Compression d'images fixes

Introduction

L'homme a toujours voulu découvrir la beauté des planètes, vues de l'espace. Il a donc envoyé des satellites capables de photographier celles-ci. Mais l'un des problème majeur est la transmission de ces photos, du satellite à la terre. Cette communication se fait à l'aide des ondes électromagnétiques.

La transmission des informations est d'autant plus facile que le nombre d'informations est faible. Il est donc avantageux d'en réduire le nombre.
Une technique employées dans de nombreux domaines est la compression des informations.

L'utilisation d'algorithmes de compression d'images permettent en effet, une réduction importante de la quantité de données.

Nous allons étudier un algorithme très répandu et utilisé par de nombreuses personnes : le Jpeg.

Après un bref historique du Jpeg, nous allons présenter son principe et son utilisation . Nous aborderons ensuite, un aspect plus mathématiques, concernant les théorèmes fondamentaux utilisés pour la compression.
Enfin, nous expliquerons la méthode de décompression de ces images. Un applet Java viendra compléter notre dossier, afin de présenter un exemple concret.

1. Vers une standardisation : J.P.E.G.

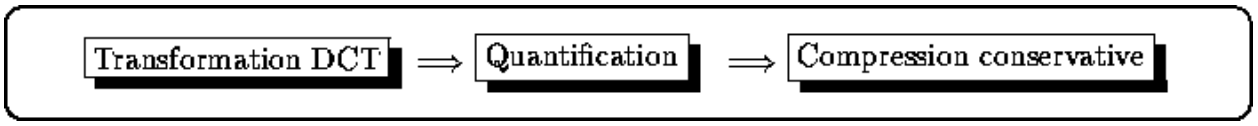
Dans de nombreuses applications : photos satellites, clichés médicaux, photos d'agences de presse, tableaux…, un standard pour archiver ou transmettre une image fixe, en couleur et de bonne qualité est nécessaire. Une première recommandation a été donnée par l'UIT-T en 1980 pour le fac-similé, c'est à dire pour transmettre sur une ligne téléphonique une image en noir et blanc au format A4 (210 x 297 mm²) de l'ISO en environ une minute. La définition est de 4 lignes par mm et de 1780 éléments d'image (pixels, picture elements) en noir et blanc par ligne. Il y a donc environ 2 Mbits à transmettre. Pendant 1 minute à 4800 bauds (bit par seconde), on transmet environ 300kbits. Le taux de compression doit donc être voisin de 7.

Une image fixe de couleur de qualité télévision réclame de l'ordre de 8 Mbits (640 x 480 x 24). Une image de qualité 35 mm en réclame 10 fois plus. Un effort de standardisation a été effectué : l'association de deux groupes de normalisation, le CCITT et l'ISO (Organisation Internationale de Standardisation), supportée par divers groupes industriels et universitaires, donna naissance au **J.P.E.G.**:(**J**oint **P**hotographic **E**xperts **G**roup). Cette norme comprend des spécifications pour le codage conservatif et non-conservatif. Elle a abouti en 1990 à une première phase d'une recommandation ISO / UIT-T. Les contraintes imposées sont importantes. La qualité de l'image reconstruite doit être excellente, le standard adapté à de nombreuses applications pour bénéficier, entre autre, d'un effet de masse au niveau des circuits VLSI nécessaires, la complexité de l'algorithme de codage raisonnable. Des contraintes relatives aux modes d'opérations ont également été rajoutées. Le balayage est réalisé de gauche vers la droite et de haut en bas. L'encodage est progressif et hiérarchique. Ces deux derniers qualificatifs signifient qu'un premier encodage peut fournir une image reconstruite de qualité médiocre mais que des encodages successifs entraîneront une meilleure résolution. Cela est utile, par exemple, lorsque l'on désire visualiser une image sur un écran de qualité médiocre puis l'imprimer sur une bonne imprimante.

2. Principe de la compression JPEG

Le principe de l'algorithme JPEG pour une image à niveaux de gris (une image couleur est un ensemble d'images de ce type), est le suivant. Une image est décomposée séquentiellement en blocs de 8x8 pixels subissant le même traitement. Une transformée en cosinus discrète bi-dimensionnelle est réalisée sur chaque bloc. Les coefficients de la transformée sont ensuite quantifiés uniformément en association avec une table de 64 éléments définissant les pas de quantification. Cette table permet de choisir un pas de quantification important pour certaines composantes jugées peu significatives visuellement, car les informations pertinentes d'une image, caractérisée par son signal bidimensionnel $Img(x,y)$, sont concentrée dans les fréquences spatiales les plus basses. On introduit ainsi un critère perceptif qui peut être rendu dépendant des caractéristiques de l'image et de l'application (taille du document). Une table type est fournie par le standard mais n'est pas imposée.

Un codage entropique, sans distorsion, est enfin réalisé permettant d'utiliser les propriétés statistiques des images. On commence par ordonner les coefficients suivant un balayage en zigzag pour placer d'abord les coefficients correspondant aux fréquences les plus basses. Cela donne une suite de symboles. Le code de Huffman consiste à représenter les symboles les plus probables par des codes comportant un nombre de bits le plus petit possible.



Nous allons détailler chaque partie de la compression JPEG, et en étudier les fondements.

3. Qu'est-ce qu'une image informatique ?

Une image informatique est constituée de points de couleurs différentes. L'association (point,couleur) est appelée pixel. La mémoire utile pour stocker un pixel peut varier de 1 bit (cas des images monochromes) à 24 bits (images en 16 millions de couleurs).

Les informations sur la luminance (paramètre **Y**) et la chrominance (**I** et **Q**) sont des combinaisons linéaires des intensités de rouge (**R**), vert (**G**), et bleu (**B**) :

Y = 0.30 R + 0.59 G + 0.11 B
I = 0.60 R - 0.28 G - 0.32 B
Q = 0.21 R - 0.52 G + 0.31 B

Soit une image 640x480 RGB 24 bits/pixel. Chacune des ces trois variables est reprise sous forme de matrice 640x480. Cependant, les matrices de **I** et de **Q** (info sur la chrominance) peuvent être réduites à des matrices 320x240 en prenant les moyennes des valeurs des pixels regroupés par carré de quatre. Cela ne nuit pas à la précision des infos sur l'image car les yeux sont moins sensibles aux écarts de couleurs qu'aux différences d'intensités lumineuses. Comme chaque point de chaque matrice est une info codée sur 8 bits, il y a chaque fois 256 niveaux possibles (0-255). En soustrayant 128 à chaque élément, on met à zéro le milieu de la gamme de valeur possible :-128 à +127. Enfin chaque matrice est partagée en blocs de 8x8.

4. Transformation DCT : transformée en cosinus discrète bi-dimensionnelle

La clé du processus de compression est la DCT (Discrete Cosine Transform). La DCT est une transformée fort semblable à la FFT : la transformée de Fourier rapide (*Fast Fourier Transform*), travaillant sur un signal discret unidimensionnel. Elle prend un ensemble de points d'un domaine spatial et les transforme en une représentation équivalente dans le domaine fréquentiel. Dans le cas présent, nous allons opérer la DCT sur un signal en trois dimensions. En effet, le signal est une image graphique, les axes X et Y étant les deux dimensions de l'écran, et l'axe des Z reprenant l'amplitude du signal , la valeur du pixel en un point particulier de l'écran. La DCT transforme un signal d'amplitude (chaque valeur du signal représente l' "amplitude" d'un phénomène, ici la couleur) discret bidimensionnel en une information bidimensionnelle de "fréquences".

DCT

$$\mathrm{F(u,v)} = \frac{2}{\mathrm{N}} c(u).c(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \mathrm{Img(x,y)}.\cos\left[\frac{\pi}{N} u\left(x+\frac{1}{2}\right)\right].\cos\left[\frac{\pi}{N} v\left(y+\frac{1}{2}\right)\right]$$

La transformation inverse l'IDCT :

$$Img(x,y) = \frac{2}{\mathrm{N}} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} c(u).c(v).\mathrm{F(u,v)}.\cos\left[\frac{\pi}{N} u\left(x+\frac{1}{2}\right)\right].\cos\left[\frac{\pi}{N} v\left(y+\frac{1}{2}\right)\right]$$

où $\begin{cases} c(0) = (2)^{-1/2} \\ c(w) = 1 \text{ pour w} = 1,2,...,\mathrm{N}-1 \end{cases}$

Le calcul de la DCT ne peut pas se faire sur une image entière d'une part parce que cela générerait trop de calculs et d'autre part parce que le signal de l'image doit absolument être représenté par une matrice carrée. Dès lors, le groupe JPEG impose la décomposition de l'image en blocs de 8 pixels sur 8 pixels. La méthode de compression sera donc appliquée indépendamment sur chacun des blocs. Les plus petits blocs en bordure devront être traités par une autre méthode.

