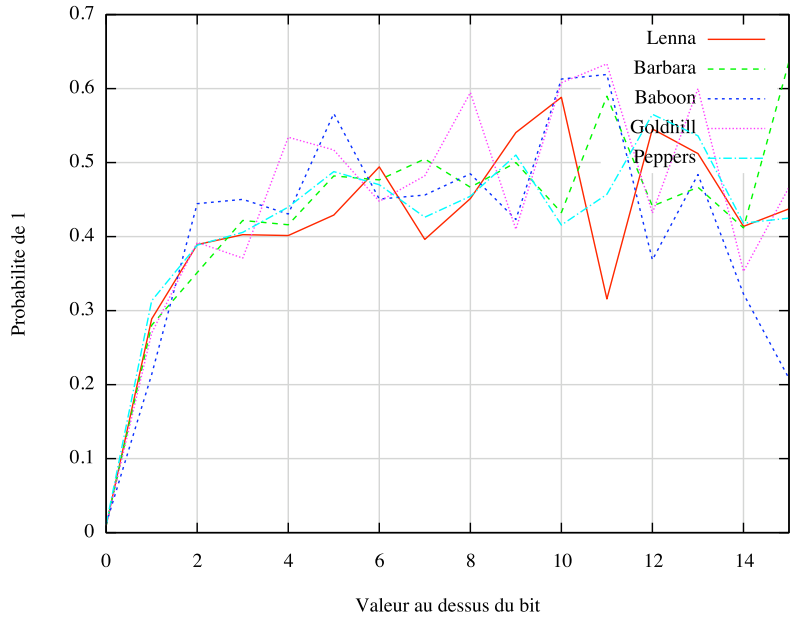


Figure 3.22 – Probabilité de 1 selon la valeur au-dessus du bit

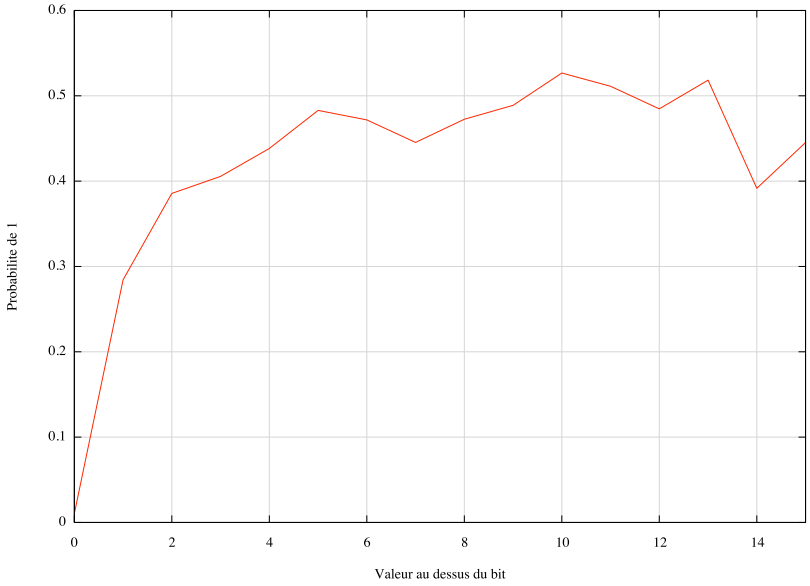


Figure 3.23 – Probabilité de 1 selon la valeur au-dessus du bit pour toutes les images

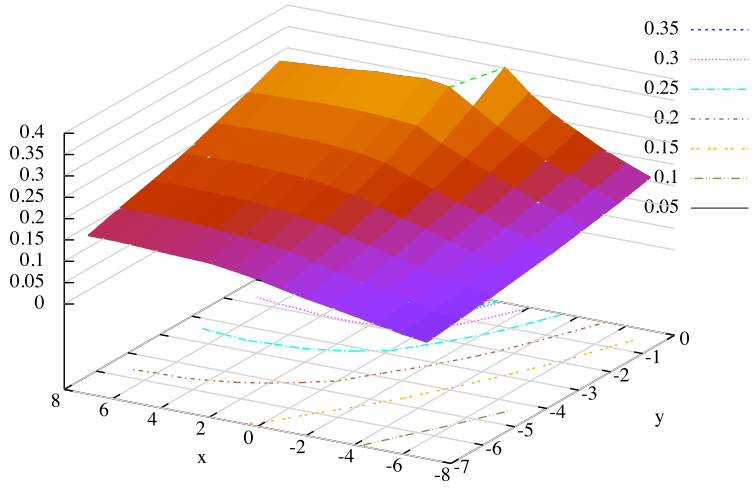


Figure 3.24 – Probabilité de 1 selon la presence d'un 1 au bit voisin déplacé de $\Delta(x, y)$

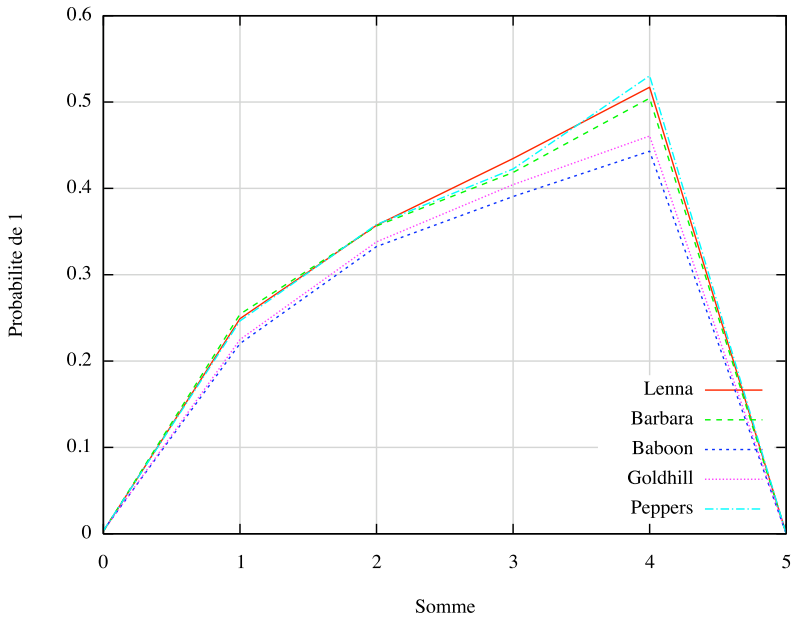


Figure 3.25 – Probabilité de 1 selon la somme des bits voisins du même plan

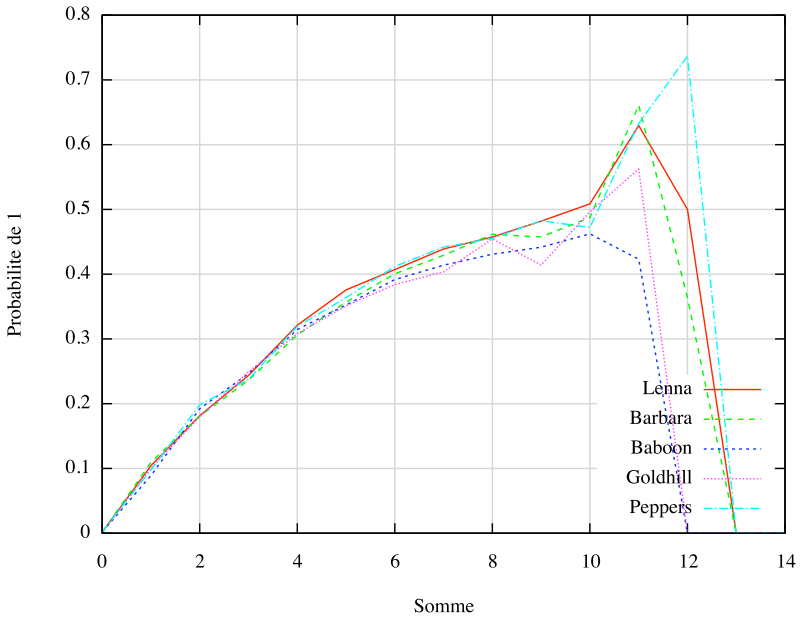


Figure 3.26 – Probabilité de 1 selon la somme des bits voisins du même plan distance de 2 ou moins

