

Algorithmes de Compression

1) Définitions.....	3
A) Lossy ou Lossless ?.....	4
B) Redondance et information utile	4
C) Taux de compression.....	4
2) Codages sans pertes.....	5
A) Codes à longueurs variables	5
B) Algorithme de Huffman.....	6
C) Algorithme de Shannon-Fano (ou de Fano) :.....	9
D) Quand peut-on parler de compression ?	11
E) Codage par longueur de plage	12
F) Méthode à base de dictionnaire	14
3) Compression avec perte	14
A) Notions de qualité.....	15
B) Les outils de la compression	16
a) Quantification.....	16
b) Sous-échantillonnage	17
c) Prédications.....	17
d) A base de transformation	18
e) Fractales	19

Les données multimédia (son, images, vidéo) ont en commun d'avoir pour origine des signaux physiques de nature analogique : onde acoustique, onde électromagnétique.

Leur codage numérique est le résultat de compromis entre des contraintes perceptuelles et celles, la plupart du temps opposées, dues à la technique : débit des réseaux, masse de données, normalisations, compatibilités...

Bien que la nature des signaux soit très différente, on peut retrouver des schémas généraux conduisant à la définition des formats multimédias existants : les notions fondamentales de quantification, d'échantillonnage, de compactage de l'information ou encore de compression avec perte.

La compression utilise des outils variés visant à réduire la quantité d'information initiale.

1) Définitions

L'explosion du nombre de signaux numériques (son, images, vidéo...) nécessite d'augmenter les puissances de traitement, d'étendre les capacités de stockage, amplifier les débits de transmission.

Pour accepter une telle croissance, il y a deux solutions :

- Attendre que la technologie progresse davantage, conduisant au risque de perpétuelle insatisfaction.
- S'adapter aux performances de la technologie du moment en réduisant les volumes binaires des fichiers à traiter : c'est l'objectif des méthodes de compression.

Les nombres associés à un signal numérique donné, peuvent être différents suivant la représentation choisie, indépendamment du système d'écriture qui est toujours binaire.

Compresser, c'est choisir la représentation qui permet de conserver minimum de bits. La compression qui concerne l'ensemble des données multimédia est nécessaire pour des raisons de stockage, mais aussi de débit.

Une image fixe en « 16 millions de couleurs » de 640*480 pixels représente $640 \times 480 \times 3 \times 8$ bits soit plus de 7 Mbits. Si on veut visualiser en temps réel une séquence vidéo composée d'image de ce type, l'écran doit recevoir une quantité de bits par seconde, débit, correspondant à 25 images par seconde c'est-à-dire près de 200 Mbit/s...

La question est de savoir si la compression est possible sans nuire au contenu du signal. L'un des premiers à s'être posé cette question fut Shannon, ses études sont à l'origine de la théorie de l'information.

A) Lossy ou Lossless ?

Très grossièrement, nous pouvons scinder les techniques de compression en deux approches, avec ou sans perte, « Lossy » ou « Lossless ».

Les techniques de compression sans distorsion se rapportent au théorème du codage de source sans pertes ou Lossless.

Ces techniques ont l'avantage d'être réversibles mais le gain reste souvent assez faible. On utilisera alors des techniques plus performantes mais impliquant une destruction d'une partie des informations originale, on parle alors de compression Lossy, avec perte. Le but sera évidemment de supprimer les éléments les moins importants...

B) Redondance et information utile

On appelle quantité d'information de source d'un signal numérique le nombre de bits nécessaire à la représentation binaire des nombres qui lui ont été attribués. On peut donc écrire :

$$\text{signal} = \text{information utile} + \text{redondance}.$$

Considérons une suite de caractère lue à partir d'un fichier ou d'un canal de transmission quelconque : **Monsieu**

Le prochain caractère, r, est connu à l'avance de façon certaine (en français), et possède une probabilité conditionnée aux sept précédents, égale à 1.

Ainsi ce huitième caractère (ou symbole), lorsqu'il apparaît (on parle d'événement), ne nous apprend rien dans la mesure où l'on s'y attendait sûrement ! Il n'apporte aucune information, on parle de redondance.

C) Taux de compression

Le but final de la compression est une réduction du poids et/ou du débit caractéristique d'un objet. Pour apprécier l'efficacité d'une technique de compression, on calcule le rapport entre ces poids ou débits avant et après la compression.

Le taux de compression s'exprime en calculant le rapport entre le volume initial des données et le volume après réduction ; si ce volume est deux fois plus faible alors on écrira qu'il s'agit d'un taux de 2:1 ;

$$\text{Taux de compression} = \frac{\text{Poids avant compression}}{\text{Poids après compression}} = \frac{\text{Débit avant compression}}{\text{Débit après compression}}$$

En considérant la relation **Poids = Débit x Temps**, le temps d'annule de lui même.