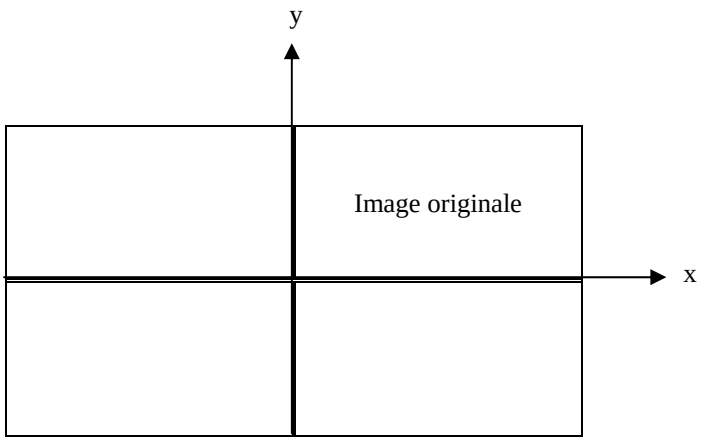
La DCT est donc effectuée sur chaque matrice 8x8 de valeurs de pixels, et elle donne une matrice 8x8 de coefficients de fréquence: l'élément (0,0) représente la valeur moyenne du bloc, les autres indiquent la puissance spectrale pour chaque fréquence spatiale. La DCT est conservative si l'on ne tient pas compte des erreurs d'arrondis qu'elle introduit.

Lorsqu'on travaille avec le signal *Img*(*x*,*y*), les axes *X* et *Y* représentent les dimensions horizontales et verticales de l'image. Lorsqu'on travaille avec la transformée de cosinus discrète du signal *DCT*(*i*,*j*), les axes représentent les fréquences du signal en deux dimensions :

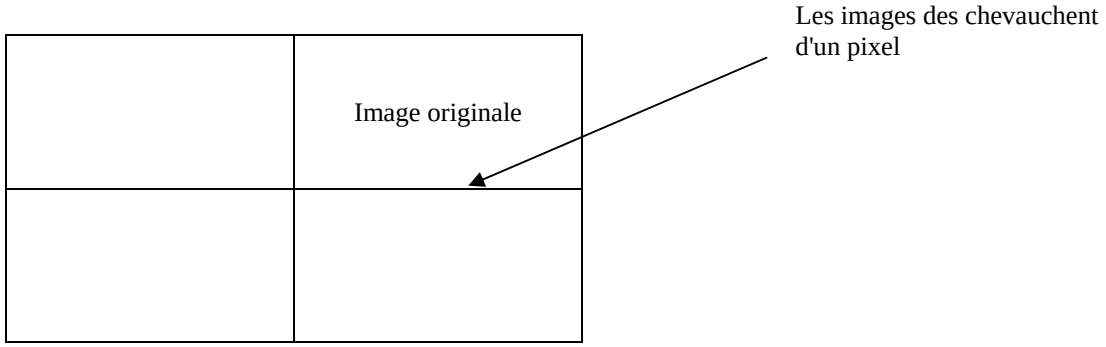
4.1 Comment est formée la DCT bidimensionnelle

La représentation en série de fourrier d'une fonction (à deux variables) continue réelle et symétrique ne contient que les coefficients correspondant aux termes en cosinus de la série. Ce résultat peut être étendu à la transformée de Fourier discrète en faisant une bonne interprétation. Il y a deux manières de rendre une image symétrique. Par une première technique, les images sont dupliquées suivant leurs contours, la seconde méthode : les images sont dupliquées et se chevauchent d'un pixel. Dans la première méthode, nous avons donc une image de 2N x 2N pixels alors que dans la deuxième méthode, nous avons (2N-1)x(2N-1) pixels. C'est la première méthode que nous utiliseront.

Première méthode ⏲



Seconde méthode ⏲



Soit Img(x,y) l'intensité lumineuse de l'image initiale.
L'intensité lumineuse Img' de la nouvelle image ainsi formée vérifie la relation :

$$\text{Img'(x,y)} = \begin{cases} \text{Img}(x, y) & \text{x} \geq 0; \text{y} \geq 0 \\ \text{Img}(-1-x, y) & \text{x} < 0; \text{y} \geq 0 \\ \text{Img}(x, -1-y) & \text{x} \geq 0; \text{y} < 0 \\ \text{Img}(-1-x, -1-y) & \text{x} < 0; \text{y} < 0 \end{cases}$$

Par cette construction, la fonction Img'(x,y) est symétrique par rapport au point x=-1/2 et y=-1/2

Lorsqu'on prend la transformée de Fourier :

$$\text{F}(u, v) = \frac{1}{2\text{N}} \sum_{x=-N}^{N-1} \sum_{y=-N}^{N-1} \text{Img'}(x, y).\exp\left(\frac{-2\pi i}{2N} \left[u\left(x + \frac{1}{2} \right) + v\left(y + \frac{1}{2} \right) \right] \right) \qquad (1)$$

pour u, v=N,…,-1,0,1,…,N-1, comme Img'(x,y) est réelle et symétrique, (1) se réduit à

$$F(u,v) = \frac{2}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} Img(x,y).\cos\left[\frac{\pi}{N}u\left(x+\frac{1}{2}\right)\right].\cos\left[\frac{\pi}{N}v\left(y+\frac{1}{2}\right)\right]$$

L'écriture de la DCT est normalisée :

$$F(u,v) = \frac{2}{N}c(u).c(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} Img(x,y).\cos\left[\frac{\pi}{N}u\left(x+\frac{1}{2}\right)\right].\cos\left[\frac{\pi}{N}v\left(y+\frac{1}{2}\right)\right]$$

La transformation inverse est donnée par

$$Img(x,y) = \frac{2}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} c(u).c(v).F(u,v).\cos\left[\frac{\pi}{N}u\left(x+\frac{1}{2}\right)\right].\cos\left[\frac{\pi}{N}v\left(y+\frac{1}{2}\right)\right]$$

où $\begin{cases} c(0) = (2)^{-1/2} \\ c(w) = 1 \text{ pour } w = 1,2,...,N-1 \end{cases}$

Le coefficient c(w) sert à normer les vecteurs lors de l'écriture matricielle de la DCT.

4.2 Ecriture matricielle et interprétation

On se ramène dans le cadre de la compression jpeg, à N=8

La transformée :

$F(i,j) = \frac{2}{N}c(i).c(j).\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} Img(x,y).\cos\left[\frac{(2x+1)i\pi}{2N}\right].\cos\left[\frac{(2y+1)j\pi}{2N}\right]$ peut s'écrire matriciellement :

La matrice A est composé des intensité Img au point (x,y)

x →

y ↓

$$\begin{bmatrix} Img(0,0) & Img(1,0) & . & . & . & . & Img(6,0) & Img(7,0) \\ Img(0,1) & Img(1,1) & . & . & . & . & Img(6,1) & Img(7,1) \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & Img(x,y) & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ Img(0,7) & . & . & . & . & . & . & Img(7,7) \end{bmatrix}$$