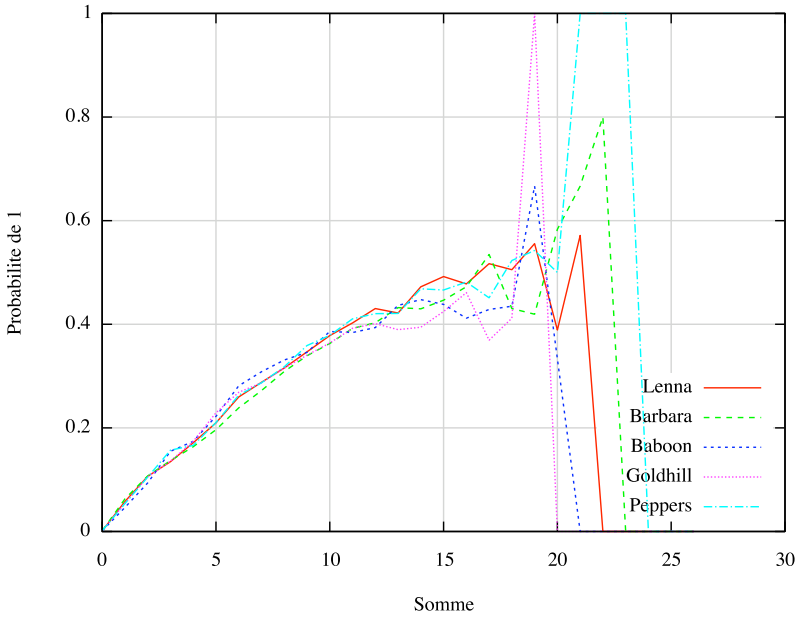


Figure 3.27 – Probabilité de 1 selon la somme des bits voisins du même plan distance de 3 ou moins

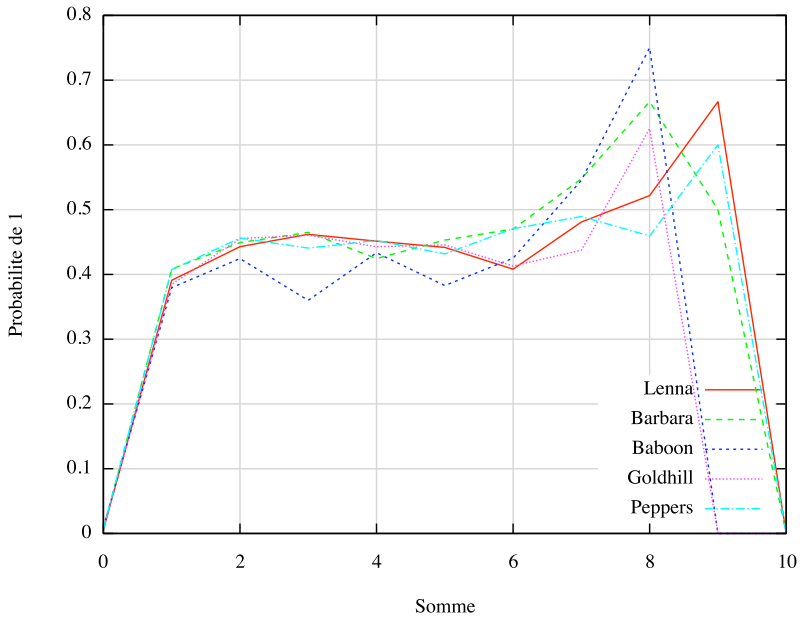


Figure 3.28 – Probabilité de 1 selon la somme des 9 bits du plan supérieur les plus près

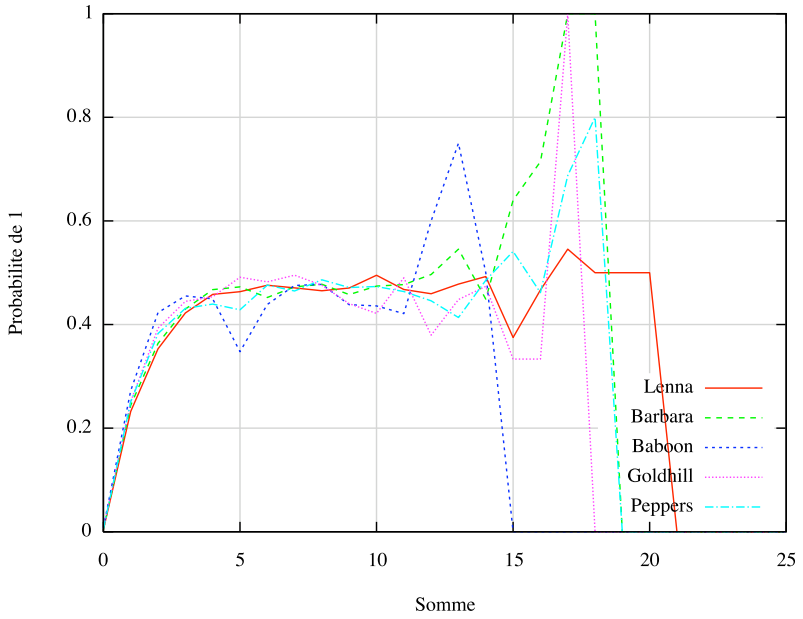


Figure 3.29 – Probabilité de 1 selon la somme des 25 bits du plan supérieur les plus près

En agrandissant davantage jusqu'à une zone de 49 bits, la situation est très peu améliorée. Quelques niveaux intermédiaires sont obtenus pour les sommes faibles, tels qu'illustrés à la figure 3.30.

