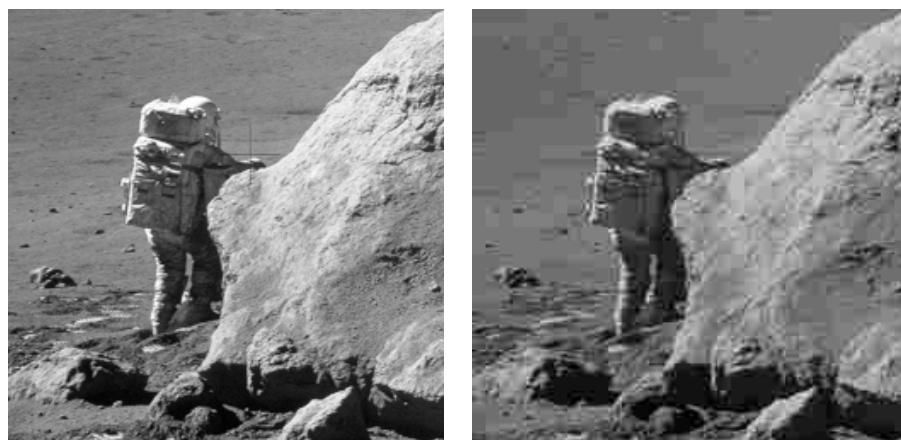


**Année universitaire
2022 - 2023**

**Compression et techniques avancées en
image – 5ETI-IMI**

Compression d'image



Eric Van Reeth

Table des matières

1 Théorie de l'information	2
1.1 Notions de base	2
1.1.1 Codage uniforme	2
1.1.2 Codage à longueur variable	2
1.1.3 Grandeurs caractéristiques	3
1.2 Quantification de l'information	4
1.2.1 L'image comme source d'information	4
1.2.2 Quantité d'information	4
1.2.3 Entropie	4
1.2.4 Borne de Shannon	5
1.2.5 Entropie minimale et maximale	6
2 Codage entropique	6
2.1 Codage de Huffman	6
2.1.1 Encodage	6
2.1.2 Décodage	7
2.2 Codage arithmétique	8
2.2.1 Codage	8
2.2.2 Décodage et implémentation	9
2.2.3 Codage adaptatif	10
3 Réduction de la redondance spatiale	10
3.1 Codage par plages (run-length)	10
3.2 Codage LZW	11
3.3 Codage prédictif	12
3.3.1 Codage prédictif sans perte	12
3.3.2 Codage prédictif avec perte	14
3.3.3 Prédicteurs optimaux	16
4 Compression par transformation	18
4.1 Transformées orthogonales usuelles	18
4.1.1 Représentation matricielle	19
4.1.2 Extension au cas 2D	20
4.1.3 Transformée en cosinus discrète	21
4.1.4 Transformée de Fourier discrète	22
4.1.5 Transformée en ondelettes discrète	23
4.2 Processus de compression	25
4.3 Choix de la transformée	25
4.4 Décorrélation	26
4.5 Troncation et quantification	27
4.6 Codage des coefficients	32
4.7 Compression par blocs	32
5 Compression d'images couleur	33
6 Compression vidéo	33

À l'heure du *big data*, le stockage des données numériques est une problématique centrale aussi bien au niveau économique qu'écologique. La compression, ou l'art de réduire la quantité de données nécessaire à la représentation d'une information, est donc un domaine d'étude central aujourd'hui.

Dans le domaine de l'image (incluant la vidéo), la compression a notamment permis la démocratisation de la photographie numérique malgré l'augmentation de la définition des capteurs, ou la diffusion sur Internet d'images voire de films en haute définition avec des débits raisonnables.

L'objectif de ce cours est d'introduire les concepts théoriques à la base de la compression d'image, ainsi que de décrire en détails certaines des approches les plus utilisées. Les approches de compression vidéo seront également abordées en fin de cours.

1 Théorie de l'information

Une image numérique est le vecteur d'une certaine quantité d'information, stockée ou transmise sous forme de données binaires. Il faut veiller ici à différencier la façon dont l'image est codée sur le support numérique, de l'information que l'image contient. En effet, la même information peut être codée sous différentes formes, avec plus ou moins de données et donc en utilisant plus ou moins d'espace mémoire. Les représentations nécessitant plus de données que les autres sont dites **redondantes**. Nous verrons que la redondance d'une image numérique s'exprime sous différentes formes (redondance spatiale/temporelle, redondance de codage). L'objectif principal de la compression d'image est donc de maximiser la quantité d'information stockée, en minimisant l'espace mémoire associé : c'est à dire de minimiser la redondance. Cette section introduit certaines notions sur la théorie de l'information, qui permettent notamment de définir des bornes sur la quantité de données requises pour coder une certaine quantité d'information.

1.1 Notions de base

L'unité élémentaire de l'information, le **bit** (*binary digit*), a été introduit par John Tukey, collaborateur de Claude Shannon. Un bit constitue la quantité minimale d'information transmise via un support numérique, et ne peut prendre que 2 valeurs, généralement notées 0 et 1. Ainsi, il est possible de quantifier la longueur d'un message en comptant le nombre de bits nécessaires à son codage. Notons que les bits sont régulièrement regroupés par paquets de 8 pour former un octet, unité de base permettant de quantifier une empreinte mémoire.

1.1.1 Codage uniforme

Le codage binaire uniforme permet d'associer un code unique de longueur fixe, à chaque symbole du message. Ainsi, le message composé des symboles suivants : {0, 1, 1, 3, 2}, pourra être codé en utilisant un code binaire de longueur 2.

$$\{0, 1, 1, 3, 2, 1\} \xrightarrow{\text{Codage}} \{00, 01, 01, 11, 10, 01\} \xrightarrow{\text{Regroupement}} 000101111001$$

La longueur du code associé à ce message est donc de $L = 12$ bits. Notons que pour un code binaire uniforme, le nombre de bits nécessaire pour coder un message comportant N valeurs différentes est :

$$l = \lceil \log_2(N) \rceil$$

Cette méthode a l'avantage d'être simple, mais elle est loin d'être efficace du point de vue de la compression.

1.1.2 Codage à longueur variable

Par opposition au codage uniforme, il est tout à fait possible d'associer un code de longueur variable aux différents symboles du message. Par exemple, l'association suivante pourra être choisie :

$$0 \mapsto 001 \quad 1 \mapsto 1 \quad 2 \mapsto 01 \quad 3 \mapsto 000$$

Le code du message de l'exemple précédent devient alors :

$$\{0, 1, 1, 3, 2, 1\} \xrightarrow{\text{Codage}} \{001, 1, 1, 000, 01, 1\} \xrightarrow{\text{Regroupement}} 00111000011$$

On remarque que la longueur du code associé à ce message est de $L = 11$ bits, soit un de moins que précédemment. Notons que le décodage n'est possible que s'il existe une conversion unique entre le code et le message original. De manière générale, l'utilisation d'un codage à longueur variable offre une flexibilité intéressante dans la manière de coder un message et permet généralement de minimiser la redondance de codage.

1.1.3 Grandeurs caractéristiques

Les grandeurs suivantes servent à quantifier l'efficacité relative de différentes représentations numériques d'un même message.

Définition : Taux de compression

On appelle taux de compression, noté C , le ratio entre deux longueurs de code représentant la même information :

$$C = \frac{L_1}{L_2}$$

Ainsi, en reprenant l'exemple précédent, le codage à longueur variable permet d'atteindre un taux de compression de $C = \frac{12}{11} \approx 1.09$ par rapport au codage uniforme.

Définition : Redondance relative

On appelle redondance relative, noté R et souvent exprimée en pourcentage, la grandeur définie telle que :

$$R = 1 - \frac{1}{C}$$

Pour l'exemple précédent, on trouve $R = 1 - \frac{11}{12} \approx 0.08$, ce qui indique que le codage uniforme possède une redondance de 8% par rapport au codage à longueur variable.

Définition : Nombre de bits moyen

On note \bar{L} , le nombre de bits moyen permettant de représenter les symboles d'un message :

$$\bar{L} = \sum_{n=0}^{N-1} l(r_n)p(r_n)$$

où $l(r_n)$ et $p(r_n)$ désignent respectivement la longueur de code et la probabilité d'occurrence du symbole r_n . La probabilité d'obtient simplement en divisant le nombre d'occurrence du symbole par le nombre total de symboles dans le message : $p(r_n) = \frac{o_n}{N}$.

Exemple 1.1.

Nous pouvons maintenant calculer le nombre de bits moyen pour les codages uniforme et à longueur variable pour l'encodage du message précédent : $\{0, 1, 1, 3, 2, 1\}$.

Le codage uniforme donne directement :

$$\bar{L} = 2 \sum_{n=0}^{N-1} p(r_n) = 2 \times \frac{1}{6} \times \sum_{n=0}^{N-1} o_n = 2 \times \frac{1}{6} \times 6 = 2 \text{ bits}$$

Le codage à longueur variable donne :

$$\bar{L} = \frac{1}{6}(3) + \frac{1}{2}(1) + \frac{1}{6}(3) + \frac{1}{6}(2) = \frac{11}{6} \approx 1.83 \text{ bits}$$

1.2 Quantification de l'information

L'exemple précédent permet d'entrevoir qu'il est possible de réduire le nombre de bits nécessaire au codage d'un message en optimisant son codage. La théorie de l'information permet de connaître le nombre de bits minimum nécessaire au codage (sans perte) d'une information.

1.2.1 L'image comme source d'information

Une source d'information discrète est un processus de génération d'une série de variables aléatoires indiquées par la variable entière k pouvant représenter par exemple le temps ou une position :

$$\dots, X_{k-1}, X_k, X_{k+1}, \dots$$

La source est dite **sans mémoire** si les variables sont statistiquement indépendantes les unes des autres. La source est associée à un **alphabet** de dimension N , qui contient les différents **symboles** que les évènements de la source peuvent prendre : $X = [r_0, r_1, \dots, r_{N-1}]$.

On associe à chaque symbole sa probabilité : $P = [p(r_0), p(r_1), \dots, p(r_{N-1})]$.

Une image peut être considérée comme une source d'information. Dans ce cadre, chaque pixel représente un nouvel évènement, et la dimension du dictionnaire est donnée par la dynamique de l'image. Le dictionnaire est fréquemment composé des entiers compris dans l'intervalle $[0, 255]$, soit 256 symboles, lorsqu'un code uniforme de 8 bits est utilisé pour le codage de l'image. Les probabilités associées à chaque symbole sont données par l'histogramme normalisé de l'image, dont les valeurs donnent directement la probabilité d'obtenir un niveau de gris particulier lorsqu'un pixel est tiré au hasard dans l'image.

1.2.2 Quantité d'information

Définition : Quantité d'information

L'apparition d'un nouvel évènement issue d'une source d'information, un nouveau pixel dans le cas d'une image, apporte une certaine quantité d'information définie par :

$$I(r_k) = -\log_m(p(r_k))$$

L'unité associée à cette grandeur dépend de m , la base du logarithme utilisée. Elle s'exprime en **bits** lorsque $m = 2$, en **nats** lorsque $m = e$ (logarithme népérien), et en **dits** (*decimal digits*) lorsque $m = 10$.

Du fait que $p(r_k) \leq 1$ et du signe moins devant le logarithme, cette grandeur est toujours positive. De plus, elle est inversement proportionnelle à la probabilité d'apparition du symbole. Intuitivement, cela s'explique par le fait que l'apparition d'un symbole fréquent apporte moins d'information qu'un symbole n'apparaissant que rarement. Ainsi, l'apparition d'un symbole de probabilité $\frac{1}{2}$ apportera $I = -\log_2(1/2) = 1$ bit d'information.

1.2.3 Entropie

Le concept d'entropie dans le contexte de la théorie de l'information a été introduit par Claude Shannon, et découle directement de la quantité d'information définie précédemment.

Définition : Entropie

Considérons une source d'information sans mémoire composée de N symboles : $\{r_0, r_1, \dots, r_{N-1}\}$, et leur probabilité associées : $\{p(r_0), p(r_1), \dots, p(r_{N-1})\}$.

L'entropie associée à cette source est donnée par :

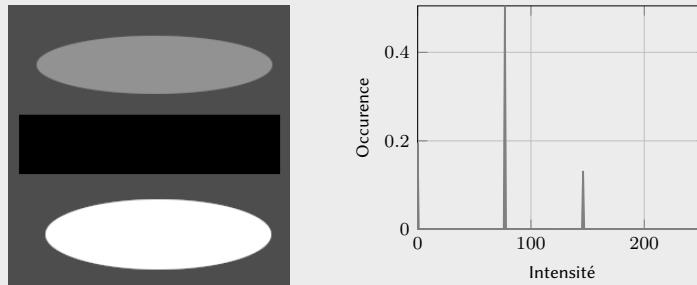
$$H = - \sum_{n=0}^{N-1} p(r_n) \log_m(p(r_n))$$

Elle représente la **quantité d'information moyenne** de la source, et s'exprime généralement en bits (la base du logarithme est $m = 2$ dans ce cas).

Note : par convention, on partira du principe que pour une probabilité nulle on aura : $0 \log_2(0) = 0$.

Exemple 1.2.

Considérons l'image synthétique suivante, ainsi que son histogramme normalisé :



Seuls 4 niveaux de gris composent cette image. Son entropie peut donc être calculée à partir des valeurs de l'histogramme normalisé, telle que :

$$H = - [0.2 \log_2(0.2) + 0.52 \log_2(0.52) + 0.13 \log_2(0.13) + 0.15 \log_2(0.15)] \approx 1.75 \text{ bits}$$

Notons que lorsque les événements d'une source dépendent des événements précédents, on parle de source ou chaîne de Markov. La formule précédente n'est pas applicable pour le calcul de l'entropie d'une source de Markov. En effet, la formule précédente surestimerait l'entropie réelle d'une source de Markov.

1.2.4 Borne de Shannon**Définition : Borne de Shannon**

Le nombre de bits moyen nécessaire pour le codage d'une source sans mémoire ne peut être inférieur à l'entropie associée à cette source, soit :

$$H \leq \bar{L}$$

Ce résultat est fondamental puisqu'il permet de connaître une limite de compression (sans perte) que l'on peut espérer atteindre en optimisant le codage des pixels d'un image. Notons toutefois qu'en présence d'une source de Markov, ce qui est le cas de la plupart des images naturelles qui présentent une forte cohérence spatiale, cette limite peut être repoussée. Utiliser des techniques de codage qui prennent en compte la redondance spatiale permet donc d'optimiser le taux de compression atteignable.

Pour conclure sur l'exemple utilisé dans cette section, à savoir le codage du message : $\{0, 1, 1, 3, 2, 1\}$, calculons son entropie :

$$H = - \left[\frac{1}{6} \log_2\left(\frac{1}{6}\right) + 0.5 \log_2(0.5) + \frac{1}{6} \log_2\left(\frac{1}{6}\right) + \frac{1}{6} \log_2\left(\frac{1}{6}\right) \right] \approx 1.79 \text{ bits}$$

Le codage à longueur variable proposé précédemment permettait d'obtenir un nombre de bits moyen relativement proche de l'optimal, à savoir $\bar{L} = 1.83$ bits. Il est donc possible de conclure que la stratégie consistant à attribuer un code court aux symboles fréquents est efficace pour le codage de ce message.