On peut exprimer le résultat en pourcentage du volume après réduction par rapport au volume initial ; si le volume final représente la moitié du volume initial on écrira qu'il s'agit d'un taux de 50 %.

Un taux de compression n'a donc pas d'unité puisqu'il s'agit du rapport de deux grandeurs exprimées dans la même unité.

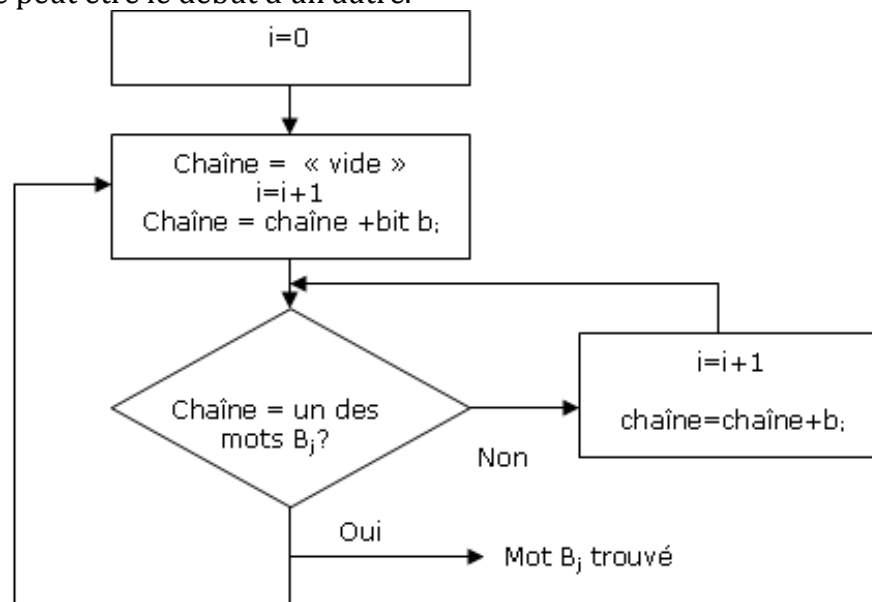
2) Codages sans pertes

A) Codes à longueurs variables

Les codes à longueurs variables sont une mise en œuvre directe et simple du théorème du codage de source sans bruit.

L'idée est simple : les caractères les plus courants doivent être codés par des mots binaires les plus courts possibles, et réserver les codages les plus longs à ceux les moins fréquents. On peut alors espérer un nombre de bits au final plus faible qu'avec un nombre de bits constant pour tous les caractères.

Mais cela pose le problème de la synchronisation au moment du décodage, c'est à dire le rétablissement des frontières entre les mots binaires successifs dans le flux binaire, problème qui ne se pose pas quand tous les mots binaires ont une longueur fixe (par exemple l'octet). Cette synchronisation est possible si le code (c'est-à-dire l'ensemble des mots binaires) est dit « préfixé ». Un code est dit préfixé lorsque aucun des mots qui le composent ne peut être le début d'un autre.



Algorithme de décodage d'un code préfixé

Différents algorithmes permettent de déterminer des codes à longueurs variables pour une source donnée en garantissant ce « préfixage ». Le codage de Huffman et celui de Shannon-Fano sont les deux exemples les plus connus.

B) Algorithme de Huffman

L'algorithme de Huffman est certainement le plus utilisé en raison de sa gratuité, sa simplicité et son efficacité. Un grand nombre de normes de compression d'images le préconisent. C'est le cas de JPEG et MPEG en particulier.

L'algorithme de Huffman a été décrit pour la première fois en 1952. L'algorithme crée des codes de longueurs variables en fonction des probabilités fournies par le modèle, ou la connaissance de la séquence complète. Les symboles sont d'abord triés et classés en fonction de leur fréquence (occurrence). Un graphe est alors construit de la manière suivante :

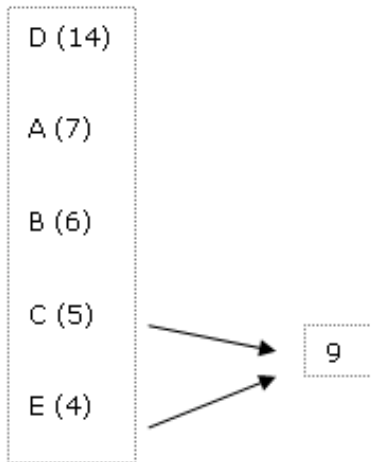
1. A partir des deux symboles présentant la fréquence la plus faible, un nœud est créé.
2. Il lui est affecté un poids égal à la somme des fréquences des deux symboles.
3. Le nœud créé remplace désormais les deux symboles dans la suite du processus.
4. A ces derniers sont affectés respectivement les chiffres binaires 0 pour le plus fréquent et 1 pour le plus rare, (ou l'inverse, il s'agit d'une convention totalement arbitraire).
5. La même démarche est reprise en considérant les deux symboles ou nœuds de poids le plus faible.
6. Elle est renouvelée tant qu'il reste plus d'un nœud libre.