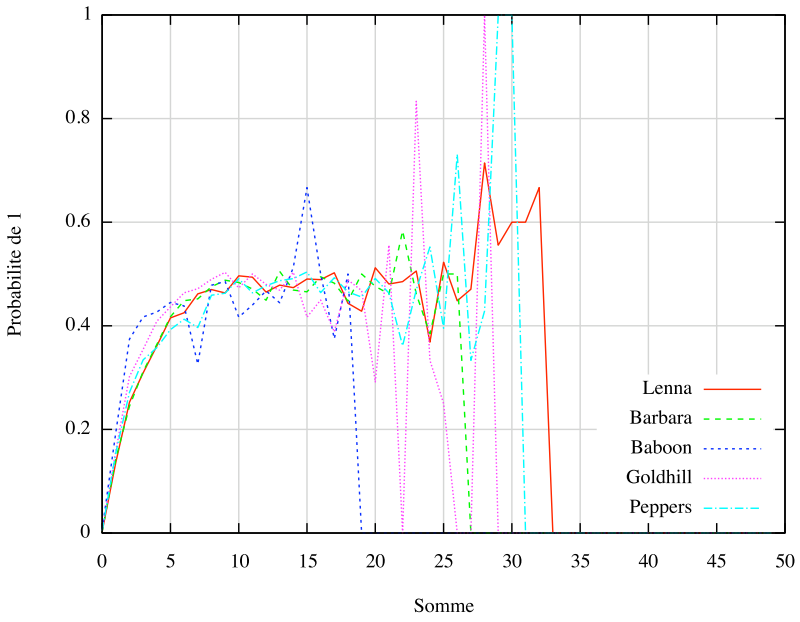
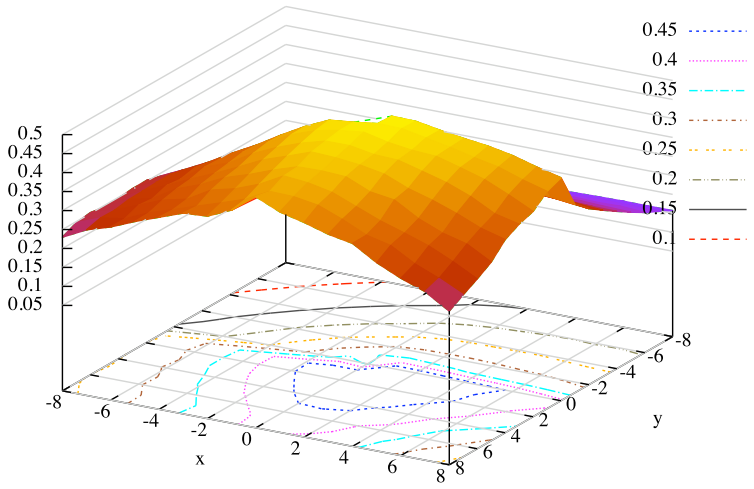


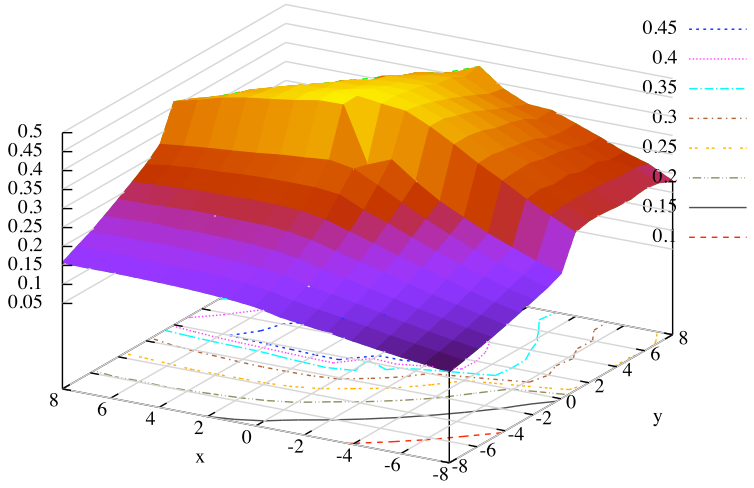
Figure 3.30 – Probabilité de 1 selon la somme des 49 bits du plan supérieur les plus près

Effet des coefficients voisins

En examinant la figure 3.35, il est possible de s'interroger sur la contribution du plan de bits courant sur les courbes de tendance. En effet, la figure 3.27 a une forme similaire mais avec une pente plus douce. La figure 3.36 présente la relation entre le nombre de coefficients ayant au moins un 1 d'encodé et la probabilité du symbole 1, et ce, en excluant les bits du plan courant. La transition est très rapide, ce qui rend la somme peu significative. Toutefois, l'absence totale de coefficients différents de zéro reste significative. Cette même situation est confirmée pour des nombre inférieurs de coefficients. La conclusion tirée est que le plan courant contient la majeure partie de l'information.

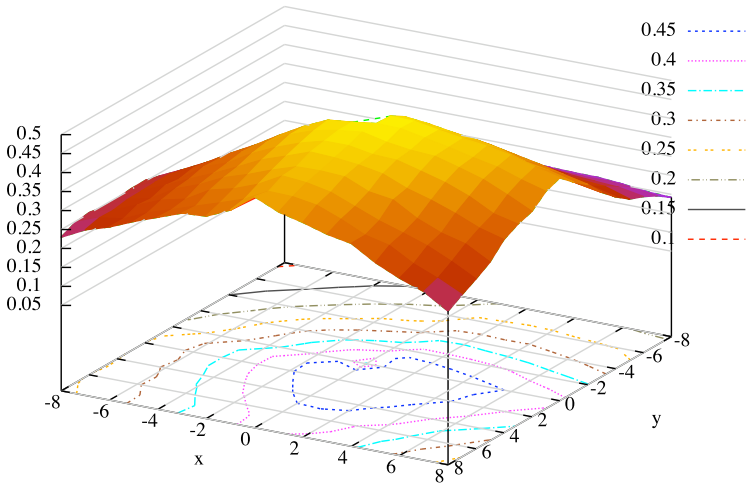


(a) Point de vue A

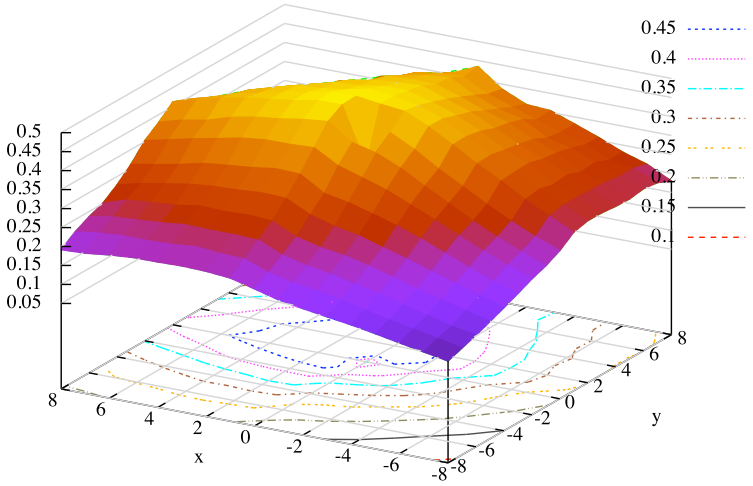


(b) Point de vue B

Figure 3.31 – Probabilité de 1 selon la presence d'un coefficient voisin différent de 0 déplacé de $\Delta(x, y)$



(a) Point de vue A



(b) Point de vue B

Figure 3.32 – Probabilité de 1 selon la presence d'un coefficient voisin différent de 0 déplacé de $\Delta(x, y)$ en excluant le plant courant

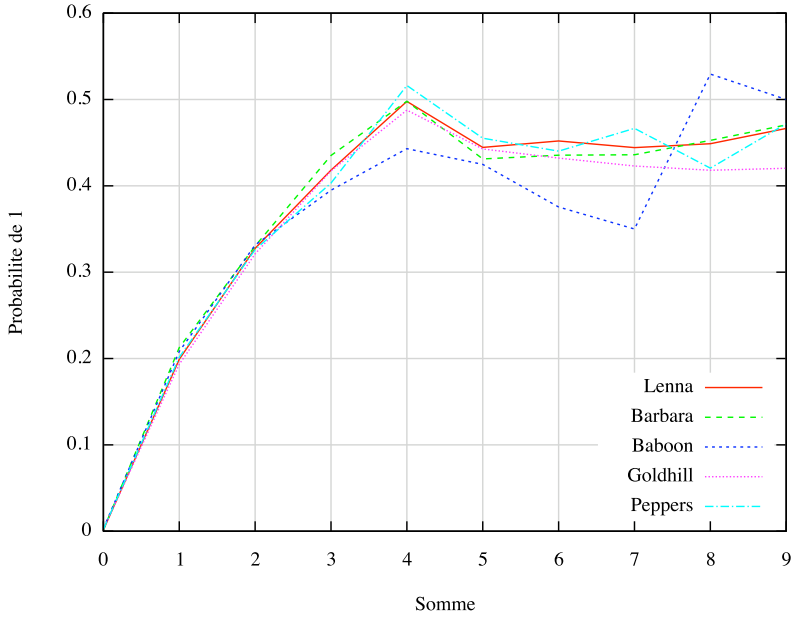


Figure 3.33 – Probabilité de 1 selon la somme des 9 coefficients les plus près ayant un 1 déjà codé