1.2.5 Entropie minimale et maximale

Comme mentionné précédemment, l'entropie est liée à la quantité d'information moyenne apportée par chaque évènement de la source.

Si une image possède une entropie nulle, $H = 0$, alors elle possède un unique niveau de gris de probabilité 1. Dans ce cas, la quantité d'information apportée par de nouveaux pixels est nulle (aucune incertitude sur la valeur du prochain pixel), et la borne de Shannon indique qu'il faudrait en théorie 0 bit pour coder une telle image. Nous voyons ici que le codage binaire classique est peu efficace puisqu'il permet au mieux d'atteindre la valeur : $\bar{L} = 1$ bit.

À l'inverse, la valeur maximale que peut prendre l'entropie d'une image composée de N pixels est de $H = \log_2(N)$. Cette valeur est atteinte lorsque tous les pixels de l'image sont équiprobables. Dans ce cas, l'information apportée par chaque pixel est importante dans le sens où il est difficile d'estimer correctement la valeur du prochain pixel.

Le taux de compression que l'on peut espérer atteindre en optimisant le codage est donc bien plus faible pour des images présentant des histogrammes uniformes. Nous verrons dans une section ultérieure qu'il peut être avantageux de transformer l'image dans un domaine autre que le domaine spatial (Fourier, ondelettes, DCT), afin de réduire son entropie et d'optimiser le taux de compression atteignable.

2 Codage entropique

Différentes stratégies d'encodage ont été proposées pour diminuer à la fois les redondances de codage et les redondances spatiales dans les images.

2.1 Codage de Huffman

Le codage de Huffman est une des techniques de codage les plus utilisées depuis sa publication en 1952. Il a pour objectif de réduire la redondance de codage en optimisant la longueur du code individuellement associé à chaque symbole. Sous réserve que l'on connaisse les probabilités de chaque symbole, le codage de Huffman permet de générer le code optimal quand les symboles sont traités de façon individuels. C'est-à-dire que lorsque la longueur de la source tend vers l'infini, cette stratégie fait tendre le nombre de bits moyen nécessaire à son codage vers l'entropie. Notons que cette technique ne prenant pas en compte une potentielle corrélation entre symboles, elle n'est pas optimale dans le cas de sources de Markov.

2.1.1 Encodage

L'encodage se déroule en deux étapes successives :

Étape 1 — Réduction de source : Cette étape consiste à trier les symboles par probabilité décroissante, puis à successivement regrouper les deux symboles de plus faibles probabilités. Ce regroupement génère un symbole fictif dont la probabilité est la somme des probabilités des symboles regroupés. Ce processus est itéré jusqu'à réduire la dimension de la source à seulement deux symboles. La Figure 1 illustre le fonctionnement de cette étape sur un exemple simple.

Étape 2 — Codage des symboles : Cette étape consiste à attribuer un code à chaque symbole, en partant de la source taille minimale. Pour la source de taille 2, on attribue arbitrairement 0 au premier symbole et 1 au deuxième. Ensuite, le symbole qui avait généré par combinaison de deux symboles est décomposé en attribuant arbitrairement 0 et 1 aux deux symboles (voir suite de l'exemple en Figure 2). Cette opération est répétée jusqu'à ce qu'un code soit attribué à chaque symbole original de la source. Il est directement possible de constater que par construction, les symboles les plus fréquents ont les codes les plus courts.

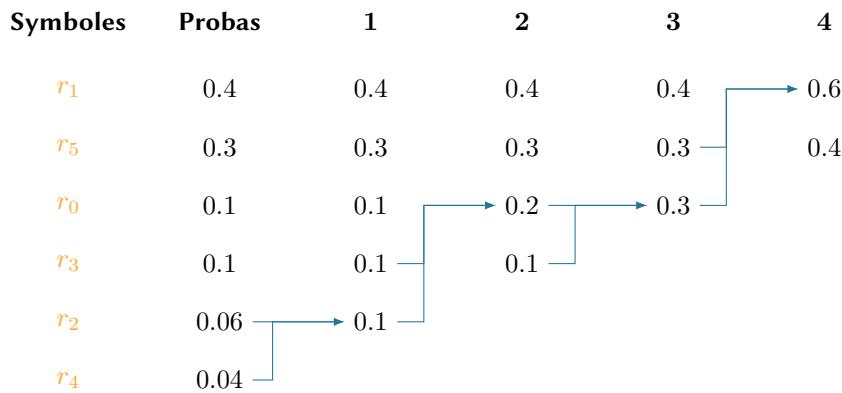


FIGURE 1 : Première étape du codage de Huffman consistant à regrouper successivement les symboles de probabilités les plus faibles. Dans cet exemple 4 réductions de sources sont nécessaires pour réduire la source à deux symboles.

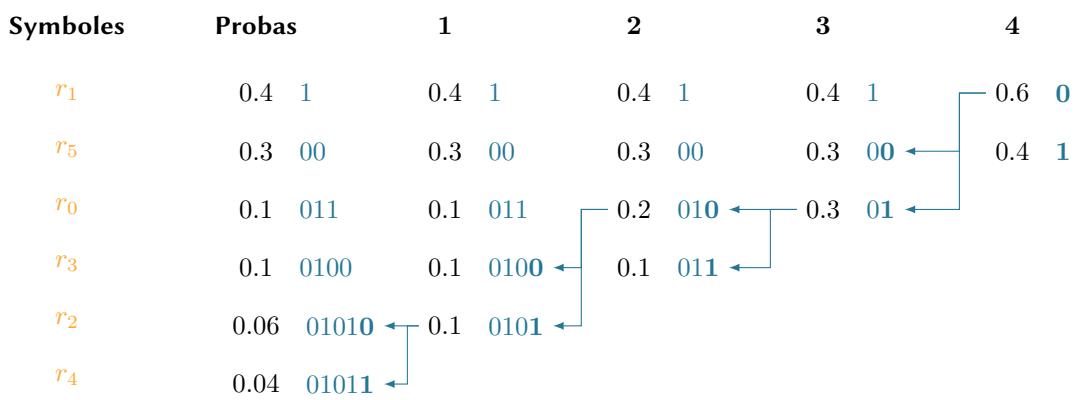


FIGURE 2 : Deuxième étape du codage de Huffman consistant à attribuer un code (en bleu) aux différents symboles de la source, en partant de la source de plus petite dimension.

Le processus de codage devient cependant de plus en plus complexe quand la dimension du dictionnaire augmente. Un dictionnaire de taille N nécessite en effet $N - 2$ réductions de source et $N - 1$ attributions de code. De plus, le calcul de l'histogramme normalisé de l'image n'est pas toujours possible selon les contraintes associées aux différentes applications. Des tables de codage par défaut, définies à partir d'information a priori sur la distribution des coefficients de l'image peuvent permettre d'obtenir un codage quasi-optimal sans perte de temps lors du codage. Cette stratégie est utilisée dans plusieurs standards de compression tels que le JPEG ou le MPEG.

2.1.2 Décodage

La relation entre les symboles de la source et le code associé est stockée dans une table qui est stockée et transmise avec la source encodée. Cela permet un décodage instantané, c'est-à-dire une traduction immédiate du code vers le symbole correspondant. Le décodage a également la particularité d'être unique par construction, dans le sens où aucune ambiguïté n'existe lors de la conversion du code vers les symboles. Par exemple, la chaîne encodée suivante : {010100111100}, correspond forcément à l'enchaînement des symboles : $\{r_2 r_0 r_1 r_1 r_5\}$.

Exemple 2.1.

Reprendons l'image synthétique de l'exemple 1.2, contenant les 4 niveaux de gris suivants : {0, 77, 146, 255}, et dont les probabilités sont : {0.2, 0.52, 0.13, 0.15}. L'application du codage de Huffman, laissé comme exercice, donne par exemple :

$$0 \mapsto 11 \quad 77 \mapsto 0 \quad 146 \mapsto 101 \quad 255 \mapsto 100$$

Le nombre de bits moyen est donc : $\bar{L} = 0.2(2) + 0.52(1) + 0.13(3) + 0.15(3) = 1.76$ bits. La borne fixée par l'entropie était de $H = 1.75$ bits, ce qui montre l'efficacité de cette approche pour cet exemple.

A fortiori, l'utilisation d'un code à longueur fixe, nécessairement de 8 bits dans ce cas, est extrêmement peu efficace. La seule utilisation du codage de Huffman permet d'obtenir un taux de compression de $C = \frac{8}{1.76} \approx 4.5$ sans perte d'information associée. De ce fait, le codage uniforme affiche une redondance de $R = 78\%$ par rapport au codage de Huffman.

2.2 Codage arithmétique

Le codage arithmétique rentre également dans la catégorie du codage entropique dans le sens où il repose sur la connaissance des probabilités des symboles de la source. La différence principale avec le code de Huffman est qu'il **attribue un code à une suite de symboles** (codage par intervalle), au lieu d'attribuer un code par symbole. Bien que théoriquement applicable à des messages de longueur quelconque, le codage d'une quinzaine de symboles successifs est réalisé en pratique.

Le codage arithmétique se différencie du codage de Huffman également dans la forme du code qu'il produit. En effet, le message est codé sous forme d'un nombre décimal compris dans l'intervalle $[0, 1]$. Cet intervalle est progressivement réduit lorsque le nombre de symboles à coder augmente, faisant également augmenter le nombre de bits nécessaire au codage du message.

2.2.1 Codage

L'initialisation de cette méthode consiste à découper l'intervalle de départ $[0, 1]$ en autant de sous-intervalles qu'il y a de symboles à coder. La dimension des sous-intervalles est liée à la probabilité de chaque symbole, comme illustré en Figure 3.

Symboles	Probas	Intervalles
r_0	0.2	$[0.0, 0.2[$
r_1	0.2	$[0.2, 0.4[$
r_2	0.4	$[0.4, 0.8[$
r_3	0.2	$[0.8, 1.0[$

FIGURE 3 : Découpage de l'intervalle $[0, 1]$ en fonction de la probabilité associée à chaque symbole du message à coder.

Pour coder le premier symbole du message, l'intervalle initial est restreint à l'intervalle attribué à ce symbole. Ce nouvel intervalle est de nouveau décomposé en respectant les mêmes proportions que le découpage de l'intervalle initial, c'est-à-dire proportionnel à la probabilité de chaque symbole du message. Cette procédure est répétée jusqu'à ce que le dernier symbole ait été pris en compte. L'intervalle final résultant sera donc d'autant plus petit que les symboles composant le message sont peu probables. Le message entier sera finalement codé en prenant n'importe quel nombre décimal compris dans l'intervalle final. La Figure 4 illustre le processus de codage du message suivant : $\{r_0 r_1 r_2 r_2 r_3\}$, dont les probabilités sont données en Figure 3. Dans cet exemple, le nombre 0.068 sera choisi pour encoder le message car il possède seulement 3 décimales à coder, soit une moyenne de 0.6 décimales par symbole. En comparaison, la valeur de l'entropie de cette source est de :

$$H = - \sum_{n=0}^3 p(r_n) \log_{10}(p(r_n)) = -(3 \times 0.2 \log_{10}(0.2) + 0.4 \log_{10}(0.4)) \approx 0.58 \text{ dits}$$

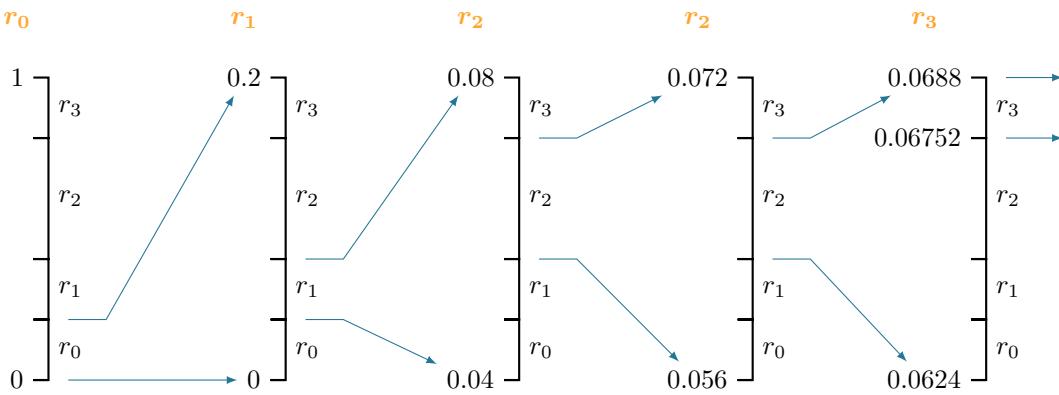


FIGURE 4 : Processus de génération du codage arithmétique. Le code attribué au message est un nombre décimal compris dans l'intervalle de sortie, ici $[0.006752, 0.0688]$.

Notons l'utilisation de la base 10 du logarithme, permettant d'exprimer la quantité information moyenne en unité décimale. Le codage arithmétique permet dans cet exemple d'atteindre un taux de compression proche de la borne optimale.

2.2.2 Décodage et implémentation

Le principe du décodage est relativement simple et permet naturellement de remonter au message de manière unique. Il nécessite simplement de garder en mémoire les probabilités et les intervalles associés à chacun des symboles du message codé. Ainsi, il suffit à chaque étape de retrouver l'intervalle dans lequel le code se situe, puis d'associer le symbole correspondant à cet intervalle à partir du découpage de l'intervalle initial $[0, 1]$. Une fois le symbole déterminé, le code est actualisé et l'opération est répétée jusqu'à l'obtention de tous les symboles du message. La procédure d'actualisation du code c à partir du code courant est la suivante :

$$c^{k+1} = \frac{c^k - b_k}{p_k}$$

avec b_k et p_k respectivement la borne inférieure et la probabilité du symbole r_k détecté à l'itération courante. Le décodage du message de l'exemple précédent peut donc s'effectuer ainsi :

1. Code final : $c = 0.068$, compris dans l'intervalle $[0, 0.2]$. Le symbole 1 est donc : r_0
2. Code actualisé : $c = (0.068 - 0)/0.2 = 0.34$, compris dans l'intervalle $[0.2, 0.4]$. Le symbole 2 est donc : r_1
3. Code actualisé : $c = (0.34 - 0.2)/0.2 = 0.7$, compris dans l'intervalle $[0.4, 0.8]$. Le symbole 3 est donc : r_2
4. Code actualisé : $c = (0.7 - 0.4)/0.4 = 0.75$, compris dans l'intervalle $[0.4, 0.8]$. Le symbole 4 est donc : r_2
5. Code actualisé : $c = (0.75 - 0.4)/0.4 = 0.875$, compris dans l'intervalle $[0.8, 1]$. Le symbole 5 est donc : r_3

En pratique, plusieurs facteurs dégradent l'efficacité de cette stratégie. Lors de la transmission de plusieurs messages successifs, il est en effet nécessaire de rajouter un indicateur de fin de message en plus du code, pour permettre la séparation et le décodage des différents messages. Cela rajoute donc une information à coder en plus du contenu du message. Un autre facteur limitant l'application de cette technique est lié à la précision de la représentation numérique d'un nombre décimal. Lors du découpage des intervalles, il peut arriver que les bornes doivent être codées avec une précision qui dépasse les capacités du système utilisé, rendant impossible le codage du message.