La matrice P est formé par $P = (p_{i,j})$ et $p_{i,j} = c(j).\sqrt{\frac{2}{N}}.\cos\left(\frac{(2.i+1).j\pi}{2N}\right)$ (i l'indice de ligne et j l'indice de colonne)

$$P = \begin{bmatrix} c(0).\sqrt{\frac{2}{N}}.\cos\left(\frac{(2.0+1).0\pi}{2N}\right) & c(1).\sqrt{\frac{2}{N}}.\cos\left(\frac{(2.0+1).1\pi}{2N}\right) & \cdots & c(N-1).\sqrt{\frac{2}{N}}.\cos\left(\frac{(2.0+1).(N-1)\pi}{2N}\right) \\ c(0).\sqrt{\frac{2}{N}}.\cos\left(\frac{(2.1+1).0\pi}{2N}\right) & c(1).\sqrt{\frac{2}{N}}.\cos\left(\frac{(2.1+1).1\pi}{2N}\right) & \cdots & c(N-1).\sqrt{\frac{2}{N}}.\cos\left(\frac{(2.1+1).(N-1)\pi}{2N}\right) \\ c(0).\sqrt{\frac{2}{N}}.\cos\left(\frac{(2.2+1).0\pi}{2N}\right) & c(1).\sqrt{\frac{2}{N}}.\cos\left(\frac{(2.2+1).1\pi}{2N}\right) & \cdots & c(N-1).\sqrt{\frac{2}{N}}.\cos\left(\frac{(2.2+1).(N-1)\pi}{2N}\right) \\ \vdots & \vdots & d_{i,j} & \vdots \\ c(0).\sqrt{\frac{2}{N}}.\cos\left(\frac{(2.(N-1)+1).0\pi}{2N}\right) & c(1).\sqrt{\frac{2}{N}}.\cos\left(\frac{(2.(N-1)+1).1\pi}{2N}\right) & \cdots & c(N-1).\sqrt{\frac{2}{N}}.\cos\left(\frac{(2.(N-1)+1).(N-1)\pi}{2N}\right) \end{bmatrix}$$

soit

$$P = \begin{bmatrix} \frac{1}{\sqrt{N}} & \sqrt{\frac{2}{N}}.\cos\left(\frac{(1).1\pi}{2N}\right) & \cdots & \sqrt{\frac{2}{N}}.\cos\left(\frac{(1).(N-1)\pi}{2N}\right) \\ \frac{1}{\sqrt{N}} & \sqrt{\frac{2}{N}}.\cos\left(\frac{(3).1\pi}{2N}\right) & \cdots & \sqrt{\frac{2}{N}}.\cos\left(\frac{(3).(N-1)\pi}{2N}\right) \\ \frac{1}{\sqrt{N}} & \sqrt{\frac{2}{N}}.\cos\left(\frac{(5).1\pi}{2N}\right) & \cdots & \sqrt{\frac{2}{N}}.\cos\left(\frac{(5).(N-1)\pi}{2N}\right) \\ \vdots & \vdots & d_{i,j} & \vdots \\ \frac{1}{\sqrt{N}} & \sqrt{\frac{2}{N}}.\cos\left(\frac{(2.(N-1)+1).1\pi}{2N}\right) & \cdots & \sqrt{\frac{2}{N}}.\cos\left(\frac{(2.(N-1)+1).(N-1)\pi}{2N}\right) \end{bmatrix}$$

on a $^tP = \begin{bmatrix} \frac{1}{\sqrt{N}} & \frac{1}{\sqrt{N}} & \cdots & \frac{1}{\sqrt{N}} \\ \sqrt{\frac{2}{N}}.\cos\left(\frac{(2.0+1).1\pi}{2N}\right) & \sqrt{\frac{2}{N}}.\cos\left(\frac{(2.1+1).1\pi}{2N}\right) & \cdots & \sqrt{\frac{2}{N}}.\cos\left(\frac{(2.(N-1)+1).1\pi}{2N}\right) \\ \sqrt{\frac{2}{N}}.\cos\left(\frac{(2.0+1).2\pi}{2N}\right) & \sqrt{\frac{2}{N}}.\cos\left(\frac{(2.1+1).2\pi}{2N}\right) & \cdots & \sqrt{\frac{2}{N}}.\cos\left(\frac{(2.(N-1)+1).2\pi}{2N}\right) \\ \vdots & \vdots & \cdots & \vdots \\ \sqrt{\frac{2}{N}}.\cos\left(\frac{(2.0+1).(N-1)\pi}{2N}\right) & \sqrt{\frac{2}{N}}.\cos\left(\frac{(2.1+1).(N-1)\pi}{2N}\right) & \cdots & \sqrt{\frac{2}{N}}.\cos\left(\frac{(2.(N-1)+1).(N-1)\pi}{2N}\right) \end{bmatrix}$

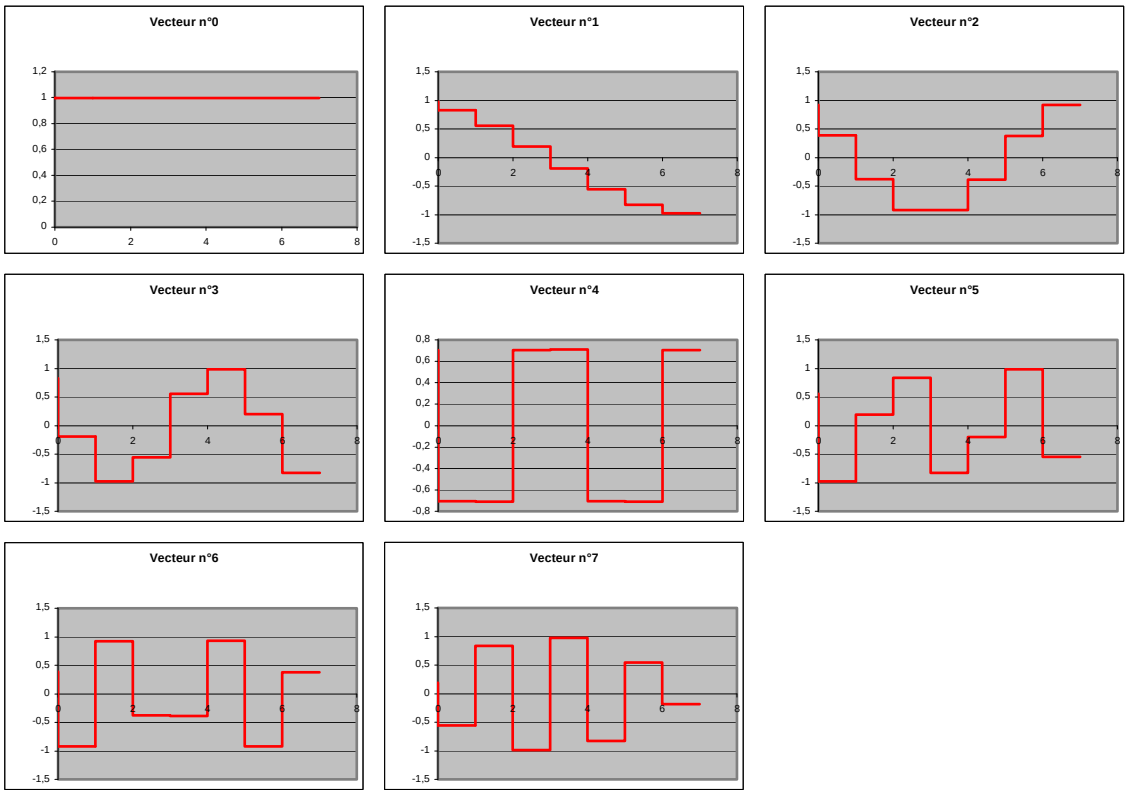
Si on écrit les coefficients F(i,j) dans une matrice avec j l'indice de ligne et i l'indice de colonne, on a

$$F = {}^tP.A.P$$

La matrice P possède des propriétés intéressantes. C'est une matrice orthogonale : les vecteurs sont orthogonaux deux à deux et orthonormés. (d'où l'utilité de c(i) pour ramener la norme du premier vecteur à 1).

Donc $^tP=P^{-1}$. Si on considère que la matrice A représente un endomorphisme de R^8 exprimé dans la base canonique de R^8, l'expression $F = {}^tP.A.P$ correspond à un changement de base avec P la matrice de passage. La DCT inverse en découle simplement : $A = PF{}^tP$.