Exemple : Soit un message de 36 caractères, composé des caractères A, B, C, D et E qui apparaissent selon les fréquences suivantes :

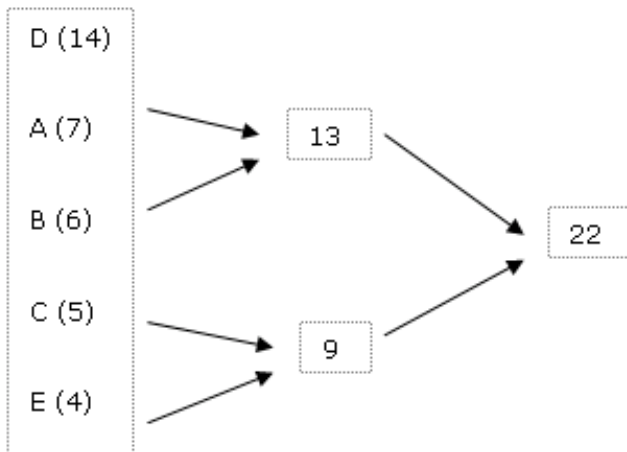
Symboles	A	B	C	D	E
Réurrences	7	6	5	14	4

Principes de l'audiovisuel - Vincent MAZO - EFREI 2012/2013

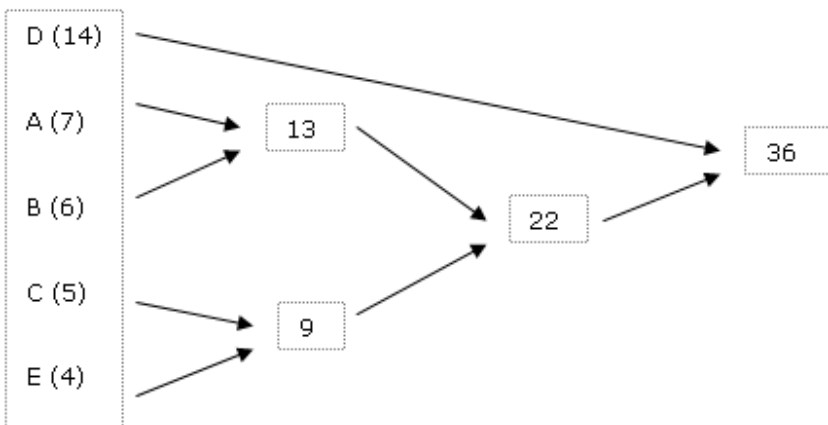
Le graphe ci-dessous fait apparaître les symboles depuis le plus fréquent (D) jusqu'au moins fréquent (E). Les symboles C et E fusionnent vers un même nœud de poids $5+4=9$



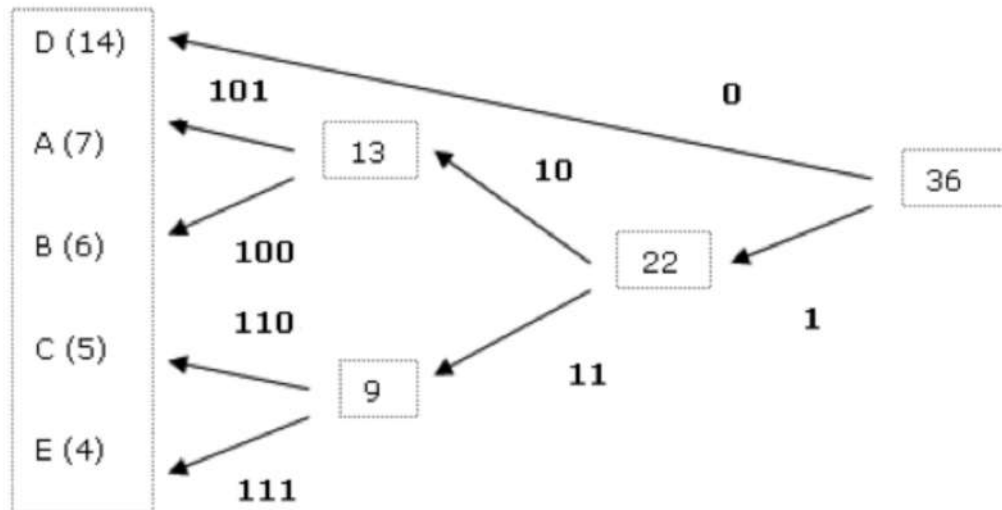
Puis les symboles ou nœuds de poids les plus faibles sont à nouveau considérés : il s'agit ici de 2 symboles à nouveau : A et B Le processus est renouvelé avec le nœud de poids 9 et celui de poids 13.