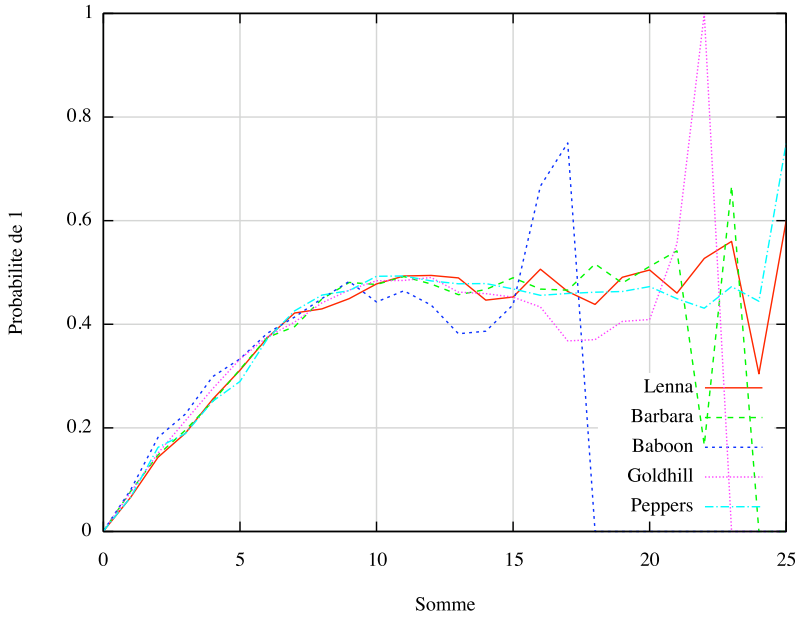


Figure 3.34 – Probabilité de 1 selon la des 25 coefficients les plus près ayant un 1 déjà codé

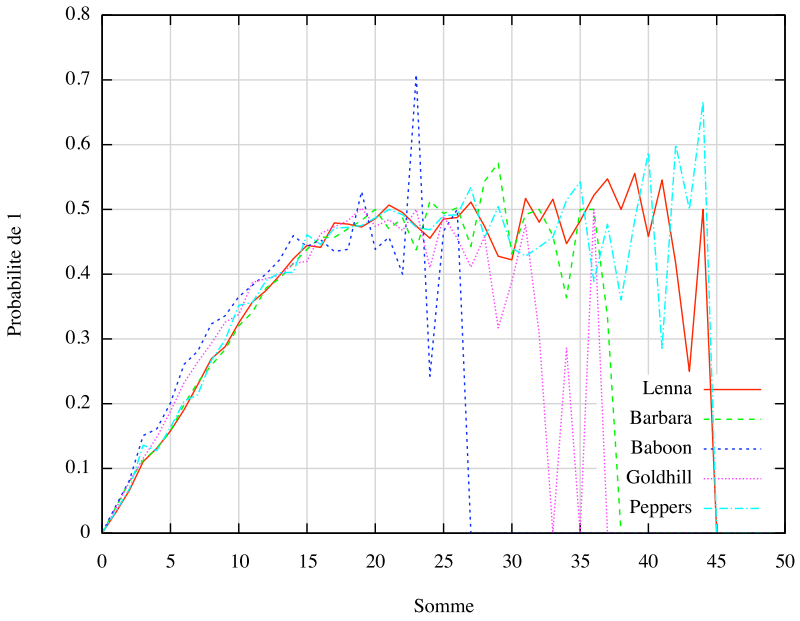


Figure 3.35 – Probabilité de 1 selon la des 49 coefficients les plus près ayant un 1 déjà codé

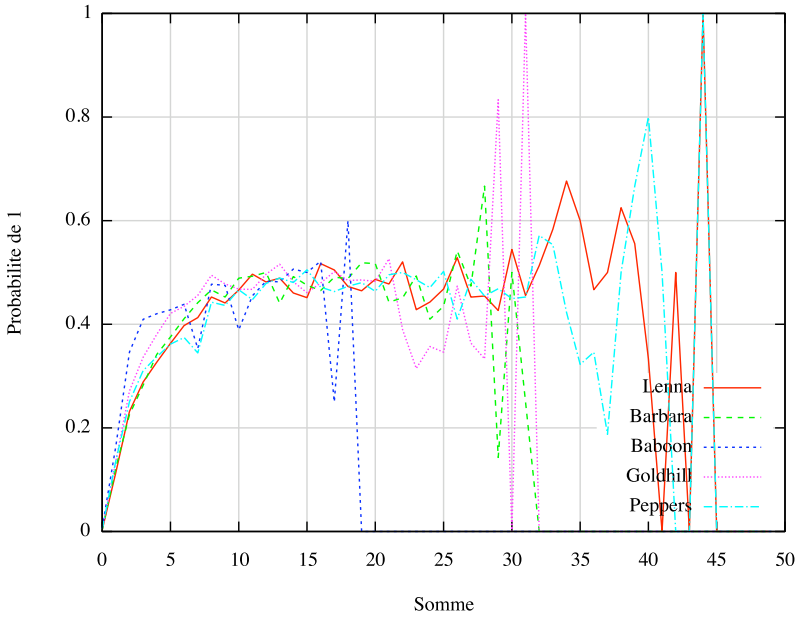


Figure 3.36 – Probabilité de 1 selon la des 49 coefficients les plus près ayant un 1 déjà codé en excluant le plan courant

CHAPITRE 4

CODAGE ARITHMÉTIQUE BINAIRE À L'AIDE D'UN RÉSEAU DE NEURONES À RÉACTION POSITIVE

Avant-propos

Auteurs et affiliations :

- J. S. A. Beaudry : étudiant à la maîtrise, Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique
- C. T. Le Dinh : professeur, Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.
- F. Michaud : professeur, Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.

Date de soumission : 16 avril 2009

Revue : *IEEE Letters on Signal Processing*

Titre original : Adaptive Binary Arithmetic Image Coding with a FeedForward Neural Network

Titre français : Codage arithmétique binaire à l'aide d'un réseau de neurones à réaction positive

Cet article présente l'utilisation d'un réseau de neurones à réaction positive de deux couches qui possède neuf entrées et une sortie pour le modelage du contexte des coefficients quantifiés de la DCT. Cette approche atteint les performances de JPEG2000.

4.1 Introduction

Adaptive Binary Arithmetic Coding [Subbalakshmi, 2003] involves three steps : binarization, context-modeling and binary arithmetic coding (BAC). BAC's performance is mainly limited by context modeling, which uses information known to both the coder and the decoder to estimate the probabilities of the symbols of the source.

Existing methods to perform this task in the realm of image compression include automatons with context models [Taubman, 2000] and trees with context models [Ponomarenko et coll., 2005]. We propose using a two-layer feedforward neural network (FFNN) consisting of a single bounded linear artificial neuron, for low bit rate coding.