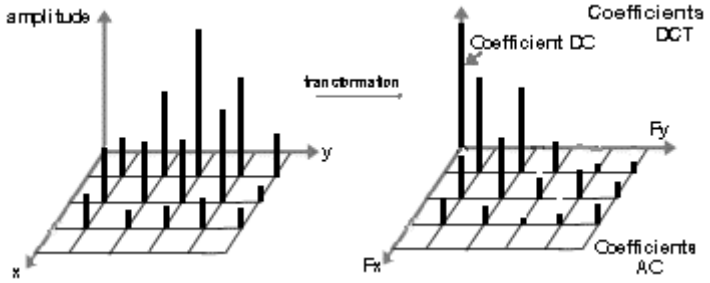
Nous pouvons tracer les fonctions (vecteurs) de base dans le cas N=8 : (nous avons tracé $y = \cos\left(\frac{(2.x+1).n\pi}{2N}\right)$ avec n le numéro du vecteur et x prend des valeurs discrètes : x=0, 1, 2, …, 7)



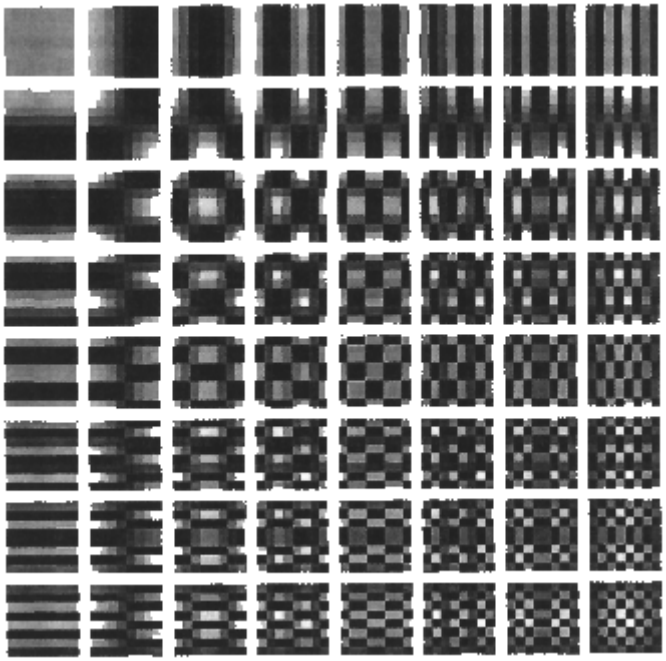
Il a était observé que les fonctions de bases de la transformée en cosinus discrète est une classe des polynômes discrets de Chebyshev.
Donc lorsqu'on passe de la matrice exprimée dans la base canonique à la nouvelle base, on fait une décomposition en fréquence.
Si on considère l'image représenté par la matrice A comme une application linéaire, cette application est décomposable dans la base canonique B des applications linéaires de R^8 dans R^8 avec R^8 associé à sa base canonique c'est à dire $\begin{bmatrix} 1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\end{bmatrix}, \begin{bmatrix} 0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\end{bmatrix}$, etc…

Cette même application maintenant dans la nouvelle base peut être aussi décomposé dans B. Ces 64 vecteurs de bases représentent les applications linéaires de R^8 dans R^8 avec R^8 associé à la nouvelle base. On applique la transformée inverse (soit l'IDCT) sur chacun de ces vecteurs. On a ainsi trouvé une nouvelle base des applications linéaires de R^8 dans R^8 avec R^8 associé à la base canonique.

On a alors trouvé les 64 images de bases (vecteurs de base) qui permettent de décomposer l'image et plus le coefficient dans la matrice F se trouve dans le coin en bas à droite, plus il correspond à des hautes fréquences.



L'image est donc décomposée en une "combinaison linéaire" de ces "images de base" :



4.3 Application pour la compression jpeg

La DCT est une transformation linéaire permettant de disproportionner certains coefficients transformés de telle sorte que leur abandon n'entraîne pas de distorsion significative après reconstruction. De plus, la DCT génère des coefficients réels , les plus petits de ceux-ci étant localisés dans une zone fréquentielle où l'œil a une acuité faible.

Une fois la DCT calculée sur un bloc, nous obtenons une matrice carrée des valeurs pour chacune des fréquences. La figure 1 montre un exemple de compression JPEG sur un bloc de 8x8 pixels à 256 niveaux de gris. Les valeurs de la matrice DCT ont été arrondies à l'entier le plus proche. La composante (0,0) est le coefficient continu (1210). Il représente une valeur ``moyenne'' de la grandeur d'ensemble de la matrice d'entrée. Ce n'est pas exactement la moyenne au sens statistique du terme, l'ordre de grandeur n'étant pas le même, mais c'est un nombre proportionnel à la somme de toutes les valeurs du signal. Les autres valeurs de la DCT représentent des ``écarts'' par rapport à cette moyenne. Les valeurs de la matrice d'indices (0,*j*) (respectivement (*i*,0)) sont les composantes continues le long de l'axe *Y* (resp. *X*) pour la fréquence *j* (resp. *i*) le long de l'axe *X* (resp. *Y*). On remarque une tendance générale des valeurs de la matrice à s'approcher de 0 lorsqu'on s'éloigne du coin supérieur gauche, c'est-à-dire lorsqu'on monte dans les plus hautes fréquences. Cela traduit le fait que l'information effective de l'image est concentrée dans les basses fréquences. C'est le cas de la majorité des images.

Figure 1 Ø

Matrice de pixels d'entrée

140	144	147	140	140	155	179	175
144	152	140	147	140	148	167	179
152	155	136	167	163	162	152	172
168	145	156	160	152	155	136	160
162	148	156	148	140	136	147	162
147	167	140	155	155	140	136	162
136	156	123	167	162	144	140	147
148	155	136	155	152	147	147	136

Matrice DCT

1210	-18	15	-9	23	-9	-14	-19
21	-34	26	-9	-11	11	14	7
-10	-24	-2	6	-18	3	-20	-1
-8	-5	14	-15	-8	-3	-3	8
-3	10	8	1	-11	18	18	15
4	-2	-18	8	8	-4	1	-7
9	1	-3	4	-1	-7	-1	-2
0	-8	-2	2	1	4	-6	0

5. La quantification

Le but de la deuxième étape de la méthode JPEG, l'étape de quantification, est de diminuer la précision du stockage des entiers de la matrice DCT pour diminuer le nombre de bits occupés par chaque entier. C'est la seule partie non-conservative de la méthode (excepté les arrondis effectués). Puisque les informations de basses fréquences sont plus pertinentes que les informations de hautes fréquences, la diminution de précision doit être plus forte dans les hautes fréquences. La perte de précision va donc être de plus en plus grande lorsqu'on s'éloigne de la position (0,0). Pour cela on utilise une *matrice de quantification* contenant des entiers par lesquels seront divisées les valeurs de la matrice DCT. Ces entiers seront de plus en plus grands lorsqu'on s'éloigne de la position (0,0). Elle filtre les hautes fréquences.

La valeur d'un élément de la matrice DCT quantifiée sera égale à l'arrondi, à l'entier le plus proche, du quotient de la valeur correspondante de la matrice DCT par la valeur correspondante de la matrice de quantification. Lors de la décompression, il suffira de multiplier la valeur de la matrice DCT quantifiée par l'élément correspondant de la matrice de quantification pour obtenir une approximation de la valeur de la DCT. La matrice obtenue sera appelée *matrice DCT déquantifiée*.

Bien que la spécification JPEG n'impose aucune contrainte sur la matrice de quantification, l'organisme de standardisation ISO a développé un ensemble standard de valeurs de quantifications utilisables par les programmeurs de code JPEG. Les matrices de quantifications intéressantes sont celles permettant de ``choisir'' la perte de qualité acceptable. Ce choix a été rendu possible grâce aux tests intensifs des matrices. Habituellement, on prend pour matrice de quantification $Q = \left(q_{i,j}\right)$ avec $q_{i,j} = \frac{1}{1 + K.(1 + i + j)}$ avec i l'indice de ligne, j l'indice de colonne et K le facteur de qualité (choisi entre 1 et 25).

On remarque que beaucoup de composantes de hautes fréquences de la matrice quantifiée ont été tronquées à zéro, éliminant leurs effets sur l'image. Par contre, les composantes pertinentes ont été peu modifiées.