Il reste enfin à fusionner le nœud de poids 22 et le symbole restant, de poids 14 :



La deuxième phase de l'algorithme consiste à « redescendre » ce graphe binaire, en affectant à la branche conduisant au poids le plus faible, la valeur 0 par exemple, et 1 à l'autre. On obtient :



Le codage obtenu est donc :

Symboles	A	B	C	D	E
Réurrences	7	6	5	14	4
Codage	101	100	110	0	111

Le symbole le plus fréquent a obtenu le code le plus court, les autres ont, dans cet exemple, des longueurs équivalentes.

Quelle est la longueur moyenne de code ?

1 bit pour D qui apparaît 14 fois dans le message, 3 bits pour les autres qui apparaissent 22 fois en tout.

$$\text{Longueur moyenne} = (1 \cdot 14 + 3 \cdot 7 + 3 \cdot 6 + 3 \cdot 5 + 3 \cdot 4) / 36 = 80 / 36 = 2.22$$

Pouvait-on faire mieux ?

La réponse est donnée par la valeur de l'entropie du message :

$$H(S) = \sum_{P(x_i) > 0} P(x_i) \log_2(1/P(x_i))$$

$$\begin{aligned} H &= (14/36) \cdot \log_2(36/14) + (7/36) \cdot \log_2(36/7) + (6/36) \cdot \log_2(36/6) + (5/36) \cdot \log_2(36/5) + (4/36) \cdot \log_2(36/4) \\ H &= \log_2(36) - (1/36) \cdot [14 \cdot \log_2(14) + 7 \cdot \log_2(7) + 6 \cdot \log_2(6) + 5 \cdot \log_2(5) + 4 \cdot \log_2(4)] \\ H &= \ln(36)/\ln(2) - 1/(36 \cdot \ln(2)) \cdot [14 \cdot \ln(14) + 7 \cdot \ln(7) + 6 \cdot \ln(6) + 5 \cdot \ln(5) + 4 \cdot \ln(4)] \\ H &= 5.170 - 3.002 = 2.168 \end{aligned}$$

Nous voyons que la longueur moyenne de code donné par l'algorithme d'Huffman est très proche du plancher théorique de l'entropie.

C) Algorithme de Shannon-Fano (ou de Fano) :

L'algorithme est le suivant :

- 1) On classe les symboles par ordre de probabilités décroissantes,
- 2) On partage l'ensemble des symboles en deux sous-ensembles, les 2 sous-ensembles devant être de probabilités aussi proches que possible,
- 3) On attribue à chaque sous-ensemble l'état 0 ou 1,
- 4) On re-partage chaque sous-ensemble en deux sous-ensembles de probabilités aussi proches que possible, on attribue à chaque nouveau sous-ensemble l'état 0 ou 1, etc.

L'algorithme est arrêté lorsque l'on atteint des sous-ensembles ne comportant qu'un symbole.

Prenons le même exemple :

Symboles	A	B	C	D	E
Réurrences	7	6	5	14	4

Une fois ordonnée par fréquences décroissantes, la liste de symboles est partagée en 2 ensembles de probabilités les plus proches (sommes des probabilités des symboles les composant) :

D (14)

A (7) somme des occurrences : $14 + 7 = 21$

B (6)