The paper is organized as follows. Section 4.2 presents the coder used to test our FFNN context modeling technique, which is described in Section 4.3. Section 4.4 presents a comparison with JPEG and JPEG2000 [Skodras et coll., 2001], demonstrating that the proposed approach performs well and can be simply implemented in hardware.

4.2 Coder setup

The encoding process of the proposed coder is illustrated in Figure 4.1. For the binarization step, the image is transformed using DCT on blocks of 32×32. The coefficients obtained are quantized uniformly using a specified quantization step and rounded to the closest integer.

All coefficients equal to ±1 that have no neighbors different from zero over three rows and three columns or less are set to zero. This reduces the compressed image size by up to 2% and only creates minor image degradation. The signs of the remaining non-zero coefficients are simply passed on to the output stream, uncoded.

The context modeling module processes blocks of integer DCT coefficients, separated in bit-planes. The resulting structure can be seen as a 3-D array of bits. We denote a DCT coefficient $B_{i,j,x,y}$, with i and j being respectively the horizontal and vertical indices of the block containing the coefficient. The x and y indices represent the horizontal and vertical coordinates of the coefficient within the block. Individual bits of a coefficient are denoted $B_{i,j,x,y,z}$, with z being the order of the bit and $z = 0$ being the index of the least significant bit. Considering w to be the width of the image, h its height and k the number of bit-planes, $i \in \{0, 1, 2, ..., w/32 - 1\}$, $j \in \{0, 1, 2, ..., h/32 - 1\}$, $x \in \{0, 1, 2, ..., 31\}$, $y \in \{0, 1, 2, ..., 31\}$ and $z \in \{0, 1, 2, ..., k - 1\}$.

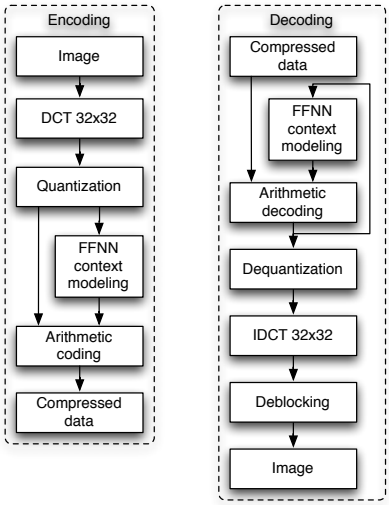


Figure 4.1 – Encoding and decoding processes

Coding starts with the most significant bit-plane ($z = k - 1$) and each bit-plane is coded entirely before going to the next most significant, for progressive decoding. Each bit-plane is coded row by row (by incrementing x first) using the context modeling module presented in Section 4.3 to estimate the probability of the current symbol. Binary arithmetic coding is then used to generate the code word.

The decoding process consists of the complement operations of the encoding process, performed in the reverse order followed by the removal of blocking artifacts using the approach described in [Egiazarian et coll., 1999]. The removal of blocking artifacts yields an increase in image quality of 0.5 to 1 dB.

4.3 FFNN Context Modeling

The context modeling operation consists of transforming characteristics of the context in an estimation of the probability of the symbols. In the considered case, only two symbols are present, i.e., 0 and 1. Thus, the estimation of the probability of one symbol is the inverse of the other, i.e. $P(B_{i,j,x,y,z} = 0) = 1 - P(B_{i,j,x,y,z} = 1)$.

We consider the following nine characteristics to estimate the probabilities of the symbols to use by our coder :

C_0 1 if $y = 0$; otherwise 0.

C_1 1 if $x = y = 0$; otherwise 0.

C_2 1 if there is at least one bit equal to 1 in the higher bit planes for this coefficient, i.e., $1 \in \{B_{i,j,x,y,z+1}, B_{i,j,x,y,z+2}, ..., B_{i,j,x,y,k-1}\}$; otherwise 0.

C_3 1 if there is at least one bit equal to 1 in the higher bit planes of the eight immediate neighboring coefficients ; otherwise 0.

C_4 1 if there is at least one bit equal to 1 in the already coded neighboring bits of the current bit plane, i.e., $1 \in \{B_{i,j,x-1,y,z}, B_{i,j,x-1,y-1,z}, B_{i,j,x,y-1,z}, B_{i,j,x+1,y-1,z}\}$; otherwise -1, which yields slightly better results than 0 for this condition.

C_5 1 if there is at least one coefficient different from 0 in the square formed by the coefficients displaced by two rows or two columns ; otherwise 0.

C_6 1 if there is at least one coefficient different from 0 in the square formed by the coefficients displaced by three rows or three columns ; otherwise 0.

C_7 1 if there is at least one coefficient different from 0 displaced by one block horizontally or vertically, i.e., $B_{i,j,x,y} \neq 0 \vee B_{i-1,j-1,x,y} \neq 0 \vee B_{i,j-1,x,y} \neq 0 \vee B_{i+1,j-1,x,y} \neq 0 \vee B_{i-1,j,x,y} \neq 0 \vee B_{i+1,j,x,y} \neq 0 \vee B_{i-1,j+1,x,y} \neq 0 \vee B_{i,j+1,x,y} \neq 0 \vee B_{i+1,j+1,x,y}$; otherwise 0.

C_8 The minimum value between four (represented over three bits 100_b) and the number of immediate neighboring coefficients that are different from 0.

The characteristics used are inspired from the 17 conditions used in [Ponomarenko et coll., 2005, 2007]. We tested over 600 algorithmically generated conditions, selected the most significant and optimized them manually for the considered configuration. These conditions examine the position of the bit to encode and, more importantly, the presence of neighbours different from 0 around the bit. The importance of neighbours is explained by the energy compaction accomplished by the DCT. Removing any of these nine remaining characteristics as inputs to the FFNN typically increase image size by over 1%.

These characteristics are used as inputs to our FFNN context modeling module. They are modulated by weights and then summed and bounded at 0^+ and 1^- using a linear clipper activation function, to accommodate the arithmetic coding stage. The output of the neural network could have been either $P(B_{i,j,x,y,z} = 0)$ or $P(B_{i,j,x,y,z} = 1)$, the other being obtained by $P(0) = 1 - P(1)$. We propose using $B_{i,j,x,y,z}$ as an estimation of $P(B_{i,j,x,y,z} = 1)$. This choice allows us to eventually conduct online training. Figure 4.2 illustrates a possible hardware implementation of the proposed context-modeling module.

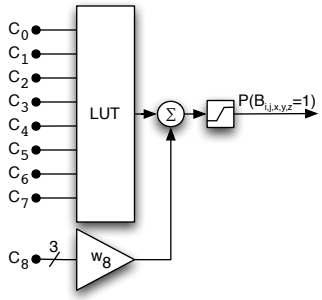


Figure 4.2 – Possible hardware implementation of the proposed FFNN.

TABLEAU 4.1 – Weights of the artificial neuron

Input	Training Set A	Training Set B
C_0	5/64	5/64
C_1	1/2	15/32
C_2	−13/64	−15/64
C_3	1/16	3/64
C_4	3/64	1/64
C_5	5/64	3/32
C_6	5/64	5/64
C_7	3/32	5/64
C_8	1/16	5/64
bias	−1/128	−1/32

The output before the clipper is $\sum_{i=0}^{8} C_i w_i + w_{\text{bias}}$. The look-up table (LUT) can compute the result of $\sum_{i=0}^{7} C_i w_i + w_{\text{bias}}$, while $C_8 w_8$ can be computed separately and added to the output of the LUT.