Voici un exemple complet de compression et de décompression jpeg :

Matrice de pixels d'entrée

140	144	147	140	140	155	179	175
144	152	140	147	140	148	167	179
152	155	136	167	163	162	152	172
168	145	156	160	152	155	136	160
162	148	156	148	140	136	147	162
147	167	140	155	155	140	136	162
136	156	123	167	162	144	140	147
148	155	136	155	152	147	147	136

Matrice DCT quantifiée

403	-4	2	-1	2	-1	-1	-1
4	-5	3	-1	-1	1	1	0
-1	-3	0	0	-1	0	-1	0
-1	0	1	-1	0	0	0	0
0	1	1	0	-1	1	1	1
0	0	-1	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Matrice DCT

1210	-18	15	-9	23	-9	-14	-19
21	-34	26	-9	-11	11	14	7
-10	-24	-2	6	-18	3	-20	-1
-8	-5	14	-15	-8	-3	-3	8
-3	10	8	1	-11	18	18	15
4	-2	-18	8	8	-4	1	-7
9	1	-3	4	-1	-7	-1	-2
0	-8	-2	2	1	4	-6	0

Matrice DCT déquantifiée (décompression)

1209	-20	14	-9	22	-13	-15	-17
20	-35	27	-11	-13	15	17	0
-7	-27	0	0	-15	0	-19	0
-9	0	13	-15	0	0	0	0
0	13	15	0	-19	21	23	25
0	0	-17	0	0	0	0	0
15	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Matrice de quantification

3	5	7	9	11	13	15	17
5	7	9	11	13	15	17	19
7	9	11	13	15	17	19	21
9	11	13	15	17	19	21	23
11	13	15	17	19	21	23	25
13	15	17	19	21	23	25	27
15	17	19	21	23	25	27	29
17	19	21	23	25	27	29	31

Matrice de pixels de sortie (décompression)

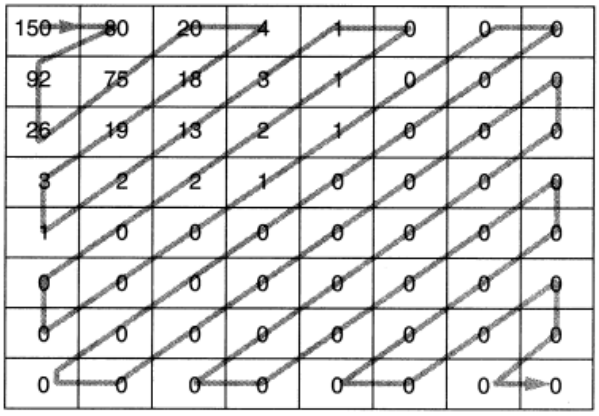
142	143	154	141	133	153	179	179
139	152	129	151	144	154	163	181
150	156	139	166	162	163	154	172
163	145	160	153	151	153	145	154
168	150	156	145	140	139	141	159
148	164	133	164	158	140	136	163
130	159	123	164	165	140	134	145
148	156	140	148	159	146	153	141

6. Codage de la matrice DCT quantifiée par une méthode de compression entropique

Un codage type **Huffman** p159 ou arithmétique continue la compression des données.

La dernière étape de la compression JPEG est le codage de la matrice DCT quantifiée. Ce codage est réalisé sans perte d'informations, en utilisant des mécanismes économes. Puisque les blocs contigus sont très corrélés, on peut coder le coefficient continu en position (0,0) en codant simplement la différence avec le coefficient continu du bloc précédent. Cela produira en général un très petit nombre.

Le codage du reste de la matrice DCT quantifiée va se faire en parcourant les éléments dans l'ordre imposé par une séquence particulière appelée *séquence zigzag* :



Ce qui donne la suite suivante : 150, 80, 92, 26, 75, 20, 4, 18, 19, 3, 1, 2, 13, 3, 1, 0, 1, 2, 2, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, etc.

Cette séquence a la propriété de parcourir les éléments en commençant par les basses fréquences et de traiter les fréquences de plus en plus hautes. Puisque la matrice DCT quantifiée contient beaucoup de composantes de hautes fréquences nulles, l'ordre de la séquence zigzag va engendrer de longues suites de 0 consécutifs. Deux mécanismes sont mis en œuvre pour comprimer la matrice DCT quantifiée. D'une part, les suites de valeurs nulles sont simplement codées en donnant le nombre de 0 successifs. D'autre part, les valeurs non nulles seront codées en utilisant une méthode statistique de type Huffman ou arithmétique.

7. Codage Entropique

7.1 Présentation du problème

Considérons un signal $x(t)$ à temps continu et à bande limitée [-B,+B]. On l'échantillonne à une fréquence supérieure à la fréquence de Nyquist $f_e = 2B$. On obtient, sans perte d'information, un signal à temps discret $X(n)$.

Ce processus est à valeurs continues. Supposons qu'il ait été ensuite quantifié avec une résolution importante. Ce processus aléatoire devient un processus aléatoire à valeurs discrètes, c'est à dire que $X(n)$ prend ses valeurs dans un ensemble fini. En théorie de l'information, on appelle $X(n)$ la source d'information, l'ensemble fini comprenant L_x éléments l'alphabet d'entrée $A_x = \{x^1 ... x^{L_x}\}$ et les éléments x^i les symboles d'entrée ou lettres de l'alphabet. On cherche maintenant à comprimer cette information.

Nous allons expliquer d'abord que, moyennant certaines hypothèses, il est possible de réaliser cette opération sans apporter de distorsion. On parle alors de codage sans bruit, sans perte ou de codage entropique. Malheureusement les taux de compression permis sont généralement insuffisants. On acceptera donc une certaine distorsion. On montrera qu'il existe une fonction appelée fonction débit-distorsion donnant une limite inférieure à la distorsion lorsque l'on impose le débit ou inversement une limite inférieure pour le débit lorsque l'on s'impose la distorsion. On introduira également dans cette partie la notion de capacité d'un canal de transmission de façon à pouvoir énoncer le théorème du codage combiné source-canal.

7.2 Codage sans bruit d'une source discrète sans mémoire :

On suppose, dans une première étape, que le processus $X(n)$ est une suite de variables aléatoires indépendantes et identiquement distribuées prenant ses valeurs dans $A_x = \{x^1 ... x^{L_x}\}$. On parle alors de source discrète sans mémoire. On note $p_x(i) = \mathrm{P_r}\{X(n) = x^i\}$, la distribution des différentes probabilités. Cette distribution est indépendante de l'instant d'observation n puisque le processus aléatoire est stationnaire.

7.2.1 Entropie d'une source

On appelle information propre de l'événement $\{X(n) = x^i\}$ la quantité $I(x^i) = -\log_2 p_X(i)$.
Elle est positive ou nulle. Elle est exprimée en bits/symbole puisque l'on a pris un logarithme de base 2. On appelle information propre moyenne ou entropie de la source $X(n)$ l'espérance de la variable aléatoire $I(x^i)$.

$$H(X) = E\{I(x^i)\}$$

$$H(X) = -\sum_{i=1}^{L_X} p_x(i) \log_2 p_x(i)$$

L'entropie est la quantité d'information qu'apporte, en moyenne, une réalisation de $X(n)$. Elle donne le nombre de bits nécessaires en moyenne pour décrire complètement la source. Elle mesure l'incertitude associée à la source .

Remarque : La notation habituelle $H(X)$ ne signifie pas que H est une fonction des valeurs prises par la variable aléatoire X. L'entropie ne dépend que de la distribution des probabilités $p_x(i)$.

7.2.2 Codage d'une source

L'information à transmettre ou à stocker, modélisée par le processus aléatoire $X(n)$, prend ses valeurs dans un ensemble fini, l'alphabet d'entrée A_X, et on désire représenter (coder) les différents éléments de cet ensemble de façon adaptée aux caractéristiques du canal de transmission et de façon efficace.
De façon adaptée aux caractéristiques du canal de transmission veut dire que la représentation de chaque symbole d'entrée, un mot du code, peut être construite à partir d'éléments d'un autre alphabet, adapté au canal. On supposera par la suite que cet alphabet est composée de deux éléments $A_C = \{a^1, a^2\}$, par exemple les deux symboles binaires habituels 0 et 1.
De façon efficace veut dire que l'on cherche à représenter la source en utilisant le minimum de bits, c'est à dire en minimisant la longueur moyenne des mots du code.