2.2.3 Codage adaptatif

Le codage arithmétique permet, au moins en théorie, d'approcher la borne optimale de compression, sous réserve que les probabilités associées à chaque symbole soient connues. Pour certaines applications au la compression d'image est nécessaire, l'obtention des histogrammes peut être trop coûteuse en temps (applications temps réel), ou tout simplement impossible (transmission continue de flux vidéo). Dans ces situations, des modèles probabilistes a priori sont régulièrement utilisés et permettent d'obtenir des résultats sous-optimaux mais d'autant plus efficaces que le modèle est proche de la distribution réelle des niveaux de gris.

Un moyen simple d'améliorer le modèle probabiliste est d'utiliser le codage adaptatif. Cela consiste à mettre à jour les probabilités au fur et à mesure que les symboles apparaissent. L'estimation des probabilités repose sur :

- la définition d'un voisinage (ou contexte) entourant le pixel à coder, définissant la région dans laquelle le modèle d'évolution des probabilités est établi
- un modèle d'évolution des probabilités en fonction de l'arrivée des nouveaux symboles. Ce modèle est pré-établi et permet d'actualiser les probabilités associées à chaque symbole connaissant les autres symboles du voisinage, soit par exemple : $P(r_{k+1} = 0|r_k)$, $P(r_{k+1} = 1|r_k)$, ...

Les probabilités ainsi mises à jour sont utilisées par l'algorithme de codage. Une des difficultés principales de cette approche est de maintenir la cohérence de l'évolution du modèle probabiliste lors du codage et du décodage. Notons enfin que l'utilisation à grande échelle du codage arithmétique utilisant du codage adaptatif a été freinée du fait d'un grand nombre de brevets associés à cette technique, ce qui explique par exemple pourquoi le format JPEG reste basé sur le codage de Huffman.

3 Réduction de la redondance spatiale

Les techniques exposées dans la section précédente permettaient de réduire la redondance de codage. Elles visaient à optimiser la conversion des symboles en un code le plus court possible en se basant sur les probabilités d'apparition des différents symboles. Il est également possible d'optimiser la longueur du code en exploitant la redondance spatiale présente dans la plupart des images. Les deux méthodes principalement utilisées dans des standards de compression actuels sont exposées ici. Notons que ces méthodes sont, tout comme les codages de Huffman et arithmétiques, des méthodes dite « sans pertes », c'est-à-dire qu'elles permettent de retrouver exactement l'image originale après décodage.

3.1 Codage par plages (run-length)

Le codage par plages, introduit dans les années 1950, est un moyen simple de tirer parti de la redondance spatiale d'une image, et plus précisément, en présence de larges plages de répétition du même symbole. Il consiste à compresser ces plages grâce à une paire de symboles :

- le premier symbole représentant la valeur du symbole (i.e. l'intensité d'un pixel)
- le deuxième symbole représentant le nombre de répétitions consécutives de ce symbole

Il apparaît de façon évidente que le taux de compression sera d'autant plus élevé que le message contiendra de longues plages uniformes. À l'inverse, lorsqu'aucune répétition n'est présente cette technique présentera un taux de compression inférieur à 1 (on parlera d'expansion). Pour ces raisons, ce codage particulièrement efficace pour l'encodage d'images binaires ou contenant un faible nombre de valeurs distinctes.

Exemple 3.1.

Soit un message de 17 symboles à encoder :

AAABBAAAAACAAABBBB

Son codage par plage sera composé des 12 symboles suivants :

A3B2A5C1A2B4

Exemple 3.2.

Dans le cadre de messages binaires à encoder, il est possible de ne coder que la longueur des chaînes de 0 et de 1 en partant du principe que le message démarre par 1.

Soit le message suivant à coder :

0111100111111000000010001

Son codage par plage pourra être restreint aux 9 symboles suivants :

015277131

Notons que le code obtenu peut à son tour être codé par une méthode de codage entropique afin d'optimiser à la fois la redondance spatiale et la redondance de codage.

3.2 Codage LZW

Ce codage - proposé en 1984 par Lempel, Ziv et Welch - fait partie de la famille des méthodes de codage par dictionnaire et ne repose pas sur les probabilités des symboles du message. En amont du processus de codage, un *dictionnaire* contenant tous les symboles de la source est en effet construit (par exemple, les entiers de 0 à 255 pour une image en niveaux de gris codée sur 8 bits). La chaîne de symboles à coder est ensuite parcourue, et chaque nouvelle séquence de symboles rencontrée est ajoutée au dictionnaire et un code lui est attribué. Ce code sera utilisé la prochaine fois que cette même séquence sera rencontrée dans le message. Le dictionnaire est donc enrichi au fut et à mesure de l'arrivée de nouveaux symboles jusqu'à ce que l'intégralité du message soit parcourue.

Cette stratégie sera donc d'autant plus efficace que la redondance spatiale de l'image sera importante. De plus, la taille allouée au dictionnaire est un paramètre important pour cette méthode. Un dictionnaire trop petit ne permettra pas de stocker un grand nombre de séquences, et donc exploitera mal la redondance présente dans l'image. À l'inverse, un dictionnaire trop grand générera des codes trop longs qui pénaliseront le taux de compression obtenu.

Exemple 3.3.

Considérons la région d'une image codée sur 8 bits, contenant un contour vertical :

$$\begin{pmatrix} 20 & 20 & 100 & 100 \\ 20 & 20 & 100 & 100 \\ 20 & 20 & 100 & 100 \\ 20 & 20 & 100 & 100 \end{pmatrix}$$

Cette image sera codée avec la stratégie LZW, en utilisant un dictionnaire de 512 éléments. Les 256 premiers éléments encodent les niveaux de gris potentiellement présents dans l'image, tandis que les 256 derniers éléments ne sont initialement pas utilisés.

Adresse	0	1	...	255	256	...	511
Code	0	1	...	255	-	...	-

Le processus d'encodage se déroule en parcourant les pixels ligne par ligne de gauche à droite. Le tableau ci-dessous résume le processus d'encodage de la région considérée :

Pixel courant	Séquence Reconnue	Valeur encodée	Nouveau code	Adresse
20	20	20	20-20	256
20	20	20	20-100	257
100	100	100	100-100	258
100	100	100	100-20	259
20	20-20	256	20-20-100	260
100	100-100	258	100-100-20	261
20	20-20-100	260	20-20-100-100	262
100	100-20	259	100-20-20	263
20	20-100	257	20-100-100	264
100	100	100	-	-

Le code transmis sera donc composé de 10 symboles de 9 bits, soit 90 bits, à comparer aux 16 symboles de 8 bits, soit 128 bits, du message original. Le taux de compression est donc de $C = \frac{128}{90} \approx 1.4$.

À noter que la génération du dictionnaire se fait au fur et à mesure de l'encodage de l'image. De façon assez remarquable, le décodage ne nécessite pas la transmission du dictionnaire. En effet, le même dictionnaire peut être successivement reconstruit à la réception des données codées et permettre de ce fait le décodage du message initial.

Enfin, un aspect important associé à cette méthode et non abordé dans cet exemple, est la gestion de la taille du dictionnaire. Plusieurs stratégies sont envisageables dans ce contexte :

- Réinitialisation du dictionnaire une fois plein
- Réinitialisation du dictionnaire lorsque il ne permet plus une compression efficace, quand la redondance spatiale devient faible.
- Remplacement de l'élément du dictionnaire le moins fréquemment utilisé ou utilisé il y a le plus longtemps

Le format de compression sans perte DEFLATE utilise une combinaison des codes LZW et Huffman pour optimiser les redondances spatiales et de codage des données à transmettre. Il est aujourd'hui largement utilisé, notamment dans les standards de compression PNG et ZIP.

3.3 Codage prédictif

Le codage prédictif est une méthode de compression pouvant être avec ou sans perte. Elle utilise dans les deux cas la redondance spatiale intrinsèquement présente dans les images naturelles. Pour ce faire, cette méthode se base sur le codage de la **nouvelle information** contenue dans le pixel courant, définie comme la différence entre la valeur du pixel courant et une valeur estimée à partir des pixels précédents.

3.3.1 Codage prédictif sans perte

Les codeurs et décodeurs d'un système de codage prédictif sans perte sont représentés en Figure 5. Les pixels de l'image à encoder sont présentés à la suite les uns des autres pour former la séquence d'entrée présentée à l'entrée de l'encodeur. Les éléments de la séquence sont transmis au prédicteur dont le rôle est de prédire la valeur de $f[n]$ en se basant sur la valeur d'un certain nombre d'éléments précédents. La sortie du prédicteur est ensuite arrondie à l'entier le plus proche pour donner $\hat{f}[n]$, permettant ainsi de définir l'erreur de prédiction :

$$e[n] = f[n] - \hat{f}[n]$$

L'erreur est ensuite codée grâce à une méthode de codage entropique afin d'être transmise au décodeur. Plus le prédicteur est précis, plus l'histogramme de la séquence d'erreur à transmettre sera resserré autour de 0 (faible entropie),

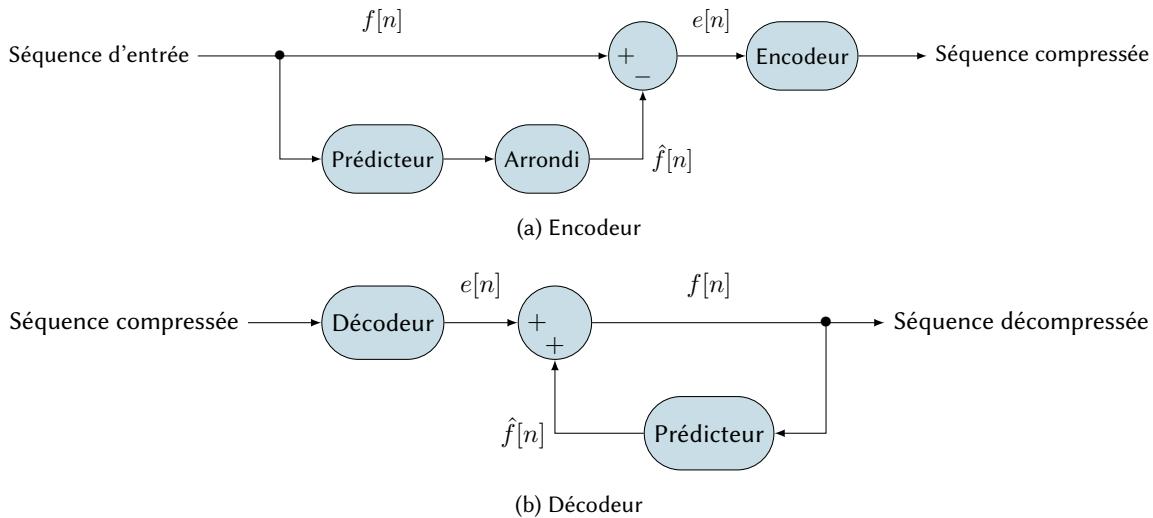


FIGURE 5 : Schémas de codage et décodage d'un système de codage prédictif sans perte

permettant ainsi d'optimiser l'efficacité des méthodes de codage entropique.

L'erreur codée est transmise au décodeur pour décodage afin de procéder à la reconstruction de $f[n]$ grâce à l'utilisation du **même prédicteur que lors de l'encodage**. L'erreur décodée est ainsi additionnée à la valeur estimée pour retrouver l'échantillon original :

$$f[n] = e[n] + \hat{f}[n]$$

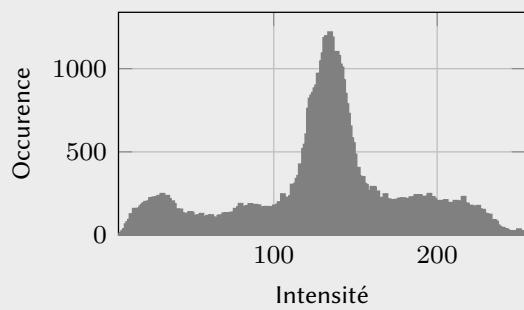
Dans la plupart des cas, le prédicteur consiste en une combinaison linéaire des m précédents échantillons :

$$\hat{f}[n] = \text{int} \left[\sum_{i=1}^m b[i] f[n-i] \right]$$

Remarquons qu'une analogie peut être faite avec la formule de la convolution discrète caractérisant le filtrage (linéaire) d'une séquence $f[n]$ avec un filtre RIF de réponse impulsionnelle $b[n]$. Les coefficients de prédiction (du filtre) pondèrent les échantillons d'entrée précédents, pour former l'estimation (la sortie du filtre) après arrondi. Dans le cas d'une image ou d'une vidéo, les échantillons pris en compte par le prédicteur peuvent appartenir aux colonnes, lignes et/ou trames temporelles précédentes. Notons enfin que la prédiction des m premiers éléments de la séquence n'étant pas possible, ces valeurs sont généralement codées directement.

Exemple 3.4. : codage prédictif à l'ordre 1

Nous cherchons ici à illustrer le processus de codage prédictif sans perte à l'ordre 1 du l'image suivante :

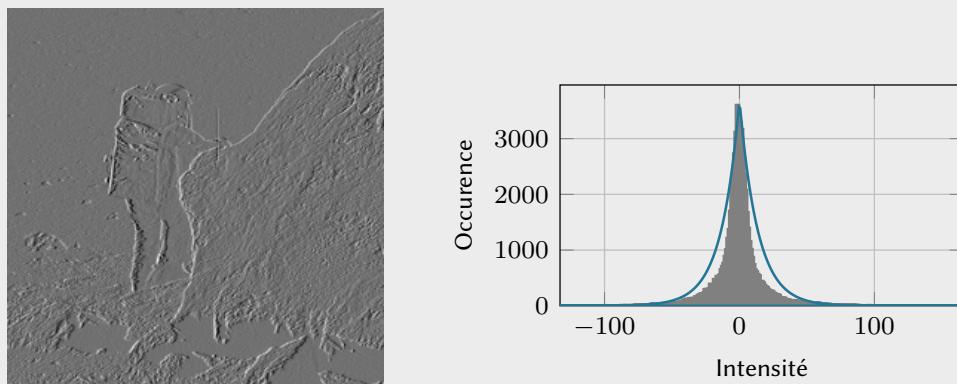


L'entropie de cette image codée sur 8 bits, est évaluée à $H_I = 7.4$ bits, ce qui permet de constater que l'utilisation d'un codage entropique serait pratiquement inefficace pour compresser l'image telle quelle.

Un prédicteur linéaire d'ordre 1 utilise le pixel précédent (colonne précédente) pour prédire le pixel courant. En prenant $b[1] = 1$, l'estimation se résume à considérer que :

$$\hat{f}[i, j] = f[i, j - 1]$$

L'image contenant les erreurs de prédiction est représentée ci-dessous avec son histogramme.