Before adopting such configuration for the neural network, we conducted extensive tests using various number of configurations. We observed that the addition of one or two hidden layers composed of 5 to 50 neurons do not improve the performance for these inputs. The use of other activation functions such as sigmoid, Gaussian, Elliott [Elliott, 1993] and sinusoidal did not result in better performances compared to the linear activation function used.

Training is done offline via the iRPROP$^-$ algorithm [Igel et Husken, 2000] using two training sets : A) the upper-left 1/16 portion of Lenna, Barbara, Baboon, Goldhill and Peppers ; and B) Lenna. Table 4.1 presents the resulting weights. The similarities between these weights show that the neural network generalizes well in such application. This shows that the results using training set A are not illegitimately improved by using the upper-left 1/16 of the images for the training set and the whole image for the test set.

TABLEAU 4.2 – Comparison of coding efficiency

		Size (bpp)			
		0.125	0.25	0.5	1
Image	Coder	PSNR (dB)			
Lenna	FFNN	31.21	34.31	37.31	40.22
	J2000	31.02	34.15	37.27	40.33
	Δ	**0.19**	**0.16**	**0.04**	**-0.11**
Barbara	FFNN	27.21	30.47	34.47	39.04
	J2000	25.87	28.89	32.87	38.07
	Δ	**1.34**	**1.58**	**1.6**	**0.97**
Baboon	FFNN	21.70	23.46	25.90	29.48
	J2000	21.68	23.18	25.57	29.11
	Δ	**0.02**	**0.28**	**0.33**	**0.37**
Goldhill	FFNN	28.70	30.88	33.46	36.85
	J2000	28.49	30.53	33.24	36.54
	Δ	**0.21**	**0.35**	**0.22**	**0.31**
Peppers	FFNN	30.65	33.10	35.25	38.04
	J2000	30.79	33.54	35.80	38.17
	Δ	**-0.14**	**-0.44**	**-0.55**	**-0.13**

In addition, using weights derived from training set B increases the size of coded images for Lenna of around 2% comprared to the training set A, even if the training set B is the entire Lenna image. For the other images, training set B increases the coded image size by and average of 5% (and up to 8%) for Baboon. Therefore, we chose to use the weights derived from training set A for our experimental results.

Other training methods have been tried with similar or worse results, namely, one-step secant backpropagation, Levenberg-Marquardt backpropagation, gradient descent with momentum and adaptive learning rate backpropagation, conjugate gradient backpropagation with Fletcher-Reeves updates or Powell-Beale restarts and BFGS quasi-Newton backpropagation [Du et Swamy, 2006].

4.4 Results

As done in [Ponomarenko et coll., 2005], the compression efficiency of the FFNN context modeling module was analyzed using common 512×512 grayscale images (Lenna, Barbara, Baboon, Goldhill, Peppers), and compared to the performance of JPEG2000 [Skodras et coll., 2001] (denoted J2000, using the Kakadu coder by D. Taubman [Taubman et Marcellin, 2001]) for different quantization step size. Results are presented in Table 4.2

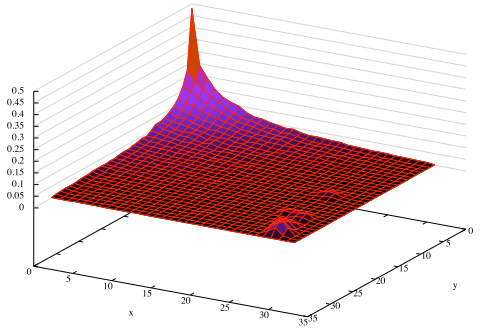


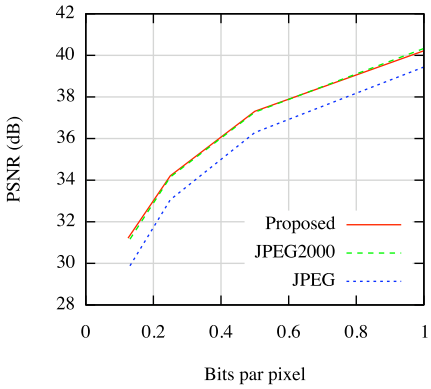
Figure 4.3 – Probability of 1 according to (x, y) position in a block for Peppers using the FFNN coder

and in Figure 4.4. The improvement in PSNR of FFNN over J2000 is presented in rows Δ. FFNN context-modeling brings an overall improvement of 0.34 dB on average over JPEG2000. The best performance is observed for Barbara (1.37 dB average improvement) because the patterns in this image are better coded using DCT rather than through DWT (Discrete Wavelet Transform) used with JPEG2000. For images with sharp edges, such as Peppers, for which DWT provides better performance, JPEG2000 gets 0.32 dB improvement on average over our FFNN context modeling module. More specifically, with Peppers, the coefficients at very high DCT frequencies, isolated from the rest, as shown in Figure 4.3, are difficult to predict. This results in making arithmetic coding produce long code words. Similar bumps at high DCT frequencies are not present in the other test images.

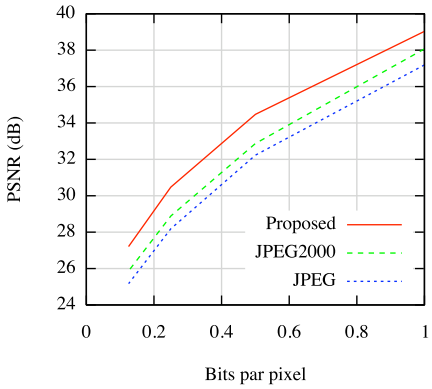
Figures 4.5 and 4.6 show Lenna at 0.125 bpp and 0.25 bpp coded by the proposed coder. It shows the subjective image quality to be acceptable.

4.5 Conclusion

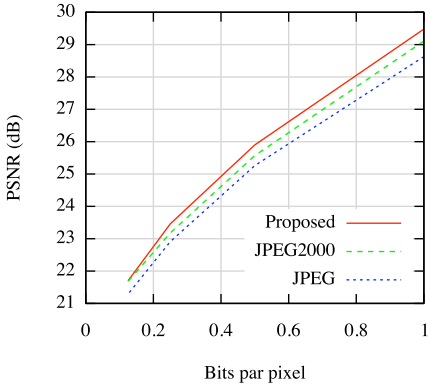
The use of a feedforward neural network context modeling module for DCT coefficients resulted to be a valid approach that matches the performance of JPEG2000, and reveals to be a promising technique for generating bit plane probability in arithmetic coding. Using the proposed FFNN with weights that do not change during coding instead of using adaptive probability models makes a parallel block coding implementation possible. In future work, improving performance by using different context characteristics or by using



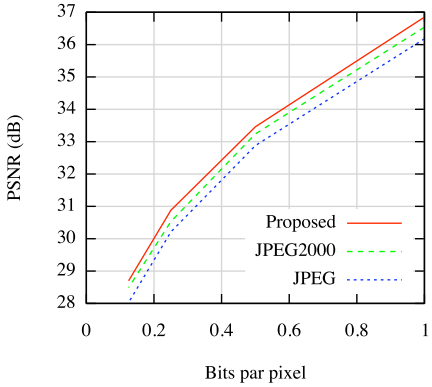
(a) Lenna



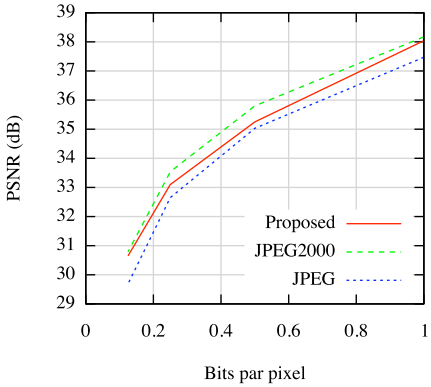
(b) Barbara



(c) Baboon



(d) Goldhill



(e) Peppers

Figure 4.4 – Image quality relative to size of all five images

Figure 4.5 – Lenna at 0.125 bits per pixel, 31.21 dB, using the FFNN coder

other transforms such as DWT or lapped orthogonal transform (GLBT [Tran et coll., 1998]) are going to be investigated.