Plus précisément, on appelle codage de la source $X(n)$ une application de l'alphabet A_X dans l'ensemble des suites finies d'éléments de l'alphabet A_C. Le code $C = \{c^1 ... c^{L_X}\}$ est l'ensemble de ces suites. Chaque suite possible $c^i = \{a^{i(1)} ... a^{i(l)}\}$ est un mot du code. Le nombre d'éléments de A_C composant un mot est la longueur l du mot. La longueur moyenne des mots du code est donnée par $\bar{l} = \sum_{i=1}^{L_X} p_x(i) l(c^i)$.

Code instantané uniquement décodable

Remarquons tout d'abord qu'il semble exister un problème associé à un code de longueur variable : faut-il ajouter des séparateurs entre mots du code dans une séquence ? Cela n'est pas nécessaire si le code vérifie la condition dite du préfixe : aucun mot du code ne doit être un préfixe d'un autre mot du code.

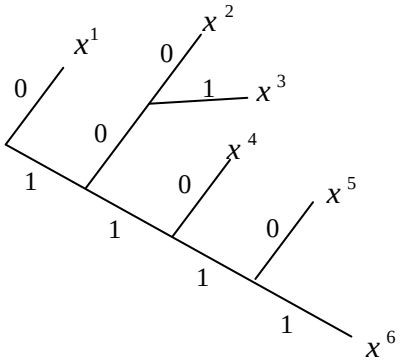
Prenons l'exemple d'une source ne pouvant prendre que six valeurs différentes $\{x^1, ..., x^6\}$. Le code défini par les associations données dans le tableau suivant, est un code valide, on dit uniquement décodable, puisque, recevant par exemple la chaîne binaire 11001110101110, on en déduit la séquence x^4, x^1, x^5, x^3, x^4.

Symboles	x^1	x^2	x^3	x^4	x^5	x^6
Codes	0	100	101	110	1110	1111

Définition d'un code

On constate que le décodage des symboles se fait sans référence aux mots de code futurs. Le code est dit instantané.

Le code précédent vérifie aussi la condition du préfixe. Une façon simple de vérifier la condition du préfixe, ou de construire un code vérifiant cette condition est de tracer un graphe orienté en forme d'arbre binaire, d'étiqueter chaque branche partant d'un nœud par les symboles 0 ou 1 et d'associer un mot du code à chaque nœud terminal en prenant comme mot de code la succession des symboles binaires sur les branches comme le montre la figure.



Arbre binaire associé à un code uniquement décodable

Le nombre de branches entre le nœud initial, la racine, et un nœud terminal, une feuille, spécifie la longueur du code associé.

Inégalité de Kraft

Considérons un code binaire $C = \{c^1 ... c^{L_X}\}$ représentant sans erreur une source. Une condition nécessaire et suffisante pour que ce code vérifie la condition du préfixe est que $\sum_{i=1}^{L_X} 2^{-l(c^i)} \leq 1$ (*)

Où $l(c^i)$ est la longueur du mot de code c^i.

Donnons simplement le principe de la démonstration de la condition nécessaire lorsque L_X est une puissance de 2. Considérons l'arbre complet représentant l'ensemble des L_X mots d'un code C' dont tous les mots du code auraient la même longueur $I_{\max}$ telle que $L_X = 2^{i_{\max}}$.

Ce code vérifie la relation (*) précédemment trouvée puisque $\sum_{i=1}^{L_X} 2^{-i_{\max}} = 2^{i_{\max}} 2^{-i_{\max}} = 1.$

On passe de l'arbre associé au code C^i à l'arbre associé au code C en élaguant un certain nombre de branches et en en créant de nouvelles. Elaguer des branches ne modifie pas le premier membre de la relation (*) puisque l'on sera toujours amené à remplacer deux termes de la forme 2^{-i} par 2^{-i+1}. Il en sera de même si l'on crée de nouvelles branches puisque l'on remplacera 2^{-i} par $2 \times 2^{-i+1}$.

Code Optimal

Relâchons la contrainte suivant laquelle les quantités $l(c^i)$ doivent être des entiers et remplaçons le signe d'inégalité par une égalité dans (*). Le code optimal est celui qui minimise la longueur moyenne

$$\bar{l} = \sum_{i=1}^{L_X} p_x(i) l(c^i)$$

Sous la contrainte

$$\sum_{i=1}^{L_X} 2^{-i(c^i)} = 1.$$

Introduisons les multiplieurs de Lagrange et écrivons pour $i = 1 ... L_X$

$$\frac{\partial}{\partial l(c^i)}\left[\sum_{j=1}^{L_X} p_x(j)l(c^j) + \lambda\sum_{j=1}^{L_X} 2^{-i(c^j)}\right] = 0$$

On obtient $p_x(i) - \lambda \log_e 2.2^{-l(c^i)} = 0$

Soit $2^{-l(c^i)} = \dfrac{p_x(i)}{\lambda \log_e 2}$

Comme $\sum_{i=1}^{L_X} 2^{-l(c^i)} = \dfrac{1}{\lambda \log_e 2}\sum_{i=1}^{L_X} p_x(i) = 1$

Cela impose la valeur de la constante $\lambda \log_e 2 = 1$. On obtient $2^{-l(c^i)} = p_x(i)$

$$l(c^i) = -\log_2 p_x(i)$$

La longueur moyenne correspondant au code optimal est donnée par

$$\bar{l} = \sum p_x(i)l(c^i) = -\sum_{i=1}^{L_X} p_x(i)\log_2 p_x(i)$$

$$\bar{l} = H(X)$$

Parmi tous les codes vérifiant la condition du préfixe, celui qui minimise la longueur moyenne des mots du code a une longueur moyenne égale à l'entropie de la source. L'entropie $H(X)$ apparaît donc comme une limite fondamentale pour représenter sans distorsion une source d'information.

7.2.3 Théorème du codage sans bruit d'une source discrète sans mémoire

Le développement qui suit a simplement pour but de préciser le résultat précédent lorsque l'on impose aux longueurs des mots du code d'être des valeurs entières.

Proposition 1

Si $p(1)...p(L)$ et $q(1)...q(L)$ sont deux distributions de probabilité quelconques, alors

$$\sum_{i=1}^{L_X} p(i)\log_2 \frac{q(i)}{p(i)} \le 0$$

L'égalité a lieu si et seulement si $p(i) = q(i)$ quel que soit $i = l,...L$.

En effet, on sait que, pour tout x positif,

$$\log_2 x \le (x-1)\log_2 e\,.$$

Donc $$\log_2 \frac{q(i)}{p(i)} \le \left(\frac{q(i)}{p(i)} - 1\right)\log_2 e$$

Ou $$\sum_{i=1}^{L} p(i)\log_2 \frac{q(i)}{p(i)} \le \sum_{i=1}^{L} p(i)\left(\frac{q(i)}{p(i)} - 1\right)\log_2 e = 0\,.$$

L'égalité est atteinte si et seulement si $q(i)/p(i) = 1$ quel que soit i.

L'expression $D(p\|q) = \sum_{i=1}^{L} p(i)\log_2 \frac{p(i)}{q(i)}$ s'appelle l'entropie ou distance de Kullback-Leibler entre deux distributions de probabilité. Elle est toujours positive ou nulle. Elle s'interprète comme une mesure de distance entre deux distributions de probabilité bien que cela ne soit pas, à proprement parler, une distance puisque ce n'est pas une expression symétrique et qu'elle ne respecte pas l'inégalité triangulaire.

Proposition 2

Tout codage de la source $X(n)$ par un code instantané uniquement décodable entraîne une longueur moyenne vérifiant $H(X) \le \bar{l}$.

En effet, tout code instantané uniquement décodable vérifie l'inégalité de Kraft $\sum_{i=1}^{L_X} 2^{-l(c^i)} \le 1.$

On crée une nouvelle distribution de probabilité de la forme $q(i) = a.2^{-l(c^i)}$

avec $a \ge 1$ puisque $\sum_{i=1}^{L_X} q(i) = 1 = a\sum_{i=1}^{L_X} 2^{-l(c^i)}$.