C (5) somme des occurrences : $6 + 5 + 4 = 15$

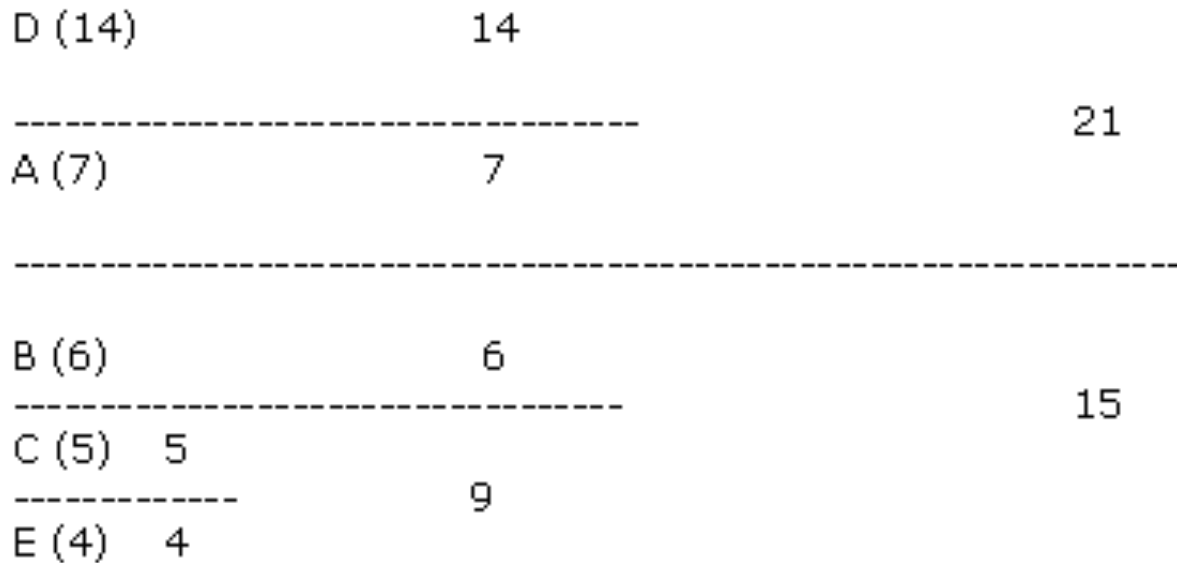
E (4)

Les autres découpages possibles présentent des disparités supérieures.

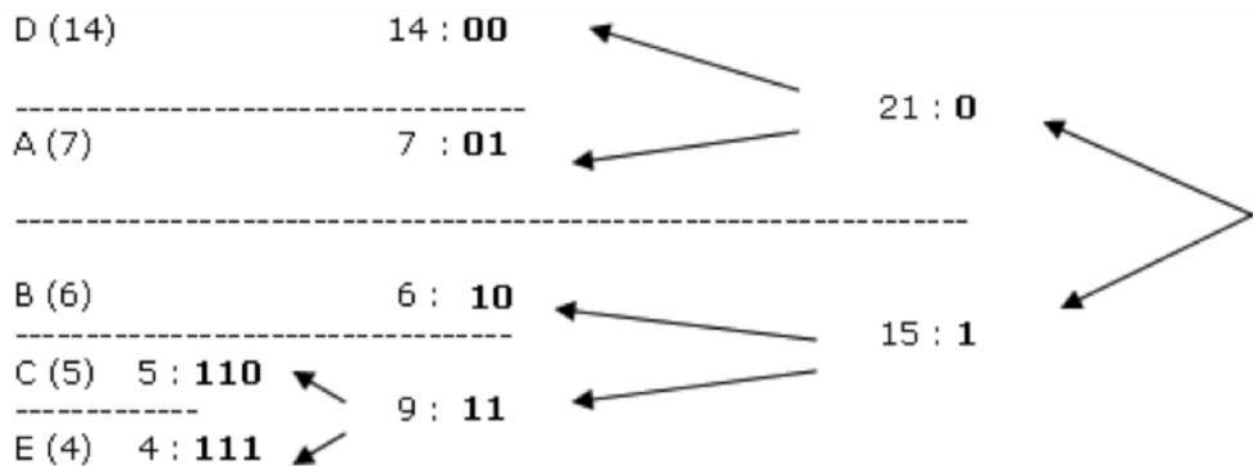
Chaque sous-ensemble est décomposé selon le même principe.

Principes de l'audiovisuel - Vincent MAZO - EFREI 2012/2013

On aboutit à :



Chaque sous-ensemble ne contient qu'un symbole. Le processus est alors arrêté et le codage est effectué en partant du sommet de la hiérarchie obtenue, en choisissant 0 pour la probabilité la plus forte par exemple :



Le codage obtenu est donc :

Symboles	A	B	C	D	E
Réurrences	7	6	5	14	4
Codage	01	10	110	00	111

Les symboles les plus fréquents ont là encore obtenu les codes les plus courts.

Quelle est la longueur moyenne de code ?

2 bits pour D, A et B qui apparaissent au total 27 fois dans le message, 3 bits pour les 2 autres qui apparaissent 9 fois en tout.

$$\text{Longueur moyenne} = (2 \cdot 27 + 3 \cdot 9) / 36 = 81 / 36 = 2.25$$

Le résultat est très proche de celui obtenu par Huffman.

Les 2 sont proches de l'entropie $H = 2.168$

Les 2 codes sont bien préfixés : aucun code n'est le début d'un autre code.