Remarquons la présence de la majorité des coefficients autour de 0, ainsi que la diminution de l'entropie qui devient $H_D = 5.9$ bits. Théoriquement cela permet un gain moyen de 1.5 bits par pixel, soit un taux de compression de 1.23 par rapport au codage uniforme. Notons également qu'il est possible d'obtenir une modélisation relativement précise de l'erreur de prédiction afin d'éviter de transmettre et d'estimer la distribution de celle-ci pour son codage. Une distribution Laplacienne de moyenne nulle dépendant de l'écart-type de $e[n]$ est généralement considérée (représentée en bleu sur la Figure ci-dessus) :

$$p_e(e) = \frac{1}{\sqrt{2}\sigma_e} e^{-\frac{\sqrt{2}|e|}{\sigma_e}}$$

3.3.2 Codage prédictif avec perte

Afin d'améliorer le taux de compression que l'on peut espérer obtenir grâce au codage prédictif, une quantification de l'erreur de prédiction peut être réalisée. Pour cela, l'étape d'arrondi est supprimée et la quantification est insérée entre l'instant auquel l'erreur est formée et son encodage (voir Figure 6). Ainsi, l'erreur de prédiction, notée $\dot{e}[n]$, est restreinte à un nombre fini de valeurs ce qui entraîne à la fois une distorsion et la compression du signal. L'étape de quantification entraîne l'insertion d'une boucle de retour autour du prédicteur, afin de lui passer en entrée la séquence $\dot{f}[n]$ définie telle que :

$$\dot{f}[n] = \dot{e}[n] + \hat{f}[n]$$

Cela permet d'effectuer une prédiction qui tient compte de la quantification effectuée sur l'erreur, et d'éviter ainsi la propagation des erreurs de quantification lors du décodage. Remarquons que la formule ci-dessus est également utilisée pour le décodage de la séquence compressée, permettant ainsi de retrouver la séquence $\dot{f}[n]$ distordue par la quantification.

La modulation delta est une forme simple et largement exploitée de codage prédictif avec perte. L'étape de prédiction est donnée par :

$$\hat{f}[n] = \sum_{i=1}^m b[i] \dot{f}[n - i]$$

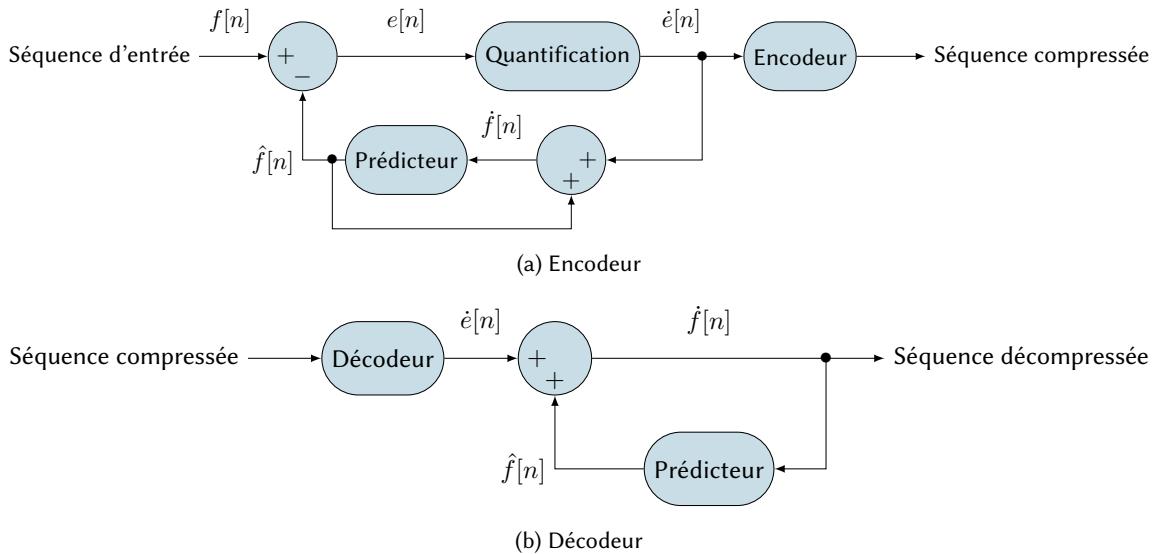


FIGURE 6 : Schémas de codage et décodage d'un système de codage prédictif avec perte

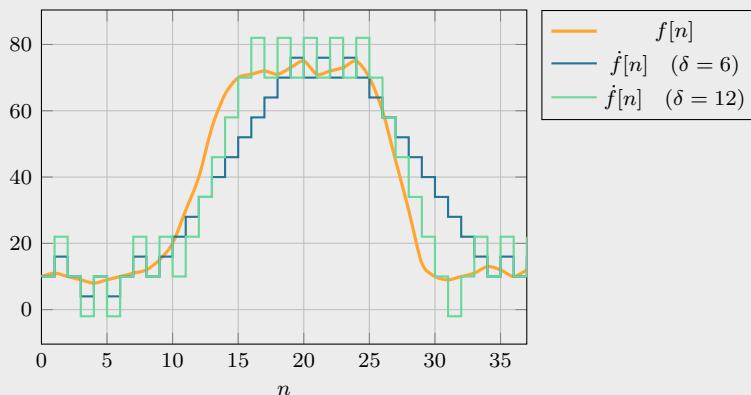
et l'étape de quantification consiste à attribuer deux valeurs à $e[n]$, à savoir :

$$\dot{e}[n] = \begin{cases} +\delta & \text{si } e[n] > 0 \\ -\delta & \text{sinon} \end{cases}$$

avec δ une constante strictement positive. En procédant ainsi, $\dot{e}[n]$ ne peut prendre que deux valeurs et donc être **codée sur un seul bit**, ce qui permet d'atteindre des taux de compression très intéressants. Comme le montre l'exemple suivant, le choix de δ est un compromis entre l'apparition d'un bruit perturbant les régions homogènes et la capacité à représenter des transitions rapides d'intensité.

Exemple 3.5. : modulation delta

Cet exemple illustre l'encodage par modulation delta de la séquence $f[n]$ représentée sur le schéma ci-dessous.



Les paramètres choisis sont $m = 1$ (ordre de la prédiction), et $b[1] = 1$. Notons que pour l'indice $n = 0$, on pose la condition initiale : $\dot{f}[0] = f[0]$. À partir de l'indice $n = 1$, on peut évaluer :

$$\dot{f}[1] = \dot{f}[0] \rightarrow e[1] = 1 \rightarrow \dot{e}[1] = +\delta \rightarrow \dot{f}[1] = \dot{f}[1] + \delta$$

Le tableau suivant contient les premiers échantillons des séquences caractéristiques du processus d'encodage pour $\delta = 12$.

n	$f[n]$	$\hat{f}[n]$	$e[n]$	$\dot{e}[n]$	$\dot{\hat{f}}[n]$
0	10	-	-	-	10
1	11	10	1	12	22
2	10	22	-12	-12	10
3	9	10	-1	-12	-2
4	8	-2	-10	12	10
...

Les séquences reconstruites après décodage pour $\delta = 6$ et $\delta = 12$ sont illustrées sur la Figure ci-dessus et permettent d'apprécier le compromis à réaliser lors du choix de δ . Une faible valeur de δ permet de représenter relativement fidèlement les régions homogènes dans lesquelles l'amplitude de $f[n]$ varie peu, mais ne permet pas de représenter précisément une transition rapide d'intensité (artefact de flou). À l'inverse, un choix de δ élevé engendre une forte distorsion des régions homogènes (artefact de grain), mais permet de représenter fidèlement des changements d'intensité rapides.

3.3.3 Prédicteurs optimaux

Le choix du prédicteur, jusqu'ici peu discuté dans les exemples donnés, peut être optimisé de sorte à minimiser l'erreur quadratique de prédiction :

$$\begin{aligned} E\{e^2[n]\} &= E\left\{\left(f[n] - \hat{f}[n]\right)^2\right\} \\ &= E\left\{\left(f[n] - \sum_{i=1}^m b[i]f[n-i]\right)^2\right\} \end{aligned} \quad (1)$$

Il s'agit donc de trouver les m coefficients $b[i]$ optimaux au sens de l'erreur quadratique de prédiction (appelé DPCM pour *differential pulse code modulation*). Nous nous retrouvons donc dans le cadre d'un **filtrage optimal (Wiener)**, tel qu'abordé dans le module de traitement de signal aléatoire, mettant en œuvre le **principe d'orthogonalité**. Ce dernier stipule que la prédiction du filtre est optimale lorsque l'erreur de prédiction est orthogonale à la séquence d'entrée :

$$\forall n, k > 1, \quad E\{e[n]f[n-k]\} = 0 \quad (2)$$

La condition décrite par l'équation 2 peut être réécrite de sorte à obtenir l'expression des coefficients optimaux en fonction de la fonction d'autocorrélation de f , notée γ_f :

$$\begin{aligned} E\left\{\left(f[n] - \hat{f}[n]\right)f[n-k]\right\} &= 0 \\ \Leftrightarrow E\{f[n]f[n-k]\} - E\{\hat{f}[n]f[n-k]\} &= 0 \\ \Leftrightarrow \gamma_f[k] - E\left\{\sum_{i=1}^m b[i]f[n-i]f[n-k]\right\} &= 0 \\ \Leftrightarrow \gamma_f[k] - \sum_{i=1}^m b[i]E\{f[n-i]f[n-k]\} &= 0 \\ \Leftrightarrow \gamma_f[k] = \sum_{i=1}^m b[i]\gamma_f[k-i] \end{aligned}$$

Cette expression peut se mettre sous forme matricielle (équation de Yule-Walker) :

$$\boldsymbol{\gamma}_f = \mathbf{R}\mathbf{b}$$

où :

$$\gamma_f = \begin{pmatrix} \gamma_f[1] \\ \gamma_f[2] \\ \vdots \\ \gamma_f[m] \end{pmatrix} \quad R = \begin{pmatrix} \gamma_f[0] & \gamma_f[1] & \dots & \gamma_f[m-1] \\ \gamma_f[1] & \gamma_f[0] & \dots & \gamma_f[m-2] \\ \vdots & \vdots & \vdots & \vdots \\ \gamma_f[m-1] & \dots & \dots & \gamma_f[0] \end{pmatrix} \quad b = \begin{pmatrix} b[1] \\ b[2] \\ \vdots \\ b[m] \end{pmatrix}$$

Notons que la même expression peut être obtenue en annulant la dérivée de l'expression de l'erreur quadratique de l'équation 1 par rapport aux coefficients $b[i]$. Dans cette approche variationnelle du problème, l'erreur quadratique de prédiction est vue comme une fonction de coût à minimiser, en trouvant les valeurs des coefficients $b[i]$ qui annulent sa dérivée.

En injectant l'expression connue de la puissance de l'erreur de prédiction donnée par :

$$\sigma_e^2 = \gamma_f[0] - \sum_{i=1}^m b[i]\gamma_f[i]$$

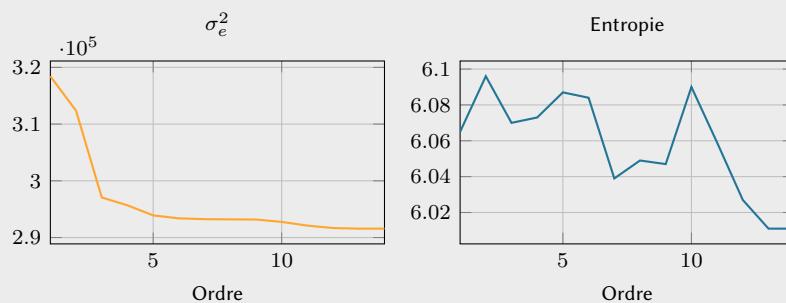
les coefficients de filtre optimaux peuvent être obtenus à partir du système suivant :

$$\begin{pmatrix} \gamma_f[0] & \gamma_f[1] & \dots & \gamma_f[m] \\ \gamma_f[1] & \gamma_f[0] & \dots & \gamma_f[m-1] \\ \vdots & \vdots & \vdots & \vdots \\ \gamma_f[m] & \dots & \dots & \gamma_f[0] \end{pmatrix} \begin{pmatrix} 1/\sigma_e^2 \\ -b[1]/\sigma_e^2 \\ \vdots \\ -b[m]/\sigma_e^2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

En général, la complexité algorithmique engendrée par le calcul de R et de son inverse, ainsi que de γ_f ne compense pas le gain de compression obtenu. Rappelons que les coefficients de filtre optimaux doivent être calculés à l'encodage et au décodage. De plus, si le calcul de l'autocorrélation est évident en 1D, la prise en compte de la connexité en 2D l'est beaucoup moins. Pour ces raisons, l'utilisation de prédicteurs optimaux propres à chaque image (locaux) est donc limitée. Cependant, des coefficients optimaux génériques (globaux) peuvent être estimés à partir de modèles d'images basés sur des sources de Markov 2D, et utilisés tels quels pour l'encodage de n'importe quelle image. L'ordre des filtres prédictifs typiquement utilisés dépassent rarement $m = 3$.

Exemple 3.6. : prédiction optimale

Considérons ici la prédiction d'une ligne d'une image naturelle à partir d'un filtre optimal dont les coefficients sont obtenus à partir de l'inversion du système précédent. Les deux courbes ci-dessous illustrent l'évolution de la puissance de l'erreur ainsi que de l'entropie en fonction de l'ordre du filtre.



Il apparaît assez clairement que si la puissance de l'erreur diminue assez rapidement jusqu'à l'ordre 3, la diminution de l'entropie est bien moins évidente. Notons toutefois que dans le cas du codage prédictif avec perte, la diminution de la variance du bruit permet de limiter la distorsion générée par l'étape de quantification.

4 Compression par transformation

Les méthodes de codage vue jusqu'ici exploitent à la fois la redondance spatiale et des techniques de codage efficaces pour compresser les images **sans perte** (i.e. avec une reconstruction exacte du contenu original via une méthode de décompression), ou **avec perte**. Ces techniques permettent au mieux d'atteindre des taux de compression de l'ordre de 4 pour des images naturelles. Le théorème de Shannon permet d'établir une limite basse théorique sur le nombre de bits moyens nécessaire au codage d'une source d'information. Cette borne est donnée par l'entropie de Shannon, elle-même directement liée à la distribution des probabilités des symboles de la source. Ainsi, une source ayant une distribution uniforme sera difficilement compressible du fait de sa grande entropie. À l'inverse, une source dont seuls quelques symboles ont une forte probabilité sera plus facile à compresser.

Le principe fondamental du codage par transformation est d'exploiter cette dernière propriété en projetant l'image dans un domaine plus favorable à sa compression, c'est-à-dire avec une entropie plus faible, afin d'optimiser son codage. Les transformées typiquement utilisées pour la compression sont les transformées (discrètes) de Fourier, en cosinus ou encore en ondelettes. Ces trois transformées ont la propriété d'avoir la plupart de leurs coefficients à très faible amplitude. Il est donc possible de les quantifier relativement grossièrement (voire de les supprimer) sans altération majeure sur la qualité visuelle de l'image. Cette perte d'information classe ces méthodes dans la catégorie des méthodes de **codage avec perte**, mais elles permettent d'obtenir des facteurs de compression très intéressants (typiquement supérieurs à 10).