La proposition 1 dit que $\sum_{i=1}^{L_X} p_x(i)\log_2 a\frac{2^{-l(c^i)}}{p_x(i)} \le 0$

Soit $$-\sum_{i=1}^{L_X} p_x(i)\log_2 p_x(i) - \sum_{i=1}^{L_X} p_x(i)l(c^i) + \log_2 a \le 0$$

$$H(X) \le \bar{l} - \log_2 a$$

On obtient donc la formule désirée puisque $\log_2 a \ge 0$.

Proposition 3

Il existe un code instantané uniquement décodable vérifiant $\bar{l} \le H(X) + 1.$

En effet, choisissons un code tel que $-\log_2 p_x(i) \le l(c^i) \le -\log_2 p_x(i) + 1.$

A partir de la première inégalité $-l(c^i) \le \log_2 p_x(i)$, on obtient $2^{-l(c^i)} \le p_x(i)$

$$\sum_{i=1}^{L_X} 2^{-l(c^i)} \le \sum_{i=1}^{L_X} p_x(i) = 1$$

On en déduit l'existence d'un code ayant cette distribution des longueurs. A partir de la deuxième inégalité, on obtient : $$\sum_{i=1}^{L_X} p_x(i)l(c^i) < -\sum_{i=1}^{L_X} p_x(i)\log_2 p_x(i) + 1$$

$$\bar{l} < H(X) + 1.$$

Théorème

Pour toute source discrète sans mémoire, $X(n)$, il existe un code instantané représentant exactement cette source et uniquement décodable vérifiant $H(X) \le \bar{l} \le H(X) + 1$ (**)

Où $H(X)$ est l'entropie de la source et l le longueur moyenne du code.

7.2.4 Construction d'un code

Code de Shannon

La façon la plus simple de procéder est de choisir $l(c^i) = \left[-\log_2 p_x(i)\right]$

Où $\left[x\right]$ représente le plus petit entier supérieur ou égal à x. On a $\bar{l} = \sum_{i=1}^{L_X} p_x(i)\left[\log_2 p_x(i)\right]$

$$\bar{l} \le -\sum_{i=1}^{L_X} p_x(i)\log_2 p_x(i) + \sum_{i=1}^{L_X} p_x(i)$$

$$\bar{l} \le H(X)+1$$

Comme

$$2^{-\left[-\log_2 p_x(i)\right]} \le 2^{\log_2 p_x(i)}$$

$$\sum_{i=1}^{L_X} 2^{-l(c^i)} \le \sum_{i=1}^{L_X} p_x(i)$$

$$\sum_{i=1}^{L_X} 2^{-l(c^i)} \le 1$$

on en déduit qu'il existe un code instantané vérifiant la condition du préfixe.

Algorithme de Huffman

L'algorithme de Huffman consiste à construire progressivement un arbre binaire en partant des nœuds terminaux.

- On part des deux listes $\left\{x^1 ... x^{L_X}\right\}$ et $\left\{p_x(1) ... px(L_X)\right\}$.
- On sélectionne les deux symboles les moins probables, on crée deux branches dans l'arbre et on les étiquette par les deux symboles binaires 0 et 1.
- On actualise les deux listes en rassemblant les deux symboles utilisés en un nouveau symbole et en lui associant comme probabilité la somme des deux probabilités sélectionnées.
- On recommence les deux étapes précédentes tant qu'il reste plus d'un symbole dans la liste.

On montre que cet algorithme est l'algorithme optimal. Pour aucun autre code uniquement décodable, la longueur moyenne des mots du code est inférieure.

Premier Exemple

Prenons l'exemple d'une source ne pouvant prendre que six valeurs différentes et supposons connues les probabilités. Elles sont données dans le tableau suivant :

Symboles	x^1	x^2	x^3	x^4	x^5	x^6
Probabilités	0,5	0,15	0,17	0,08	0,06	0,04

Probabilités associées aux six événements $\left\{X(n) = x^i\right\}$

L'entropie de cette source est égale à 2,06 bits. Le code de Shannon entraîne une longueur moyenne égale à 2,28 bits. L'algorithme de Huffman fournit l'arbre binaire schématisé ci dessous :

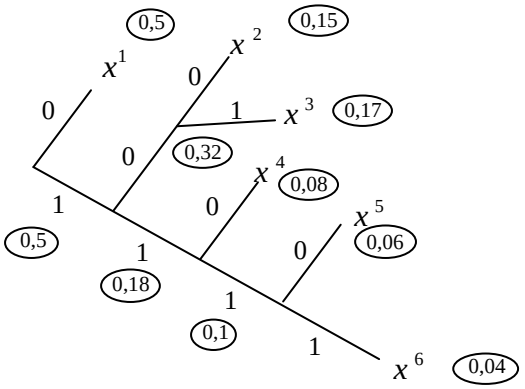


Illustration de l'algorithme de Huffman

La table de codage est indiquée dans le tableau de la partie **Code instantané uniquement décodable**. La longueur moyenne est égale à 2,1 bits, valeur très voisine de la limite théorique.

7.2.5 Généralisation

La double inégalité (**) du théorème précédent, est trop imprécise car la valeur de $H(X)$ est généralement faible. Pour diminuer cette imprécision, on forme un vecteur aléatoire, noté X_N, en regroupant N variables aléatoires $X(mN)...X(mN+N-1)$ et on cherche à associer à toute réalisation possible de ce vecteur un mot du code. Cela correspond, dans un cas, à une application de l'ensemble produit $A_X \times ... \times A_X$ dans l'ensemble des suites finies. On conserve, dans ce paragraphe, l'hypothèse que la source est sans mémoire.

Théorème

On montre que si on regroupe N symboles de la source et si on lui associe un mot du code c^i de longueur $l(c^i)$, alors il existe un code tel que la longueur moyenne $\bar{l} = \sum_{X_N} \Pr(X_N) l(c^i)$ vérifie

$$H(X) \le \frac{\bar{l}}{N} < H(X) + \frac{1}{N}$$

Le rapport l/N représente le nombre moyen de bits par symbole nécessaire et suffisant pour pouvoir représenter exactement la source.
Il existe un autre théorème consistant à supposer que la longueur de tous les mots du code est identique et à permettre à N de varier. Ce théorème dit que quel que soit $\varepsilon > 0$, il existe un code tel que si

$$\frac{l}{N} \ge H(X) + \varepsilon$$

alors la probabilité p_e pour qu'à une séquence $X(mN)...X(mN+N-1)$ on ne puisse pas lui associer un mot du code, peut être rendue arbitrairement petite pourvu que N soit suffisamment grand. Ce théorème dit également que si

$$\frac{l}{N} \le H(X) - \varepsilon$$

il n'existe pas de code pour lequel la probabilité p_e puisse être rendue arbitrairement petite.

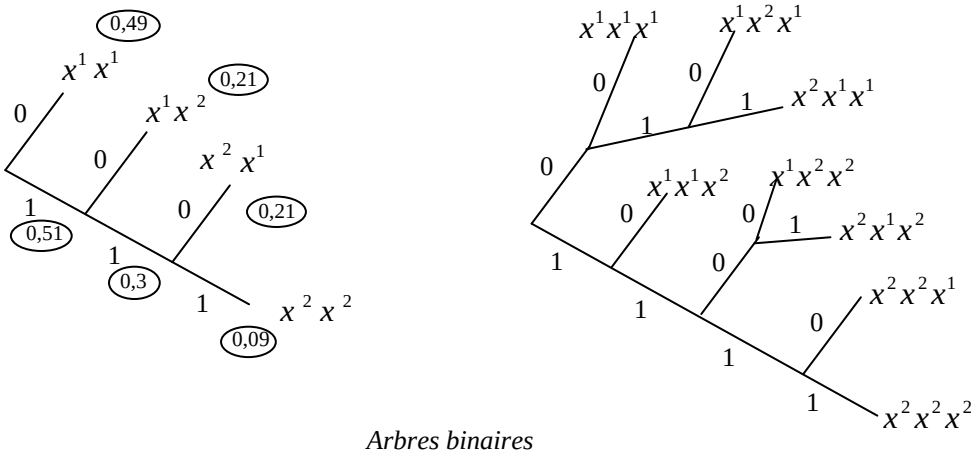
Deuxième exemple

Supposons une source sans mémoire ne pouvant prendre que deux valeurs de probabilités connues. Formons un vecteur de dimension 2 puis un vecteur de dimension 3. L'ensemble des probabilités des différents événements est donné dans le tableau suivant :

			x^1	x^2			
			0,7	0,3			
		$x^1 x^1$	$x^1 x^2$	$x^2 x^1$	$x^2 x^2$		
		0,49	0,21	0,21	0,09		
$x^1 x^1 x^1$	$x^1 x^1 x^2$	$x^1 x^2 x^1$	$x^1 x^2 x^2$	$x^2 x^1 x^1$	$x^2 x^1 x^2$	$x^2 x^2 x^1$	$x^2 x^2 x^2$
0,343	0,147	0.147	0,063	0,147	0,063	0,063	0,027