Le décodage pourra se faire sans ambiguïté. Il faut toutefois fournir la table de correspondance entre les codes obtenus et les symboles de départ : on parle de dictionnaire.

D) Quand peut-on parler de compression ?

Le taux de compression peut se définir comme le rapport entre la taille du message dans son code d'origine, et la taille du message après re-codage.

Pour effectuer la comparaison, on peut considérer que les symboles sont des caractères, codés généralement sur 1 octet (8 bits soit 256 combinaisons possibles). Mais sur cet exemple ne faisant appel qu'à 5 caractères différents, il est plus juste de considérer qu'un codage à taille fixe nécessiterait seulement 3 bits pour chaque caractère soit $3 \cdot 36 = 108$ bits pour le message complet.

Le taux de compression obtenu est donc de $108/80 = 1.35$. Si on considère qu'il faudrait ajouter le dictionnaire pour retrouver le format d'origine, la compression risque ici de ne pas être valable !

Elle devient intéressante pour des messages longs (la taille du dictionnaire devient négligeable) et des grandes disparités dans les occurrences.

Intuitivement, on se rend compte que si les symboles étaient tous équiprobables, les techniques précédentes conduiraient à obtenir des codes dont les longueurs moyennes correspondent à celles d'un code à longueur fixe. Dans le cas où le nombre de symboles à coder est une puissance de 2, le code à longueur fixe est alors optimal et il n'y a aucune redondance à exploiter.

Dans le cas général, le principe de la synthèse d'un code compact consiste donc à ramener le codage d'une série de symboles au codage de deux symboles ou de deux groupes de symboles ayant des probabilités d'émission les plus égales possible.

C'est sur ce principe que reposent les algorithmes de Huffman ou de Fano. En règle générale, ces 2 codages sont concurrents et le plus efficace des 2 dépend de la source.

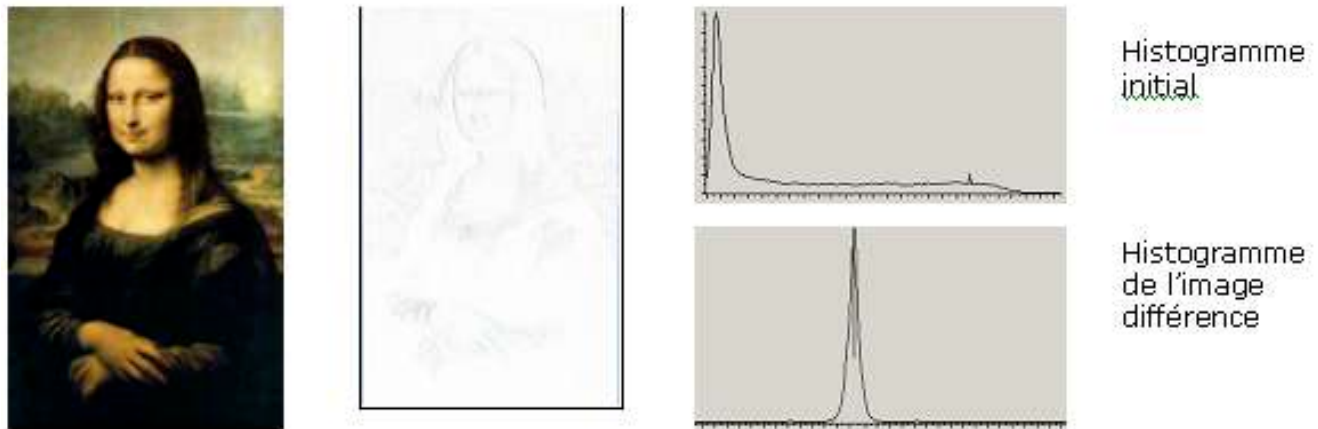
Ces algorithmes sont-ils adaptés à la compression des images ?

Calculons la différence entre le niveau de gris du pixel courant $s(x,y)$ et celui du précédent sur la même ligne $s(x-1,y)$

$$d(x,y) = s(x,y) - s(x-1,y)$$

On s'aperçoit que très souvent cette différence est très faible, les pixels contigus ayant des valeurs voisines.

On peut s'en convaincre par les histogrammes de l'image suivante et de l'image différence.



Corrélation spatiale des images

Dans la pratique, les algorithmes de codage entropique ne sont pas utilisés directement sur les valeurs des pixels de l'image, mais sur des grandeurs obtenues après transformation des valeurs des pixels (transformation DCT).