Acknowledgments

F. Michaud holds the Canada Research Chair (CRC) in Mobile Robotics and Autonomous Intelligent Systems. Support for this work is provided by the Natural Sciences and Engineering Research Council of Canada and the Universite de Sherbrooke.

Figure 4.6 – Lenna at 0.25 bits per pixel, 34.31 dB, using the FFNN coder

CHAPITRE 5

DISCUSSION

La richesse des outils présentée à la section 3.2 a permis des expérimentations diverses. Ces expérimentations visaient l'amélioration du codec AGU principalement en ce qui a trait à sa performance qualité-compression. Elles sont reliées au projet global de codec vidéo pour la télésupervision, mais dissociées des objectifs de simplification du codec d'image. Les pistes explorées sont l'amélioration de l'arbre de conditions, l'ordre de codage des bits et l'utilisation de modeleurs de contexte similaires pour la DWT. L'objectif de cette section est de présenter sommairement les conclusions obtenues.

L'arbre utilisé dans [Ponomarenko et coll., 2005] comporte 44 conditions, qui plus est, une copie différente est utilisée pour chaque plan de bits. Pour améliorer ce modeleur de contexte, il est possible d'obtenir des performances similaires avec un plus petit nombre de conditions ou avec des conditions plus simples. Les expérimentations menées ont révélé que les conditions utilisées dans [Ponomarenko et coll., 2005] étaient bien choisies et qu'il est difficile d'en créer de meilleures pour la source étudiée. Toutefois, la réorganisation de ces conditions à l'aide d'un algorithme de type *greedy* permet d'organiser automatiquement des conditions en arbre. Les résultats révèlent que les arbres créés sont supérieurs à celui utilisé dans [Ponomarenko et coll., 2005] par une marge non significative. Cela porte à croire que la performance est maximisée pour les conditions utilisées et que la simple modification de l'arbre n'est pas une piste intéressante. D'autre part, la création de conditions optimisées est une piste qui reste à explorer.

En poussant les expérimentations afin de comparer les performances obtenues avec celles de [Ponomarenko et coll., 2005], les résultats présentés au tableau 5.1 indiquent que les performances du FFNN sont légèrement inférieures (0,2-0,3 dB) à celles du codec AGU. Ce désavantage est minime considérant que la structure du codec FFNN permet un codage et un décodage des blocs en parallèle, ce qui est un avantage considérable au niveau de la latence du codec.

L'ordre de codage des bits est également sommairement couvert par les outils développés. Les expérimentations menées révèlent que l'ordre de codage fait une grande différence sur la performance du codeur. La modification de l'ordre de codage a pour effet d'exposer la source d'une façon différente et a donc un impact majeur sur les mesures et statistiques.

TABLEAU 5.1 – Comparison of coding efficiency

		Size (bpp)			
		0.125	0.25	0.5	1
Image	Coder	PSNR (dB)			
Lenna	FFNN	31.21	34.31	37.31	40.22
	AGU	31.50	34.51	37.46	40.52
Barbara	FFNN	27.21	30.47	34.47	39.04
	AGU	27.55	30.77	34.65	39.26
Baboon	FFNN	21.70	23.46	25.90	29.48
	AGU	22.01	23.69	26.12	29.70
Goldhill	FFNN	28.70	30.88	33.46	36.85
	AGU	28.97	31.09	33.65	37.03
Peppers	FFNN	30.65	33.10	35.25	38.04
	AGU	30.90	33.32	35.55	38.33

Des outils ont été créés pour tester divers ordres de codage qui doivent être programmés manuellement. Les résultats préliminaires semblent indiquer que l'ordre zigzag est supérieur à l'ordre ligne par ligne dans la majorité des situations. Toutefois, une étude plus poussée est requise pour confirmer les résultats préliminaires. De surcroît, la création automatisée d'un ordre optimal peut être une avenue qui augmenterait les performances du codec.

Finalement, un des résultats assez intéressants est la compatibilité apparente des outils développés avec la DWT. Tant au niveau de l'optimisation d'arbre que de l'utilisation d'un réseau de neurones, les résultats obtenus avec la DWT sont prometteurs, quoiqu'ils soient inférieurs aux résultats obtenus avec la DCT.

CONCLUSION

Ce mémoire présente un codec d'image basé sur un modeleur de contexte réalisé à l'aide d'un réseau de neurones artificiel. Cette nouvelle approche a l'avantage d'être simple et parallélisable. Les performances du codec dépassent celles de la figure bien établie JPEG2000.

La configuration choisie est un réseau de neurones à deux couches, neuf entrées et une sortie. Cela constitue la plus simple configuration trouvée sans dégrader les performances en-dessous des objectifs établis. Les poids utilisés dans le réseau de neurones étant assez généraux pour des images très différentes, aucune adaptation n'est requise pendant le codage. Cela a pour résultat de permettre le codage des blocs en parallèle et ainsi réduire la latence du système. Qui plus est, la basse précision des poids (six bits) facilite davantage l'implémentation matérielle.

Les résultats obtenus dans le cadre de ce projet de recherche laissent entrevoir beaucoup de possibilités dans le futur. Bien que des réseaux plus complexes pour les entrées utilisées ne permettent pas d'augmenter la performance, il est possible d'étudier comment utiliser davantage d'entrées et un réseau plus complexe pour améliorer la performance.

Toujours dans le même contexte, il est possible d'étudier comment effectuer l'entraînement en même temps que le codage afin d'obtenir un réseau plus spécifique à l'image courante en exploitant sa nature adaptative. Une attention particulière doit être portée à la structure du réseau, ses poids initiaux et aux paramètres d'entraînement.

Une autre avenue est d'étudier comment adapter la technique présentée au codage d'autres transformées telles la DWT ou la GBLT. Cette adaptation requiert qu'on s'intéresse aux entrées significative pour la nouvelle source et à l'ordre de codage, entre autres.

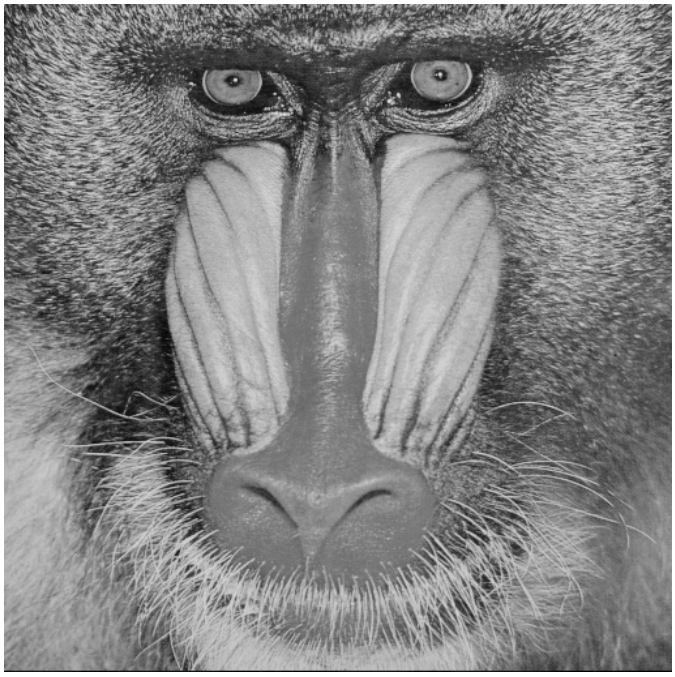
Outre les améliorations de la technique et du codec en lui-même, il est enfin important de s'intéresser à la transformation de celui-ci en codec inter-image afin de répondre aux besoins de télésupervision.