Probabilités associées aux 2,4 et 8 événements possibles

L'entropie de cette source est égale à 0,88 bit. L'application de l'algorithme de Huffman donne les arbres binaires suivant :



Arbres binaires

Les nombres moyens de bits par symbole sont donnés dans ce tableau :

N	1	2	3
l/N	1	0,905	0,908

Nombres moyens de bits par symbole

Ils ne créent pas forcément une suite décroissante lorsque N augmente mais ils vérifient la double inégalité :

$$H(X) \le \frac{l}{N} < H(X) + \frac{1}{N}$$

7.3 Codage sans bruit d'une source discrète avec mémoire :

On a examiné dans le développement précédent le cas le plus élémentaire. Généralement il existe une dépendance statistique entre les échantillons successifs de la source que l'on va chercher à exploiter pour réduire le nombre de bits nécessaire pour représenter exactement la source.

7.3.1 Nouvelles définitions

Considérons deux variables aléatoires discrètes X et Y prenant leurs valeurs dans respectivement $A_X = \{x^1 ... x^{L_X}\}$ et $A_Y = \{y^1 ... y^{L_X}\}$ et notons les probabilités conjointes $p_{XY}(i,j) = P_r\{X = x^i, Y = y^i\}$ et les probabilités conditionnelles $p_{X|Y}(i|j) = P_r\{X = x^i | Y = y^i\}$.

On appelle :

- information conjointe des 2 événements $\{X = x^i\}$ et $\{Y = y^i\}$, la quantité $I(x^i, y^i) = -\log_2 p_{XY}(i,j)$.
- Entropie conjointe de deux variables aléatoires discrètes X et Y, l'espérance de l'information conjointe $H(X,Y) = -\sum_{i=1}^{L_X}\sum_{j=1}^{l_Y} p_{XY}(i,j)\log_2 p_{XY}(i,j)$.
- Information conditionnelle de l'événement $\{Y = y^i\}$ sachant que l'événement $\{X = x^i\}$ est réalisé, la quantité $I(y^j|x^i) = -\log_2 p_{Y|X}(j|i)$.
- Entropie conditionnelle de la variable aléatoire discrète Y sachant que l'événement $\{X = x^i\}$ est réalisé : $H(Y|x^i) = -\sum_{j=1}^{L_Y} p_{Y|X}(j|i)\log_2 p_{Y|X}(j|i)$.
- Entropie conditionnelle de Y sachant X, l'espérance de l'entropie conditionnelle précédente, c'est à dire
$$H(Y|X) = \sum_{i=1}^{L_X} p_x(i) H(Y|x^i)$$
$$H(Y|X) = -\sum_{i=1}^{L_X}\sum_{j=1}^{L_Y} p_{XY}(i,j)\log_2 p_{Y|X}(j|i).$$

La relation $p_{XY}(i,j) = p_X(i)p_{Y|X}(j|i)$ entraîne la relation suivante entre l'entropie $H(X)$, l'entropie conjointe $H(X,Y)$ et l'entropie conditionnelle $H(Y|X)$:

$$H(X,Y) = H(X) + H(Y|X).$$

En effet
$$H(X,Y) = -\sum_{i=1}^{L_X}\sum_{j=1}^{L_Y} p_{XY}(i,j)\log_2 p_{XY}(i,j)$$

$$H(X,Y) = -\sum_{i=1}^{L_X}\sum_{j=1}^{L_Y} p_{XY}(i,j)\log_2 p_X(i)p_{Y|X}(j|i)$$

$$H(X,Y) = -\sum_{i=1}^{L_X} p_X(i)\log_2 p_X(i) - \sum_{i=1}^{L_X}\sum_{j=1}^{L_Y} p_{XY}(i,j)\log_2 p_{Y|X}(j|i)$$

7.3.2 Théorème du codage sans bruit d'une source discrète avec mémoire

Les définitions précédentes et la relation $H(X,Y) = H(X) + H(Y|X)$ se généralisent au cas de N variables aléatoires discrètes. Considérons le vecteur $X_N = [X(mN)...X(mN+N-1)]^t$ et appelons $p_{X_N}(i_0,...,i_{N-1})$ la probabilité conjointe

$$p_{X_N}(i_0,...,i_{N-1}) = p_r\left\{X(mN) = x^{i_0},..., X(mN+N-1) = x^{i_{N-1}}\right\}.$$

On définit l'entropie de ce vecteur par $H(X_N) = -\sum_{i_0,...,i_{N-1}} p_{X_N}(i_0,...,i_{N-1}) \log_2 p_{X_N}(i_0,...,i_{N-1})$.

On appelle débit entropique $\overline{H}(X) = \lim_{N\to\infty} \frac{1}{N} H(X_N)$.

On montre qu'il est possible d'associer à une source un code uniquement décodable pourvu que le nombre de bits moyen par symbole soit supérieur ou égal au débit entropique.

Si la source est sans mémoire, on a $H(X_N) = -\sum_{i_0,...,i_{N-1}} p_X(i_0)...p_X(i_{N-1}) \log_2 p_X(i_0)...p_X(i_{N-1})$

$$H(X_N) = -N\sum_i p_X(i) \log_2 p_X(i)$$

$$H(X_N) = NH(X)$$

Le débit entropique de cette source est donc égal à l'entropie de la source. De façon générale, on a l'inégalité

$$0 \le \overline{H}(X) \le H(X) \le \log_2 L_X$$

Tout ce développement démontre qu'il existe un code capable de représenter exactement la source à un débit égal au débit entropique mais n'explique pas toujours comment construire ce code.

On remarquera également qu'un codage efficace ne peut être réalisé qu'au dépend du délai de reconstruction.

8. Décompresser

Une fois compris la méthode de compression d'une image au format JPEG, il n'est pas difficile de d'aborder la décompression. En fait, il s'agit de faire le chemin inverse de la compression. Pour cette raison, nous allons nous restreindre à ses étapes principales.

Un logiciel qui va décompresser une image au format JPEG va suivre les étapes principales suivantes :

- ouverture du fichier concerné
- Remise en forme de la matrice quantifiée, en suivant le chemin inverse de la méthode de Huffman.
- Produit terme à terme des coefficients de la matrice DCT quantifiée, par les coefficients de la matrice de quantification.
- Régénération de la matrice de pixels en appliquant la DCT inverse.

La DCT inverse se fait rapidement en utilisant les notations matricielles : si on note F la matrice représentant les valeur DCT et A l'image reconstituée, on a

$$\mathrm{A} = \mathrm{P}\mathrm{F}^{\mathrm{t}}\mathrm{P}$$

La matrice de pixels de sortie n'est plus exactement la même que la matrice d'entrée, mais la perte de données doit rester peu perceptible.

La décompression est un processus plus rapide que la compression. En effet, il n'y a plus de divisions à arrondir et de termes à négliger.

Afin de pouvoir réellement se rendre compte à quoi correspond les différentes matrices citées, nous avons réalisé un applet Java qui permet de les découvrir.

9. Applet JAVA

Conclusion

Comme de nombreuses méthodes de compression, le Jpeg est basé sur des principes mathématiques très compréhensibles. Mais la difficulté intervient lorsque l'on entre en détail dans les démonstrations de l'algorithme. Théorèmes et principes fondamentaux doivent alors être démontrés, pour expliquer les fondements de la méthode.

L'utilisation de telles méthodes de compression des informations est très répandue et utilisées dans de nombreux domaines : informatique, téléphonie, hi-fi, vidéo,…

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

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