4.1 Transformées orthogonales usuelles

Les transformées abordées dans cette section présentent un certain nombre d'avantages dans le cadre de la compression. Leur étude formelle permet à la fois de bien cerner leur comportement et d'identifier leurs avantages respectifs. Pour cette étude, nous nous placerons dans un espace hermitien, qui est un espace vectoriel **complexe**, de **dimension finie** muni d'un **produit scalaire hermitien**. Par souci de clarté, les vecteurs et matrices seront notés en gras (minuscules pour les vecteurs et majuscules pour les matrices). Pour rappel, le produit scalaire hermitien entre deux vecteurs de \mathbb{C}^N s'écrit :

$$\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u}^{*T} \cdot \mathbf{v} = \sum_{i=0}^{N-1} u_i^* v_i$$

où * représente l'opération complexe conjugué.

Les transformations que nous allons étudier ont en commun la propriété de former des **bases orthonormées** dans cet espace. Rappelons qu'en algèbre linéaire, une base est dite orthonormée si et seulement si :

$$\langle b_i, b_j \rangle = \delta_{ij} = \begin{cases} 0 & \text{si } i \neq j \\ 1 & \text{si } i = j \end{cases}$$

C'est à dire que la norme des vecteurs de la base est égale à 1, et que tous ces vecteurs sont orthogonaux entre eux.

L'intérêt d'utiliser des bases orthonormées dans le cadre de la compression est double :

- cela élimine la redondance des données à compresser dans le domaine transformé
- cela permet d'exprimer les coefficients de la transformée comme le produit scalaire (la projection) entre un vecteur quelconque et les vecteurs composants la base (aussi appelés atomes)

Ainsi, la décomposition d'un vecteur \mathbf{f} (e.g. un signal) dans la base $\{b_u\}_{u \in \mathbb{N}}$ s'écrit comme une combinaison linéaire des atomes de la base :

$$\mathbf{f} = \sum_{u=0}^{N-1} t_u \mathbf{b}_u = t_0 \mathbf{b}_0 + t_1 \mathbf{b}_1 + \cdots + t_{N-1} \mathbf{b}_{N-1} \quad (3)$$

Les coefficients (scalaires) t_u , correspondent aux coordonnées de \mathbf{f} dans la base $\{b_u\}$, qu'on appelle régulièrement coefficients de la transformée, ou simplement transformée de \mathbf{f} . Ils peuvent être individuellement obtenus ainsi :

$$t_u = \langle \mathbf{b}_u, \mathbf{f} \rangle = \mathbf{b}_u^{*T} \cdot \mathbf{f} \quad (4)$$

Les notations régulièrement rencontrées en traitement du signal explicitent la dépendance de f à une variable directe (temporelle, spatiale, ...). L'introduction de cette dépendance dans l'équation 3, donne :

$$f[n] = \sum_{u=0}^{N-1} t[u] b[n, u] = \sum_{u=0}^{N-1} t_u b_{n,u} \quad (5)$$

dans laquelle n représente la variable du domaine direct, t la transformée de f (toutes deux de dimension N), u la variable du domaine transformé, et $b_{n,u}$ le **noyau de la transformée inverse** (passage du domaine transformé au domaine direct). Notons que ces noyaux correspondent aux vecteurs de la base associée à cette transformation et qu'ils sont donc orthogonaux entre eux.

De la même manière, en introduisant cette dépendance dans l'équation 4, nous pouvons écrire l'équation permettant d'obtenir les coefficients de la transformation :

$$t[u] = \sum_{n=0}^{N-1} b^*[n, u] f[n] = \sum_{n=0}^{N-1} f[n] b^*[n, u] = \sum_{n=0}^{N-1} f_n b_{n,u}^* \quad (6)$$

où $b_{n,u}^*$ représente le **noyau de la transformée directe** (passage du domaine direct au domaine transformé). Notons que pour une transformée à coefficients réels, les noyaux des transformées directe et inverse sont égaux.

4.1.1 Représentation matricielle

Les transformées abordées ici étant linéaires, il est possible d'exprimer leurs opérateurs matriciels permettant d'appliquer ces transformations. En repartant par exemple de l'équation 5, on obtient :

$$\begin{aligned} \mathbf{f} &= \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_{N-1} \end{pmatrix} = \begin{pmatrix} t_0 b_{0,0} + t_1 b_{0,1} + \cdots + t_{N-1} b_{0,N-1} \\ t_0 b_{1,0} + t_1 b_{1,1} + \cdots + t_{N-1} b_{1,N-1} \\ \vdots \\ t_0 b_{N-1,0} + t_1 b_{N-1,1} + \cdots + t_{N-1} b_{N-1,N-1} \end{pmatrix} \\ \mathbf{f} &= \downarrow \mathbf{z} \begin{pmatrix} \rightarrow u \\ b_{0,0} & b_{0,1} & \dots & b_{0,N-1} \\ b_{1,0} & b_{1,1} & \dots & b_{1,N-1} \\ \vdots & & & \\ b_{N-1,0} & b_{N-1,1} & \dots & b_{N-1,N-1} \end{pmatrix} \begin{pmatrix} t_0 \\ t_1 \\ \vdots \\ t_{N-1} \end{pmatrix} = \mathbf{B}\mathbf{t} \end{aligned} \quad (7)$$

La matrice \mathbf{B} de taille $N \times N$, contient les différents termes du noyau de la transformée inverse. Chaque colonne représente en effet un vecteur de la base orthonormale $\{b_u\}$. De telles matrice sont dites **unitaires** et possèdent la propriété suivante :

$$\mathbf{B}^{*T} \times \mathbf{B} = \mathbf{B} \times \mathbf{B}^{*T} = \mathbf{Id} \quad \text{ou, de façon équivalente : } \mathbf{B}^{-1} = \mathbf{B}^{*T}$$

Nous pouvons déduire de cette propriété que la matrice contenant les termes du noyau de la transformée directe est la matrice conjuguée transposée de \mathbf{B} :

$$\begin{aligned} \mathbf{f} &= \mathbf{B}\mathbf{t} \\ \mathbf{t} &= \mathbf{B}^{*T}\mathbf{f} \end{aligned} \quad (8)$$

Notons que la relation 8 aurait tout aussi bien pu s'obtenir à partir de l'équation 6. Enfin, dans le cas d'une transformée à coefficients réels, les matrices de transformation directe et inverse sont simplement les transposées l'une de l'autre. Remarquons que l'opération de transposition se relie naturellement au changement d'indice de sommation entre les formules de transformations inverse (somme sur u dans l'équation 5), et directe (somme sur n dans l'équation 6).

4.1.2 Extension au cas 2D

Dans le cas de transformées à deux dimensions telles qu'appliquées sur des images, le même type de relation peut être établi. Soient \mathbf{F} et \mathbf{T} deux matrices de \mathbb{C}^{N^2} respectivement dans le domaine direct et transformé, les équations 5 et 6 deviennent :

$$\begin{aligned} F_{i,j} &= \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T_{u,v} b_{i,j,u,v} \\ T_{u,v} &= \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} F_{i,j} b_{i,j,u,v}^* \end{aligned} \quad (9)$$

Étant donné que le noyau $b_{i,j,u,v}$ ne dépend ni de $F_{i,j}$ ni de $T_{u,v}$, les relations précédentes peuvent s'écrire comme une somme de produits matriciels :

$$\begin{aligned} \mathbf{F} &= \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T_{u,v} \mathbf{B}_{u,v} \\ \mathbf{T} &= \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} F_{i,j} \mathbf{B}_{u,v}^{*,T} \end{aligned} \quad (10)$$

où $\mathbf{B}_{u,v}$ désigne la matrice $N \times N$ contenant les termes du noyau de la transformée inverse pour des valeurs de u et v particulières :

$$\mathbf{B}_{u,v} = \underbrace{\downarrow}_{\rightarrow j} \left(\begin{array}{cccc} b_{0,0,u,v} & b_{0,1,u,v} & \dots & b_{0,N-1,u,v} \\ b_{1,0,u,v} & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ b_{N-1,0,u,v} & \dots & \dots & b_{N-1,N-1,u,v} \end{array} \right) \quad (11)$$

Attention, cette matrice ne doit pas être confondue avec l'opérateur matriciel permettant de réaliser le calcul de la transformée inverse. En effet, l'ensemble des matrices $\mathbf{B}_{u,v}$ forme les images de la base définie par le noyau $b_{i,j,u,v}$. La visualisation de l'ensemble des images de la base permet d'obtenir une représentation graphique 2D interprétable des atomes de cette base.

L'opérateur matriciel associé à la transformation 2D peut toutefois être obtenu à partir des matrices $\mathbf{B}_{u,v}$, et en vectorisant les grandeurs 2D \mathbf{F} et \mathbf{T} . En développant l'équation 9, on obtient une somme de N^2 termes :

$$\begin{aligned} F_{i,j} &= T_{0,0} b_{i,j,0,0} + T_{0,1} b_{i,j,0,1} + \dots + T_{1,0} b_{i,j,1,0} + \dots + T_{N-1,N-1} b_{i,j,N-1,N-1} \\ F_{i,j} &= \left(b_{i,j,0,0} \ b_{i,j,0,1} \ \dots \ b_{i,j,1,0} \ \dots \ b_{i,j,N-1,N-1} \right) \left(\begin{array}{c} T_{0,0} \\ T_{0,1} \\ \vdots \\ T_{1,0} \\ \vdots \\ T_{N-1,N-1} \end{array} \right) \ N^2 \end{aligned}$$

On peut généraliser ce calcul et établir la matrice globale de la transformée 2D :

$$\mathbf{F} = \underbrace{\left(\begin{array}{cccccc} b_{0,0,0,0} & b_{0,0,0,1} & \dots & b_{0,0,1,0} & \dots & b_{0,0,N-1,N-1} \\ b_{0,1,0,0} & b_{0,1,0,1} & \dots & b_{0,1,1,0} & \dots & b_{0,1,N-1,N-1} \\ \vdots & & & \vdots & & \vdots \\ b_{1,0,0,0} & b_{1,0,0,1} & \dots & b_{1,0,1,0} & \dots & b_{1,0,N-1,N-1} \\ \vdots & & & \vdots & & \vdots \\ b_{N-1,N-1,0,0} & b_{N-1,N-1,0,1} & \dots & b_{N-1,N-1,1,0} & \dots & b_{N-1,N-1,N-1,N-1} \end{array} \right)}_{\rightarrow (u,v)} \left(\begin{array}{c} T_{0,0} \\ T_{0,1} \\ \vdots \\ T_{1,0} \\ \vdots \\ T_{N-1,N-1} \end{array} \right) \quad (12)$$

Notons que chaque colonne de cette matrice contient les coefficients des matrices $\mathbf{B}_{u,v}$ vectorisées.

4.1.3 Transformée en cosinus discrète

La transformée en cosinus discrète (DCT) est une **transformation orthogonale réelle**, qui projette un vecteur dans une **base de cosinus à différentes fréquences**. Dans sa forme la plus classique (DCT-II), le noyau de la transformée 1D inverse s'écrit :

$$b_{n,u}^C = \alpha(u) \cos\left(\frac{(2n+1)u\pi}{2N}\right) \quad \text{où} \quad \alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{si } u = 0 \\ \sqrt{\frac{2}{N}} & \text{si } u = 1, 2, \dots, N-1 \end{cases}$$

La matrice associée à la transformation inverse 1D, notée B^C , est donc à coefficients réels, orthonormée mais pas symétrique. Du fait de son caractère réel, la matrice de la transformation directe 1D est obtenue en transposant B^C . La figure 7a représente les atomes de la base de la DCT directe 1D pour $N = 16$, dans laquelle l'augmentation de la fréquence des cosinus à chaque ligne est clairement visible. La première ligne représente la fréquence nulle (une constante valant : $\frac{1}{\sqrt{N}}$), l'incrément fréquentiel en fréquence réduite est de : $\frac{1}{2N}$, et la fréquence réduite la plus élevée est de : $\frac{1}{2} - \frac{1}{2N}$ (soit pratiquement la fréquence de Nyquist). On retrouve en effet ces valeurs en ré-arrangeant le terme interne au cosinus :

$$\cos\left(\frac{(2n+1)u\pi}{2N}\right) = \cos\left(\frac{2\pi u}{2N}n + \frac{u\pi}{2N}\right) = \cos\left(2\pi f_n n + \frac{u\pi}{2N}\right) \quad \text{avec} \quad f = \frac{u}{2N}$$

avec n et u allant de 0 à $N - 1$.

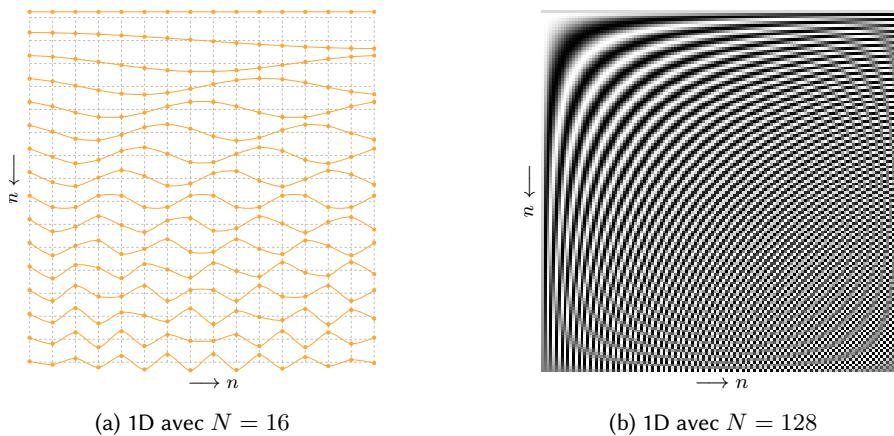


FIGURE 7 : Représentations graphiques des atomes 1D de la DCT directe pour différentes valeurs de N .

La Figure 7b contient les noyaux de la DCT 1D directe pour $N = 128$, et permet d'appréhender graphiquement les coefficients de la matrice de transformation.

La DCT est dite séparable, c'est à dire que son noyau 2D peut s'exprimer comme le produit de deux noyaux 1D dépendant chacun d'une variable transformée. Le noyau de la transformée 2D inverse s'écrit :

$$b_{i,j,u,v}^C = \alpha(u)\alpha(v) \cos\left(\frac{(2i+1)u\pi}{2N}\right) \cos\left(\frac{(2j+1)v\pi}{2N}\right)$$

Les images de la base de la DCT 2D inverse sont montrées en Figure 8a avec $N = 16$. La figure 8b est une représentation visuelle de la matrice de transformation directe, de taille $(N^2 \times N^2)$. Comme dans le cas 1D, la fréquence des oscillations contenues dans ces images de base augmentent en s'éloignant du coin en haut à gauche de l'image. Contrairement au cas 1D, les fréquences évoluent selon les deux dimensions de l'espace (horizontale et verticale).

L'équation 10 nous indique que **toute image de dimension $N \times N$ peut être représentée comme une combinaison linéaire des images de base** de dimension $N \times N$ représentées dans l'image 8a. Les coefficients qui pondèrent ces images de base sont données par les valeurs de la transformée. La Figure 9 illustre l'association entre les images de la base et les coefficients de la DCT qui est effectuée lors de la reconstruction d'une image.