ANNEXE A
IMAGES DE TEST

(a) Lenna

Figure A.1 – Images de test

(b) Barbara



(c) Baboon

Figure A.1 – Images de test

(d) Goldhill

(e) Peppers

Figure A.1 – Images de test

BIBLIOGRAPHIE

Ahmed, N., Natarajan, T., Rao, K.R. (1974) *Discrete cosine transform*, IEEE Transactions on Computers, vol. C-23, p. 90–94.

Antonini, M., Barlaud, M., Mathieu, P., Daubechies, I. (1992) *Image coding using wavelet transform*, IEEE Transactions on Image Processing, vol. 1, n° 2, p. 205–220.

Cassereau, P., Staelin, D., de Jager, G. (1989) *Encoding of images based on a lapped orthogonal transform*, IEEE Transactions on Communications, vol. 37, n° 2, p. 189–193.

Ding, W., Wu, F., Wu, X., Li, S., Li, H. (2007) *Adaptive directional lifting-based wavelet transform for image coding*, IEEE Transactions on Image Processing, vol. 16, n° 2, p. 416–427.

Du, K.L., Swamy, M.N.S. (2006) *Neural Networks in a Softcomputing Framework*, Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Egiazarian, K.O., Helsingius, M., Kuosmanen, P., Astola, J.T. (1999) *Removal of blocking and ringing artifacts using transform domain denoising*, Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, vol. 4, p. 139–142.

Elliott, D. (1993) *A better activation function for artificial neural networks*, Technical Report, Institute for Systems Research, University of Maryland, vol. 8, n° 93.

Huffman, D.A. (1952) *A method for the construction of minimum-redundancy codes*, Proceedings of the Institute of Radio Engineers, vol. 40, n° 9, p. 1098–1101.

Igel, C., Husken, M. (2000) *Improving the Rprop learning algorithm*, Proceedings of the Second ICSC International Symposium on Neural Computation, vol. 1, n° 1, p. 115–121.

Langdon, G., Rissanen, J. (1981) *Compression of black-white images with arithmetic coding*, IEEE Transactions on Communications, vol. 29, n° 6, p. 858–867.

Le Gall, D., Tabatabai, A. (1988) *Sub-band coding of digital images using symmetric short kernel filters and arithmetic coding techniques*, International Conference on Acoustics, Speech, and Signal Processing, vol. 2, p. 761–764.

Lemieux, R., Masson, P., Bellemarre, C., Martin, M., Bisson, D., Bernard, M.È., Fauteux, P., Julien, C., Lalonde-Filion, M., Morier, C., Morin-Guimond, A., Patry, M.A., Saint-Pierre, A., Moisan, P., Michaud, F. (2007) *Telecoaching in traumatology*, Canadian Medical and Biological Engineering Conference.

Malvar, H., Staelin, D. (1989) *The LOT : Transform coding without blocking effects*, IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 37, n° 4, p. 553–559.

Na, S., Yoo, S. (2002) *Allowable propagation delay for VoIP calls of acceptable quality*, *Proceedings of the First International Workshop on Advanced Internet Services and Applications*, Springer-Verlag, London, UK, p. 47–56.

Pennebaker, W.B., Mitchell, J.L., G. G. Langdon, J., Arps, R.B. (1988) *An overview of the basic principles of the q-coder adaptive binary arithmetic coder*, IBM Journal of Research and Development, vol. 32, n° 6, p. 717–726.

Pennebaker, W.B., Mitchell, J.L. (1992) *JPEG Still Image Data Compression Standard*, Kluwer Academic Publishers, Norwell, MA, USA.

Pinson, M., Wolf, S. (2004) *A new standardized method for objectively measuring video quality*, IEEE Transactions on Broadcasting, vol. 50, n° 3, p. 312–322.

Ponomarenko, N.N., Egiazarian, K.O., Lukin, V.V., Astola, J.T. (2005) *DCT based high quality image compression*, Proceedings of the Scandinavian Conference Image Analysis, vol. 3540, n° 14, p. 1177–1185.

Ponomarenko, N.N., Egiazarian, K.O., Lukin, V.V., Astola, J.T. (2007) *High-quality DCT-based image compression using partition schemes*, IEEE Signal Processing Letters, vol. 14, n° 2, p. 105–108.

PRECARN (2005) *Robotics-assisted intervention - final report*, Intelligent Systems Application Studies : Robotics and Automation for Medical Applications.

Rissanen, J., Langdon, G.G. (1979) *Arithmetic coding*, IBM Journal of Research and Development, vol. 23, n° 2, p. 149–162.

Said, A., Pearlman, W. (1996) *A new, fast, and efficient image codec based on set partitioning in hierarchical trees*, IEEE Transactions on Circuits and Systems for Video Technology, vol. 6, n° 3, p. 243–250.

Shannon, C.E. (1948) *A mathematical theory of communication*, Bell System Technical Journal, vol. 27, p. 379–423.

Shapiro, J. (1993) *Embedded image coding using zerotrees of wavelet coefficients*, IEEE Transactions on Signal Processing, vol. 41, n° 12, p. 3445–3462.

Skodras, A., Christopoulos, C., Ebrahimi, T. (2001) *The JPEG 2000 still image compression standard*, IEEE Signal Processing Magazine, vol. 18, n° 5, p. 36–58.

Subbalakshmi, K. (2003) *Lossless Compression Handbook*, Communications, Networking, and Multimedia, Academic Press.

Taubman, D. (Jul 2000) *High performance scalable image compression with EBCOT*, IEEE Transactions on Image Processing, vol. 9, n° 7, p. 1158–1170.

Taubman, D.S., Marcellin, M.W. (2001) *JPEG 2000 : Image Compression Fundamentals, Standards and Practice*, Kluwer Academic Publishers, Norwell, MA, USA.

Tran, T., de Queiroz, R., Nguyen, T. (1998) *The generalized lapped biorthogonal transform*, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 3, p. 1441–1444.

Tran, T., Nguyen, T. (1998) *A lapped transform progressive image coder*, Proceedings of the IEEE International Symposium on Circuits and Systems, vol. 4, p. 1–4.

Wang, Z., Lu, L., Bovik, A.C. (2004) *Video quality assessment based on structural distortion measurement*, Signal Processing : Image Communication, vol. 19, n° 2, p. 132–121.

Watson, A.B., Hu, J., III, J.F.M. (2001) *Digital video quality metric based on human vision*, Journal of Electronic Imaging, vol. 10, n° 1, p. 20–29.

Wu, X. (1997) *High-order context modeling and embedded conditional entropy coding of wavelet coefficients for image compression*, Conference Record of the Thirty-First Asilomar Conference on Signals, Systems and Computers, vol. 2, p. 1378–1382.

Xiong, Z., Ramchandran, K., Orchard, M. (1997) *Space-frequency quantization for wavelet image coding*, IEEE Transactions on Image Processing, vol. 6, n° 5, p. 677–693.

Xiong, Z., Wu, X. (1999) *Wavelet image coding using trellis coded space-frequency quantization*, IEEE Signal Processing Letters, vol. 6, n° 7, p. 158–161.