Une autre technique consiste à utiliser les corrélations de l'image en cherchant à appliquer le principe d'extension de source : comme il est difficile de prévoir la longueur de blocs de pixels voisins uniformes, c'est sous une forme légèrement différente que le principe d'extension de source va être adopté : il s'agit du codage par longueur de plages.

E) Codage par longueur de plage

La redondance est basiquement le fait de répéter une information.

Le phénomène de redondance peut également être observé dans un message.

On se souvient d'Yves Montand dictant dans un télégramme pour Mademoiselle Colette Mercier de Besançon (Doubs) :

« Je pense à toi, je t'aime, je t'aime, je t'aime... »

et l'opératrice, Simone Signoret, lui répondant « alors trois fois je t'aime ».

La redondance dans les messages humains est souvent nécessaire et signifiante... « trois fois je t'aime » ce n'est évidemment pas la même chose que « je t'aime, je t'aime, je t'aime ».

La solution adoptée dans la méthode appelée RLE (Run Length Encoding) ou RLC (Run Length Coding) consiste à créer des couples de valeurs pour décrire des plages consécutives en indiquant d'une part la valeur caractéristique et d'autre part le nombre de fois où cette valeur est répétée.

On forme, en lisant l'image ligne par ligne, des couples {longueur de plage, intensité}, où la longueur de plage est le nombre de pixels consécutifs ayant la valeur intensité.

C'est une méthode très simple si l'image comporte de nombreuses « plages » de valeurs identiques : il est alors avantageux de coder la longueur (le nombre de symboles identiques) et la valeur de chaque plage. C'est particulièrement vrai dans le cas d'images avec « aplats » (Ex. logos, images graphiques), ou le codage des coefficients DCT quantifiés dans l'algorithme de type JPEG.

Soit la suite suivante, correspondant à des niveaux de gris prélevés dans une ligne d'image :

11, 11, 15, 16, 16, 16, 16, 25, 25, 25, 31, 31, 31, 31, 31, 8

Le codage par plage d'un tel signal donne la suite des paires de nombres suivantes :

(2, 11), (1, 15), (4, 16), (3, 25), (5, 31), (1, 8)

fournissant un nouveau codage de la ligne :

2, 11, 1, 15, 4, 16, 3, 25, 5, 31, 1, 8,

Soit la suite des coefficients suivante :

15, 0, -2, -1, -1, 0, 0, 0, 1, 0, 0, 0, ..., 0 (64 coefficients au total)

Un codage légèrement modifié par rapport au précédent, qui serait

{nombre de zéros qui précèdent une valeur donnée, valeur} donne :

0, 15, 1, -2, 0, -1, 0, -1, 3, 1, EOB (End of Block = la suite ne comprend plus que des zéros)

Dans tous les cas, retenir que l'information codée n'est plus la source originale, mais on utilise les propriétés spécifiques aux images pour dé-corréler la source, ce qui permet d'extraire et concentrer la source.

Les méthodes de type RLE correspondent aux schémas les plus courants de compression des images (dans les formats BMP, TGA, TIFF, SGI, PICT, ...). Efficace dans les images « simples », il devient mauvais sur les images complexes (scènes naturelles).

F) Méthode à base de dictionnaire

Dans les codages entropiques précédents, les sources étudiées étaient considérées comme aléatoires. Dans la pratique, les sources ne sont pas purement aléatoires, et on peut observer des corrélations qui s'étendent au delà de plages consécutives : un motif peut réapparaître plus loin par exemple.

Dans les méthodes précédentes, un tel motif serait à nouveau codé, ce qui conduit à coder de l'information inutile. Les codages par dictionnaire sont fondés sur la constitution d'une suite de motifs, apparaissant dans le flux du message.

Ces motifs remplissent un dictionnaire, où ils sont indexés, et lorsque les motifs réapparaissent, ils sont simplement codés par leur index dans ce dictionnaire. Ils seront efficaces pour les sources longues, codés sur des alphabets restreints, ce qui permet d'envisager la présence de motifs récurrents.

Les codages de ce type les plus connus appliquent les algorithmes de type LZ ou l'une de ses nombreuses variantes (LZ77, LZ78, LZW, LZSS...). Ces noms viennent de celui de leurs auteurs : Lempel, Ziv, Welch, ou de ceux leur ayant apporté des améliorations (Storer et Szymanski).

LZ77 est utilisé dans les outils de compression de type ZIP : gzip sous linux, pkzip sous DOS, winzip sous windows, stuffit sous Macintosh... LZW est souvent plus efficace et fait l'objet d'une norme pour les images (formats GIF et TIFF).