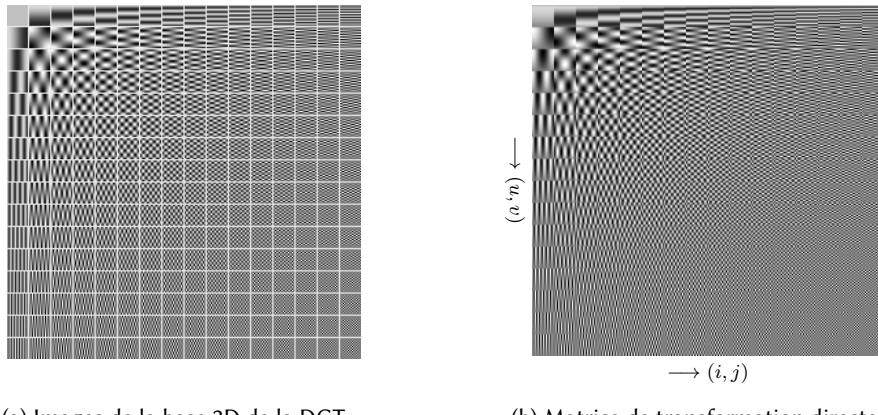


FIGURE 8 : Représentations graphiques de la DCT 2D pour $N = 16$. Les images de la base sont séparées par une grille blanche pour faciliter la différentiation des différents atomes de taille (16×16).

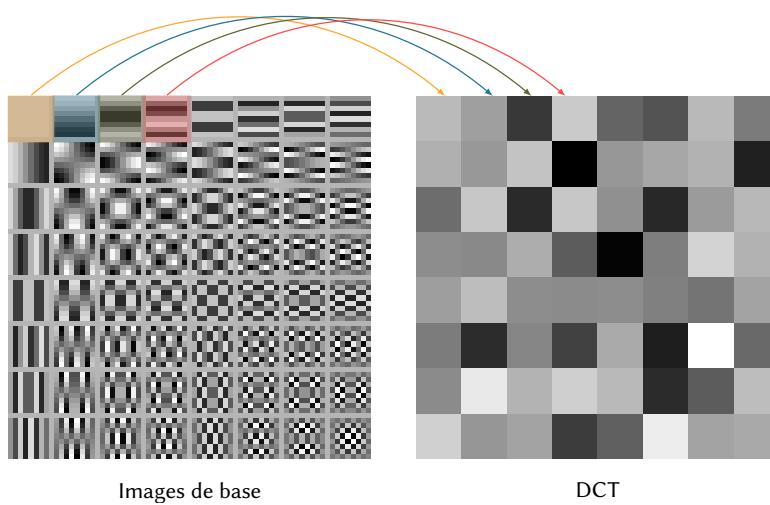


FIGURE 9 : Illustration du processus de reconstruction d'une image (8×8) à partir des images de bases de la DCT 2D (gauche), et des coefficients de la DCT (droite).

4.1.4 Transformée de Fourier discrète

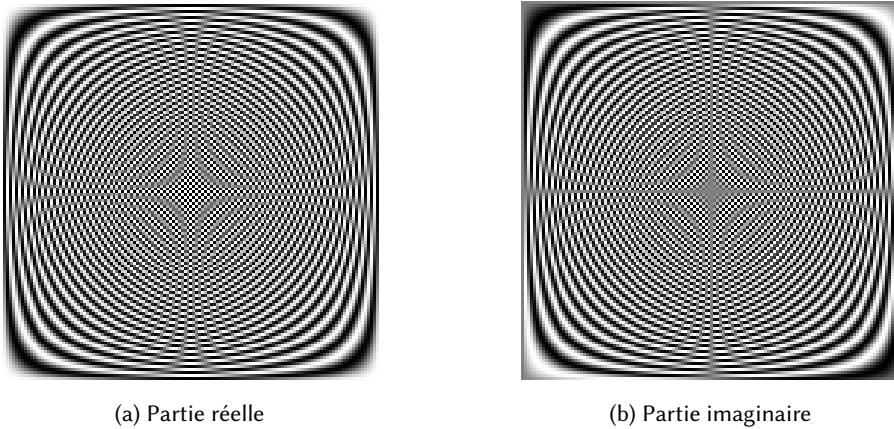
La transformée de Fourier discrète (DFT) est largement utilisée pour l'analyse fréquentielle de signaux numériques. Il s'agit également d'une transformation orthonormée qui décompose les signaux sur une base d'exponentielles complexes. Le noyau de la transformée inverse s'écrit :

$$b_{n,u}^F = \frac{1}{\sqrt{N}} e^{2i\pi n \frac{u}{N}}$$

La matrice associée à cette transformation inverse, notée B_F , est à coefficients complexes, orthonormée, symétrique et unitaire. La Figure 10 représente les noyaux 1D de la DFT directe pour $N = 128$. Remarquons la symétrie non présente pour la DCT.

La fréquence réduite associée au noyau de la DFT inverse est $\frac{u}{N}$, d'où un incrément fréquentiel de $\frac{1}{N}$ d'une ligne à l'autre, alors qu'elle était de $\frac{1}{2N}$ pour la DCT. La fréquence réduite maximale est de : $1 - \frac{1}{N}$. Remarquons que les DCT et DFT parcourront le même intervalle fréquentiel, mais que la résolution fréquentielle de la DCT est deux fois meilleure :

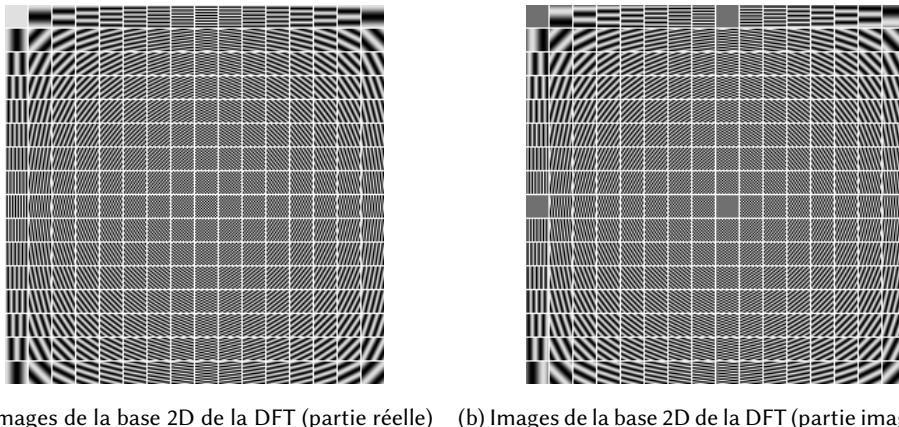
- DCT : la fréquence maximale correspond à l'indice u (dernière ligne)
- DFT : la fréquence maximale correspond à l'indice $\frac{u}{2}$ (ligne centrale)

FIGURE 10 : Représentations graphiques des atomes 1D de la DFT directe pour $N = 128$.

La DFT est une transformation séparable. Son noyau 2D inverse, pour une image de taille ($N \times N$), s'écrit :

$$b_{n,m,u,v}^F = \frac{1}{N} e^{2i\pi n \frac{u}{N}} e^{2i\pi m \frac{v}{N}}$$

Les noyaux 2D de la DFT inverse sont représentés en Figure 11.

FIGURE 11 : Atomes 2D de la DFT inverse séparées par une grille blanche pour faciliter la visualisation ($N = 16$)

4.1.5 Transformée en ondelettes discrète

Les ondelettes ont été introduites dans le cadre de l'analyse multi-résolution des signaux, qui consiste en une décomposition successive à différents niveaux de résolution (échelles). Chaque niveau de résolution est composé d'une **approximation** du signal, obtenue via une fonction d'échelle, et d'une information de **détails**, obtenue via une fonction d'ondelette. Les coefficients de détails représentent la différence entre deux niveaux successifs d'approximation. Ainsi, tout signal discret peut être décomposé comme une somme entre l'approximation à l'échelle la plus grossière, et les coefficients d'ondelettes aux différentes échelles considérées.

$$f[n] = \frac{1}{\sqrt{N}} \left(a_0 \phi[n] + \sum_{j=0}^{J-1} \sum_{k=0}^{2^j-1} d_{j,k} \psi_{j,k}[n] \right)$$

où $N = 2^J$, a_0 est le coefficient d'approximation, $\phi[n]$ la fonction d'échelle, $d_{j,k}$ les coefficients de détails, et $\psi[n]$ la fonction d'ondelette. La fonction d'ondelette $\psi_{j,k}[n]$ peut être dilatée (indice j) en fonction de l'échelle courante, et translatée (indice k) pour effectuer une **analyse locale du signal**. La compacité du support des fonctions d'ondelettes

à la fois dans les domaines direct (spatial, temporel) et transformé est une spécificité de la transformée en ondelettes discrète (DWT), par rapport aux transformées précédentes. La Figure 12 illustre la compacité des supports des fonctions d'ondelettes pour une ondelette de Haar, par opposition aux supports des atomes des DCT et DFT 1D représentés en Figures 7b et 10b.

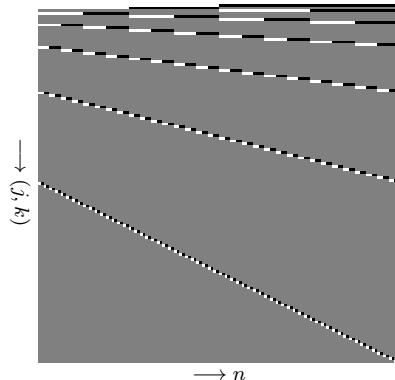


FIGURE 12 : Atomes de la transformée en ondelettes 1D de Haar pour $N = 128$

L'extension au 2D, i.e. aux images, nécessite la génération de fonctions d'échelle et d'ondelettes 2D, qui s'obtiennent à partir des fonctions 1D définies préalablement. La fonction d'échelle est simplement :

$$\phi[n, m] = \phi[n]\phi[m]$$

Les fonctions d'ondelettes s'écrivent :

$$\begin{aligned}\psi^H[n, m] &= \psi[n]\phi[m] \\ \psi^V[n, m] &= \phi[n]\psi[m] \\ \psi^D[n, m] &= \psi[n]\psi[m]\end{aligned}$$

Ces trois fonctions font respectivement apparaître les détails horizontaux (fortes variations d'une ligne à l'autre), verticaux et diagonaux de l'image. L'obtention d'un niveau de résolution supérieur (échelle $j + 1$) se fait en ajoutant l'image d'approximation aux trois images de détails de l'échelle (j). La Figure 13 illustre une décomposition sur 3 échelles d'une image simple.

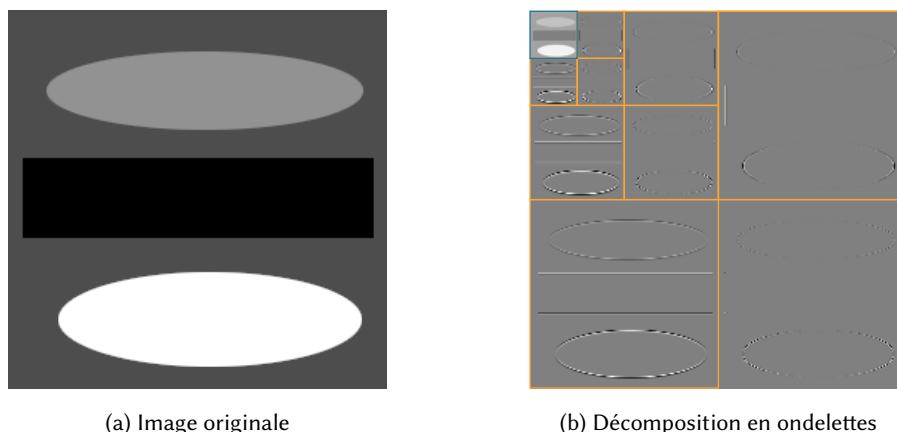


FIGURE 13 : Décomposition en ondelettes (Daubechies-2) sur 3 échelles d'une image simple. Les sous-images encadrées en orange contiennent les coefficients de détails à différentes échelles et orientations, et l'image encadrée en bleu contient l'approximation la plus grossière.

4.2 Processus de compression

Transformer l'image dans un domaine différent ne permet pas en soi de compresser une image. La compression est obtenue lors de la quantification puis du codage des coefficients transformés de l'image. En partant du principe que la plupart de l'énergie de l'image est concentrée sur seulement quelques coefficients transformés, la quantification grossière (voire la suppression) des coefficients de faible amplitude permet d'atteindre des taux de compression importants en limitant la dégradation de l'image. Le schéma de la Figure 14 illustre le principe de compression par transformation. Le codage des coefficients quantifiés pourra par exemple être réalisé avec les techniques de codage étudiées en début de cours.



FIGURE 14 : Schéma du principe de compression par transformation

Le schéma de la Figure 15 illustre lui, le principe de décompression par transformation.

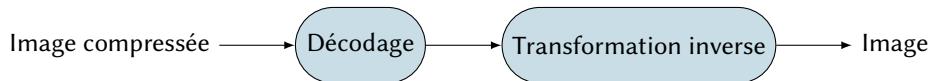


FIGURE 15 : Schéma du principe de décompression par transformation

4.3 Choix de la transformée

Le choix de la transformation dépendra principalement de deux facteurs : la complexité calculatoire de mise en œuvre (ressources de mémoire et de calcul), et l'erreur de reconstruction (la perte d'information) autorisée par l'application. Il est possible de quantifier l'erreur de reconstruction engendrée par le processus de quantification de ses coefficients dans l'espace transformé. Considérons une image \mathbf{I} décomposée dans une base orthonormée quelconque dont les atomes 2D sont notés $\mathbf{B}_{u,v}$, et sa transformée \mathbf{T} :

$$\mathbf{I} = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T_{u,v} \mathbf{B}_{u,v}$$

Appliquons maintenant la technique de quantification suivante : mise à zéro des coefficients $T_{u,v}$ inférieur à un certain seuil. Cette opération est modélisée par l'opérateur χ , défini tel que :

$$\chi_{u,v} = \begin{cases} 1 & \text{si } T_{u,v} \text{ est au-dessus d'un seuil} \\ 0 & \text{sinon} \end{cases}$$

L'image reconstruite après quantification est donc obtenue par :

$$\hat{\mathbf{I}} = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \chi_{u,v} T_{u,v} \mathbf{B}_{u,v}$$

L'erreur quadratique moyenne entre l'image originale et l'image approximée (estimateur de I) est donnée par :

$$\begin{aligned}
 e_q &= E \left\{ \| \mathbf{I} - \hat{\mathbf{I}} \|^2 \right\} \\
 &= E \left\{ \left\| \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T_{u,v} \mathbf{B}_{u,v} - \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \chi_{u,v} T_{u,v} \mathbf{B}_{u,v} \right\|^2 \right\} \\
 &= E \left\{ \left\| \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T_{u,v} \mathbf{B}_{u,v} (1 - \chi_{u,v}) \right\|^2 \right\} \\
 &= E \left\{ \left\langle \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T_{u,v} \mathbf{B}_{u,v} (1 - \chi_{u,v}), \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T_{u,v} \mathbf{B}_{u,v} (1 - \chi_{u,v}) \right\rangle_F \right\} \\
 &= E \left\{ \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} (T_{u,v} (1 - \chi_{u,v}))^2 \right\} \quad (\text{car base orthonormée}) \\
 &= \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} E \left\{ (T_{u,v} (1 - \chi_{u,v}))^2 \right\} \quad (\text{car espérance linéaire})
 \end{aligned}$$

où $\langle ., . \rangle_F$ désigne le produit scalaire de Frobenius entre deux matrices, équivalent au produit scalaire usuel entre les deux matrices vectorisées.

Posons maintenant :

$$\tilde{T}_{u,v} = T_{u,v} (1 - \chi_{u,v})$$

$\tilde{T}_{u,v}$ correspond aux coefficients $T_{u,v}$ non retenus, c'est-à-dire situés en dessous du seuil choisi. En faisant l'hypothèse que ces coefficients sont issus d'un processus aléatoire de moyenne nulle, nous pouvons clore le calcul précédent :

$$\begin{aligned}
 e_q &= \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} E \left\{ (\tilde{T}_{u,v})^2 \right\} \\
 &= \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} Var \left\{ (\tilde{T}_{u,v})^2 \right\} + \left(E \left\{ \tilde{T}_{u,v} \right\} \right)^2 \\
 &= \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \sigma_{\tilde{T}_{u,v}}^2
 \end{aligned} \tag{13}$$

L'équation 13 montre que l'erreur de reconstruction est la somme des variances des coefficients transformés éliminés par le seuillage. Cette erreur est donc directement liée à la capacité de la transformation à minimiser la variance des coefficients de faible amplitude, c'est-à-dire à concentrer le maximum d'énergie sur les coefficients conservés après seuillage. Cette propriété associée à la transformation est appelée **décorrélation**. Notons que maximiser la décorrélation est une étape nécessaire à la **minimisation de l'entropie des coefficients transformés**, qui permettra d'exploiter au mieux les capacités des méthodes de codage pour coder (sans perte) les coefficients quantifiés.

4.4 Décorrélation

La décorrélation obtenue par la transformation peut être visualisée grâce au coefficient de corrélation qui quantifie la corrélation entre deux variables aléatoires. Cette grandeur est comprise entre -1 (anti-corrélation) et 1 (corrélation parfaite). Afin de visualiser la corrélation présente entre les lignes d'une image de taille $(N \times N)$, nous pouvons générer la matrice de corrélation \mathbf{R} également de taille $(N \times N)$. Chaque élément $R_{i,j}$ de cette matrice représente la corrélation entre les lignes i et j de l'image. Elle est donc symétrique.

La Figure 16 représente une image naturelle ainsi que sa matrice de corrélation. La matrice \mathbf{R} contient un grand nombre de coefficients d'amplitude élevée, indiquant un fort niveau de corrélation entre les lignes de l'image.

La Figure 17 représente les coefficients de la DCT de la même image, ainsi que la matrice de corrélation associée. La

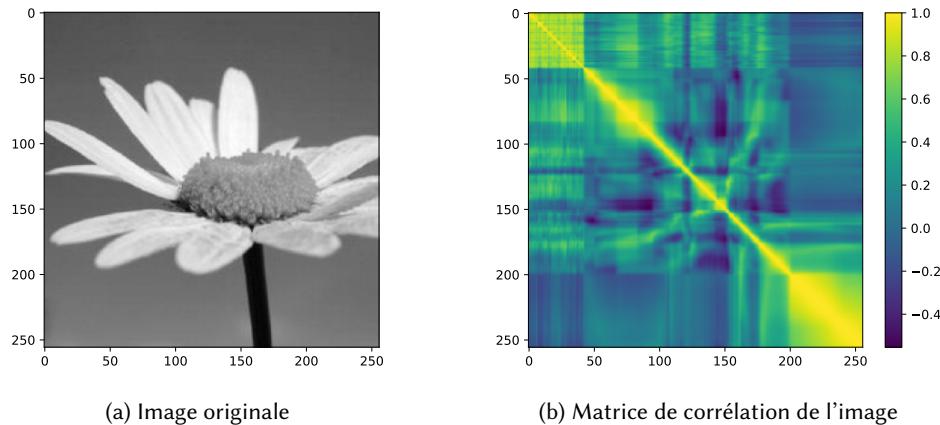


FIGURE 16 : Visualisation de la corrélation présente entre les lignes d'une image naturelle

décorréléation engendrée par la DCT se visualise très bien sur la Figure 17b, où seule la diagonale contient des coefficients de forte amplitude. Notons que les autres transformées présentées dans ce cours possèdent également la propriété de décorréléation.

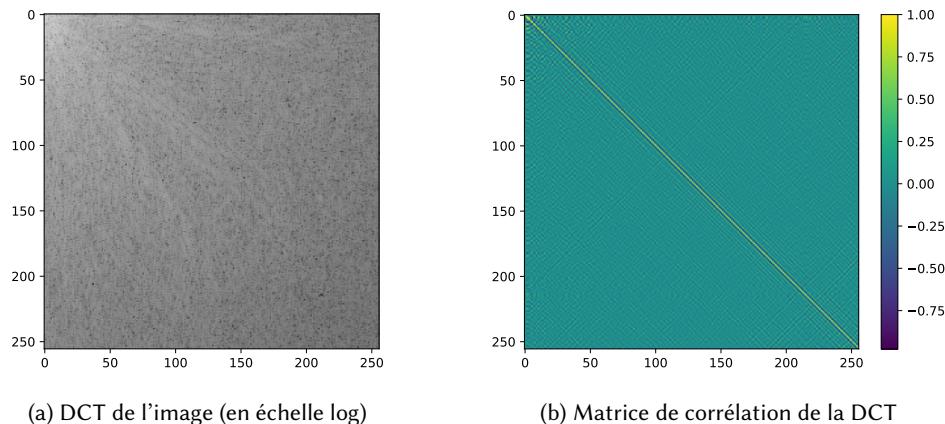


FIGURE 17 : Visualisation de la corrélation présente entre les lignes de la DCT d'une image naturelle

Enfin, les histogrammes de l'image et de sa DCT représentés en Figure 18. La décorréléation se traduit par la répartition de la plupart des coefficients autour de 0, et donc très peu de coefficients sur lesquels est concentrée l'énergie de l'image. De ce fait, la distribution des coefficients de DCT peut être relativement bien approchée par une loi de Laplace pour la plupart des images naturelles. Cette connaissance *a priori* sur la distribution des intensités permet d'établir un modèle global des probabilités associés à chaque symbole, sans besoin d'être adapté à l'image à compresser. Ce modèle pourra être utilisé dans le cadre d'un codage entropique sans perte pour le codage des coefficients la DCT (bien que rarement effectué).

4.5 Troncation et quantification

La dynamique des coefficients transformés est souvent bien plus importante que la dynamique des intensités des pixels de l'image (voir Figure 18b). Selon la transformée choisie, ces coefficients peuvent être par exemple complexes, décimaux ou négatifs. L'entropie associée à ces coefficients est donc initialement beaucoup plus importante, ce qui rend leur codage très peu efficace. Le rôle de la troncation et de la quantification est de réduire la taille du dictionnaire (nombre de symboles de la source) et donc de réduire l'entropie associée aux coefficients transformés pour permettre un codage et donc une compression efficace. Ce sont ces opérations qui entraînent la perte d'information lors du processus de compression.

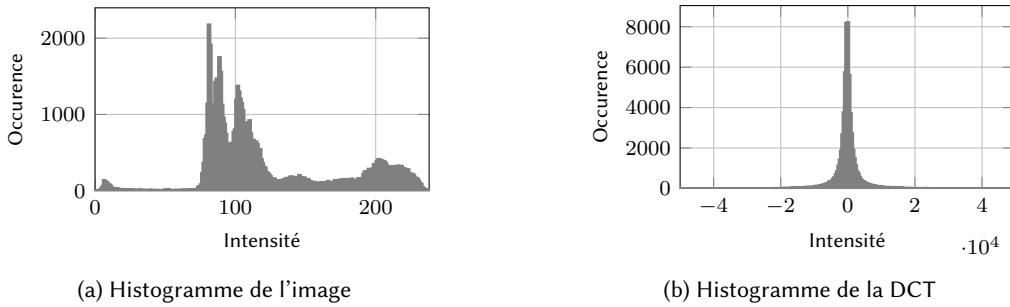


FIGURE 18 : Comparaison des histogrammes de l'image et de sa DCT

La troncation des coefficients peut être faite par exemple en fixant un seuil, où en fixant un nombre de coefficients à conserver. Cette dernière méthode a l'avantage de rendre prédictible le taux de compression obtenu. La Figure 19 montre l'effet de différents de troncation de coefficients de la DCT sur la qualité des images reconstruites. L'erreur moyenne de reconstruction est également indiquée. Ces exemples montrent que les hautes fréquences sont de plus en plus dégradées lorsque le seuil de troncation diminue, l'énergie étant effet principalement répartie sur les coefficients basse fréquence. Les régions texturées et les contours sont donc majoritairement affectés par la troncation. La Figure 20 illustre l'effet de la troncation des coefficients d'ondelettes (décomposition sur 4 niveaux de résolution avec une ondelette de Daubechies-4). Ces exemples montrent que l'erreur moyenne de reconstruction obtenue par la DWT est significativement plus faible que celle obtenue pour la DCT. Le support compact des ondelettes aux différentes échelles considérées permet en effet de mieux préserver les détails locaux de l'image. De plus, l'élimination d'un coefficient de la DCT entraînera la suppression complète de la fréquence correspondante dans l'ensemble de l'image, alors que l'élimination d'un coefficient d'ondelettes affectera uniquement le contenu de son voisinage. De manière plus générale, l'approche multi-résolution permet une meilleure décorrélation des intensités des images que les transformées non-locales. Ces dernières sont en effet optimales pour la représentation de contenus périodiques, pour lesquels seuls quelques coefficients dans le domaine transformé seront non nuls.

La troncation des coefficients transformés, est une étape nécessaire mais pas suffisante à la compression des images. En effet, les coefficients conservés après troncation sont généralement décimaux et nécessitent une forte précision numérique, et donc une forte empreinte en mémoire. En pratique, la troncation et la quantification peuvent être réalisés lors d'une seule et même opération. Les coefficients transformés peuvent être tronqués et quantifiés via l'opération suivante :

$$\hat{T}_{u,v} = \text{int} \left[\frac{T_{u,v}}{Z_{u,v}} \right] \quad (14)$$

où les $Z_{u,v}$ forment une matrice de pondération dont les éléments sont d'autant plus grands que l'on souhaite quantifier grossièrement le coefficient $T_{u,v}$ correspondant. L'image reconstruite sera obtenue par transformation inverse des coefficients quantifiés après multiplication par cette même matrice $Z_{u,v}$ pour rétablir la dynamique initiale :

$$\hat{\mathbf{I}} = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \hat{T}_{u,v} Z_{u,v} B_{u,v}$$

Les coefficients de pondération de la matrice $Z_{u,v}$ sont définies par les standards de compression, et ont été obtenues de manière heuristique de sorte à optimiser la qualité perçue (subjective) des images reconstruites. L'exemple suivant illustre l'application de ces matrices de pondération lors d'une étape de troncation/quantification d'une DCT sur une image de taille (8×8) .

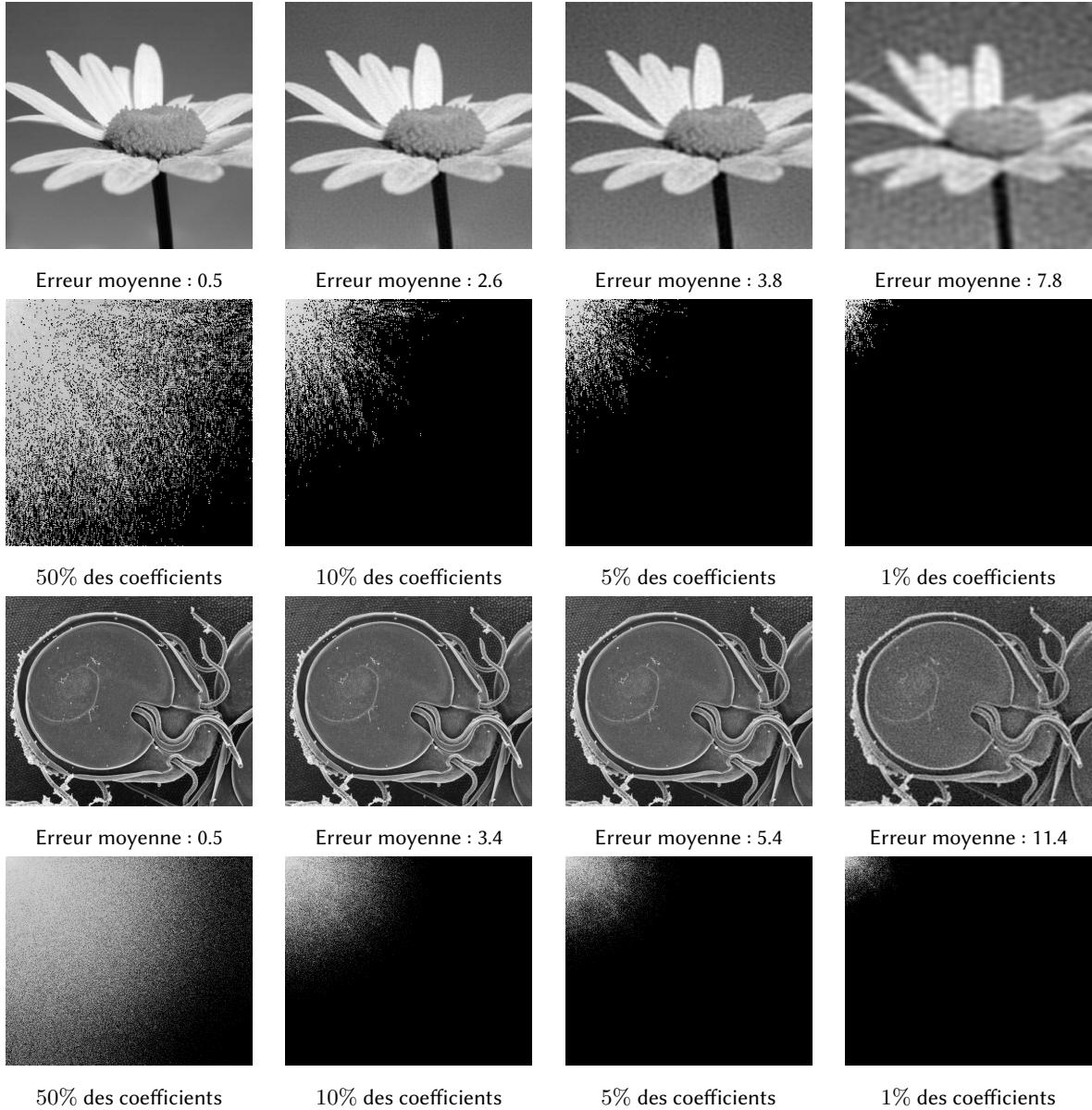


FIGURE 19 : Visualisation de la dégradation sur l'image reconstruite engendrée par différents niveaux de troncation des coefficients de la DCT (conservation des coefficients de plus fortes amplitudes)

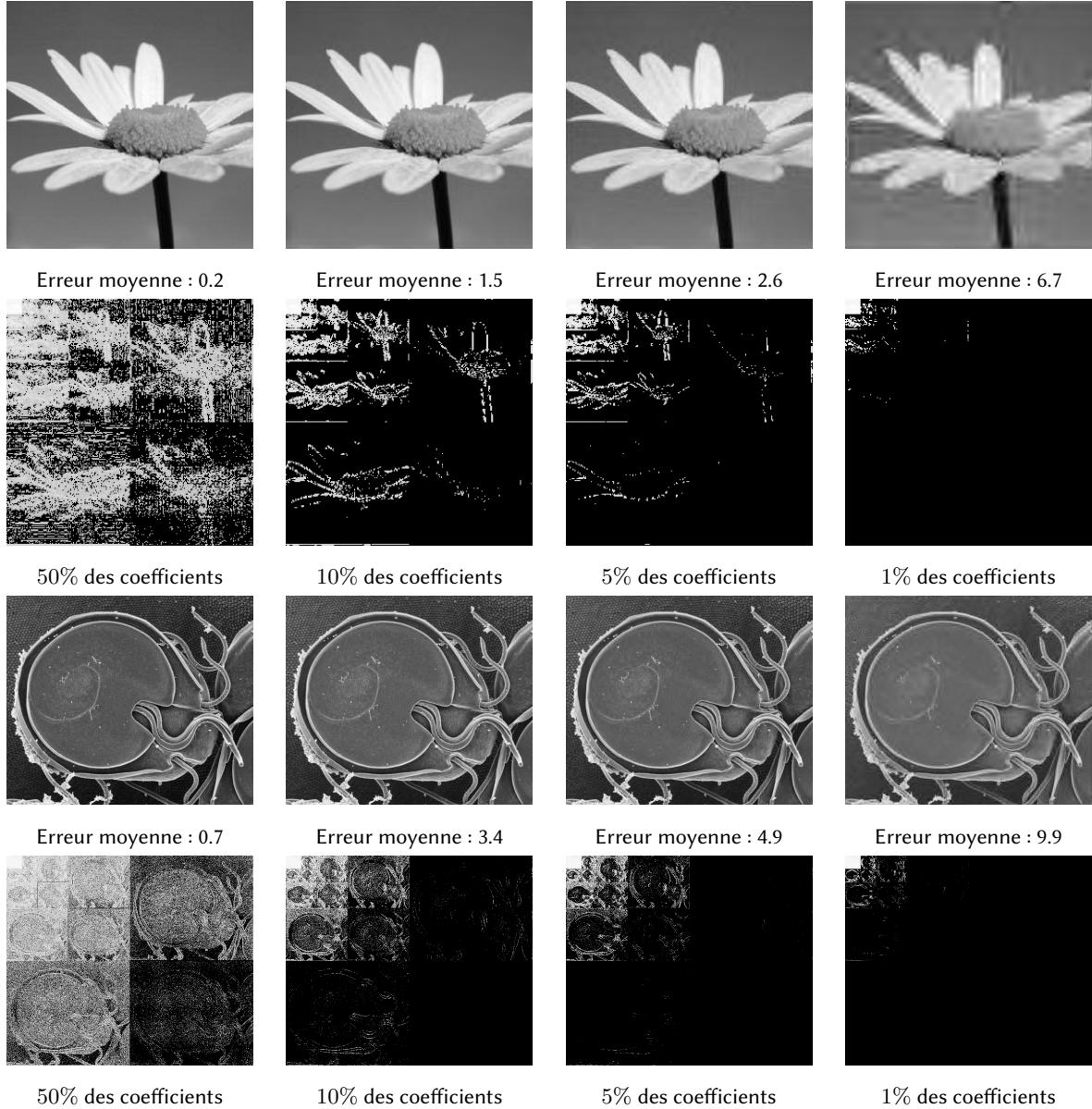


FIGURE 20 : Visualisation de la dégradation sur l'image reconstruite engendrée par différents niveaux de troncation des coefficients de la DWT - Daubechies-4 (conservation des coefficients de plus fortes amplitudes)

Exemple 4.1.

Considérons l'image de taille (8×8) et sa décomposition DCT (arrondie à la deuxième décimale) :

62	70	106	140	138	123	114	120	592.25	-6.07	6.90	38.72	49.00	-5.10	2.28	-2.23
78	45	68	108	124	114	104	126	169.60	-108.16	-29.76	-2.91	17.57	14.33	19.05	1.72
107	57	60	86	99	106	94	114	19.20	-14.24	-77.98	-26.27	13.46	-0.24	-5.92	-6.60
111	76	51	58	80	85	74	84	-5.45	7.31	3.06	-30.53	-25.89	5.10	-7.44	6.73
100	78	50	40	66	67	71	71	23.00	4.64	-18.22	11.00	-6.75	-11.35	-5.44	0.21
83	74	59	43	36	39	54	51	-7.78	21.35	2.74	-10.14	-3.78	1.17	-1.82	-4.13
74	69	44	50	42	20	20	29	16.94	-0.54	-2.92	-3.45	-8.01	6.21	2.98	-1.64
69	66	42	71	81	51	22	24	2.83	-0.02	1.95	1.11	1.98	-0.99	-0.45	1.02

Image originale (I)DCT (T)

L'entropie de I est $H_I = 5.5$ bits. Considérons maintenant la matrice de pondération Z typiquement utilisée par le standard JPEG pour l'encodage de blocs (8×8) , ainsi que les coefficients quantifiés \hat{T} obtenus à partir de la formule de l'équation 14 :

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

Matrice de pondération (Z)

37	-1	0	3	3	0	0	0
15	-9	-2	0	1	0	0	0
2	-1	-5	-1	0	0	0	0
0	0	0	-1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

DCT tronquée/quantifiée (\hat{T})

Cette opération présente le double intérêt de réaliser simultanément la troncation et la quantification des coefficients de la DCT. L'entropie de la matrice des coefficients de la DCT quantifiés est de : $H_{\hat{T}} = 1.1$ bits, soit 5 fois moins que l'image originale.

Lors de l'étape de reconstruction (décodage), l'image est reconstruite en multipliant les coefficients quantifiés par la matrice Z , puis en appliquant la DCT inverse.

592	-11	0	48	72	0	0	0
180	-108	-28	0	26	0	0	0
28	-13	-80	-24	0	0	0	0
0	0	0	-29	0	0	0	0
18	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

($\hat{T} \odot Z$)

62	63	93	138	146	121	114	130
74	58	70	109	127	112	104	114
96	62	50	78	103	100	94	100
112	71	46	61	81	83	85	93
105	73	49	52	59	59	65	78
84	64	50	49	46	39	41	52
71	58	52	55	52	38	28	27
71	60	56	65	66	49	28	15

Image reconstruite (\hat{I})

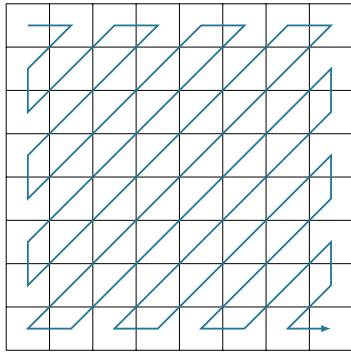
L'image reconstruite est relativement proche de l'image originale, avec une différence moyenne d'intensité d'environ 6.2, ce qui est perceptuellement acceptable pour une image initialement codée sur 8 bits.

4.6 Codage des coefficients

Afin de finaliser le processus de compression, les coefficients transformés tronqués et quantifiés (\hat{T}) doivent être codés. En réalité, deux informations différentes associées à \hat{T} seront codés :

- la **position** des coefficients non nuls
- la **valeur** des coefficients non nuls

Le codage des positions se fait en parcourant la matrice \hat{T} suivant un chemin en zig-zag prédefini, illustré pour une matrice (8 × 8) dans la Figure 21a. Ce trajet permet de chaîner les coefficients transformés d'amplitude supposée décroissante.



(a) Parcours de la matrice des coefficients \hat{T}

1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	0	0	0	0
0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(b) Matrice valant 1 pour les coefficients non nuls de \hat{T}

FIGURE 21 : Convention de codage de la position des indices non nuls de la matrice \hat{T} , appliquée à l'exemple précédent

En reprenant la matrice de l'exemple précédent, la matrice indiquant les valeurs non nulles de \hat{T} peut être construite (Figure 21b). La chaîne codant ces positions est donnée par :

1111101110101010110000001000

Le codage par plage de ce message binaire permet de bénéficier des plages de valeurs constantes, et donne :

5 1 3 1 1 1 1 1 1 2 6 1 39

L'utilisation de la carte de positionnement des coefficients non nuls permet de coder uniquement la valeur de ces 14 coefficients, soit le message suivant :

37 -1 15 2 -9 3 -2 -1 1 -5 3 1 -1 -1

Ces deux messages peuvent être de nouveau codés avec des méthodes de codage entropique afin d'optimiser le taux de compression obtenu. Notons que du fait du caractère local des atomes de la transformée en ondelettes, ces dernières favorisent la présence de longues plages successives de 0, optimisant ainsi le codage des coefficients.

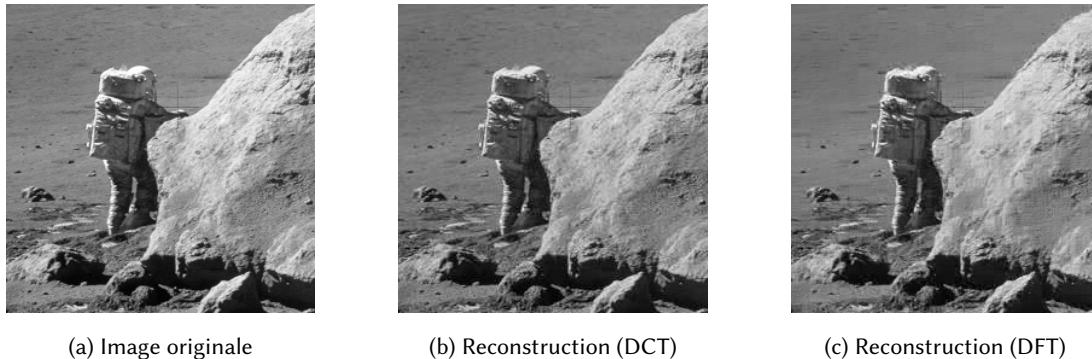
4.7 Compression par blocs

Les transformées non-locales (DCT, DFT) permettent une meilleure décorrélation des coefficients lorsque le contenu de l'image est régulier (stationnaire). C'est à dire que le contenu de l'image fréquentiel est homogène sur son domaine de définition. Cette hypothèse paraît irréaliste dans le cadre d'images naturelles.

C'est pourquoi les méthodes de compression utilisant ces transformées, et en particulier le standard JPEG basé sur la DCT, compressent indépendamment différentes régions de l'image. Ces régions, appelées blocs, sont généralement de taille (8 × 8) et rendent beaucoup plus raisonnable l'hypothèse de stationnarité. Les Figures 22 et 23 compare l'image originale et deux version reconstruites, obtenues à partir de la transformation, quantification puis transformation inverse en DCT et DFT de blocs de taille (8 × 8). Remarquons que ces images ont été compressées de sorte à obtenir une

erreur moyenne de reconstruction équivalente. Cependant, il est à noter qu'à qualité de reconstruction équivalente, il est nécessaire de stocker 2 à 3 fois plus de coefficients de DFT (complexes) que ceux de la DCT. Cette dernière permet en effet une meilleure décorrélation des coefficients, une représentation plus fine du contenu fréquentiel des blocs (voir partie 4.1.4), ce qui justifie son utilisation dans beaucoup d'algorithmes de compression.

En observant attentivement les images reconstruites, il est possible de discerner la frontière des blocs, artefact typique du standard JPEG. Cet artefact sera d'autant plus visible que la quantification des coefficients sera grossière.

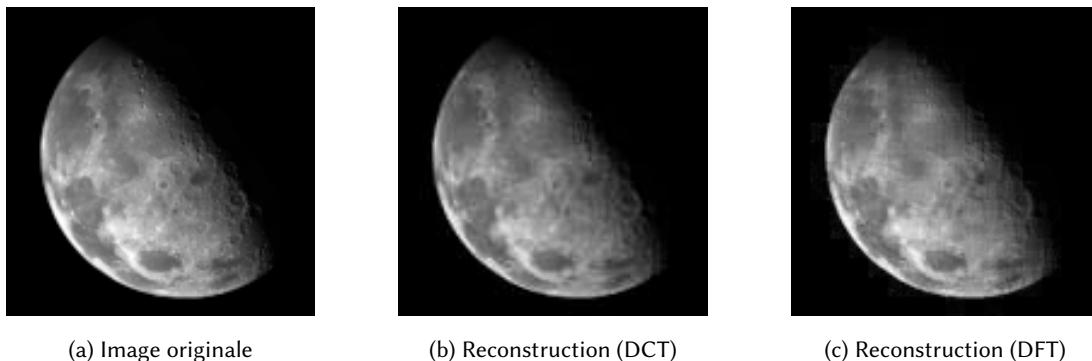


(a) Image originale

(b) Reconstruction (DCT)

(c) Reconstruction (DFT)

FIGURE 22 : Reconstruction par blocs d'une image de taille (256, 256), à partir de différentes transformations (voir la version numérique pour plus de précision)



(a) Image originale

(b) Reconstruction (DCT)

(c) Reconstruction (DFT)

FIGURE 23 : Reconstruction par blocs d'une image de taille (128, 128) à partir de différentes transformations (voir la version numérique pour plus de précision)

Les ondelettes ayant un support spatial compact ne bénéficient pas d'un découpage par blocs. Elles sont utilisées dans le standard de compression JPEG2000, plus performant que le JPEG mais dont l'approche est aujourd'hui moins généralisée.

5 Compression d'images couleur

6 Compression vidéo

Références

- [1] Steven L BRUNTON et J Nathan KUTZ. *Data-driven science and engineering : Machine learning, dynamical systems, and control*. Cambridge University Press, 2022.
- [2] Daniela COLTUC. *Compression d'image*. Poly de cours. CPE Lyon, 2018.

- [3] R. C. GONZALEZ et R. E. Woods. *Digital Image Processing*. 4th Edition. Pearson/Prentice Hall, NY, 2018.
- [4] Erwan LE PENNEC. *Compression d'image*. <https://images.math.cnrs.fr/Compression-d-image.html>. 2006.
- [5] Stéphane MALLAT. *A wavelet tour of signal processing, The sparse way*. 3rd Edition. Elsevier, 2009.
- [6] Gabriel PEYRÉ. *Claude Shannon et la compression des données*. <https://images.math.cnrs.fr/Claude-Shannon-et-la-compression-des-donnees.html>. 2016.