LZW n'est plus gratuit depuis janvier 1995, et cela après une longue période de liberté qui avait favorisé son utilisation. Cet algorithme est couvert par un brevet américain détenu par Unisys. Une licence de Unisys est obligatoire pour toute utilisation. Les images au format PNG (Portable Network Graphics) utilise une variante de l'algorithme LZ77 ; ce format a été spécifié par le consortium W3C (World Wide Web), qui lui octroie un certain nombre d'avantages par rapport à GIF et JPEG. On notera que le format PNG (et l'algorithme LZ77) est dans le domaine public.

3) Compression avec perte

Les systèmes de compression Lossless décrit jusqu'ici ont l'avantage de ne pas détruire d'information. On peut retrouver les données identiques après décodage. Cependant les taux de compression sont relativement faibles en général.

On utilisera alors des systèmes de compression Lossy pour augmenter le taux de compression. Mais les données seront irrémédiablement altérées...

A) Notions de qualité

Il existe un grand nombre d'applications pour lesquelles il n'est pas nécessaire de garantir un décodage parfait. Ce sont, pour l'essentiel, les applications qui traitent les signaux destinés à nos sens : le son et l'image en particulier, si une part de l'information initiale ne nous est pas perceptible.

Nos sens sont mal commodes pour mesurer objectivement la distorsion. Et pourtant, le seul instrument vraiment capable d'apprécier la qualité d'une image est bien l'oeil lui-même.

Il existe des mesures dites « objectives », parfois appelés critères de qualité. Leur inconvénient majeur est d'être discutables dans la mesure où aucun de ces critères ne correspond exactement à ce que ressent l'œil ou plutôt l'ensemble du système visuel . Parmi les critères objectifs, on peut retenir l'Erreur Quadratique Moyenne (EQM) :

$$EQM = \frac{1}{N} \sum_{i=1}^N (Sci - Si)^2$$

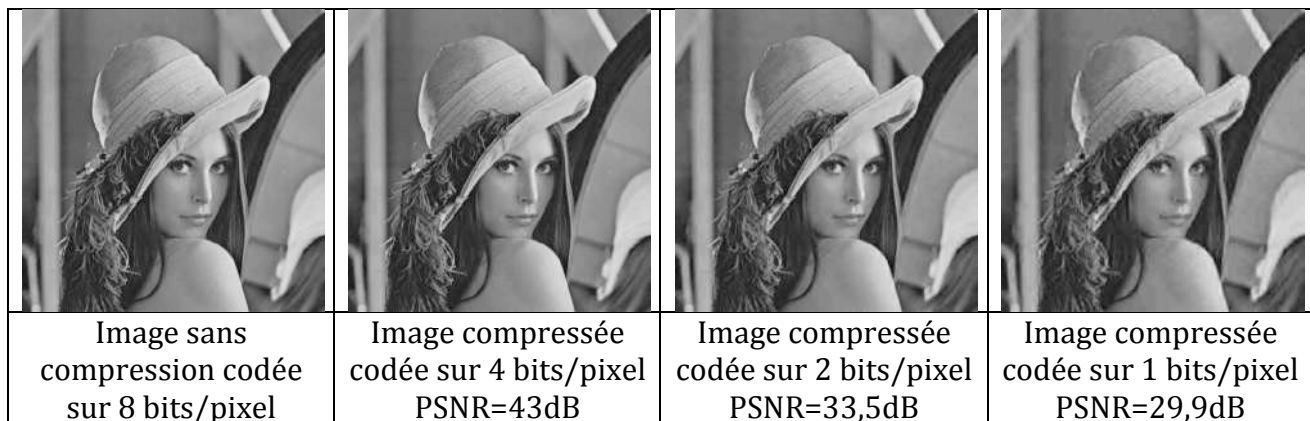
avec sci, valeur décodée de l'échantillon si

On utilise fréquemment une grandeur dérivée de l'EQM, plus physiologique : le rapport signal sur bruit crête (PSNR, Peak Signal Noise Ratio), défini (en dB) par :

$$PSNR = 10 \log \frac{(\max(Si))^2}{EQM}$$

Par exemple, si **si** représente la valeur des niveaux de gris des pixels d'une image, codés sur 8 bits, alors $\max(si)=255$.

On considère que si le PSNR est inférieur à 30 dB, la compression conduit à une image décodée dont les différences avec l'originale sont visibles à l'oeil nu. (Encore faut-il disposer de l'originale et de disposer d'un système de visualisation de qualité suffisante pour apprécier visuellement les différences).



B) Les outils de la compression

Parmi les méthodes de compression avec perte appliquées aux images, on peut distinguer:

- les méthodes à base de quantification
- les méthodes à base de sous-échantillonnage
- les méthodes à base de prédiction
- les méthodes à base de transformation
- les méthodes à base de fractales
- ...

a) Quantification

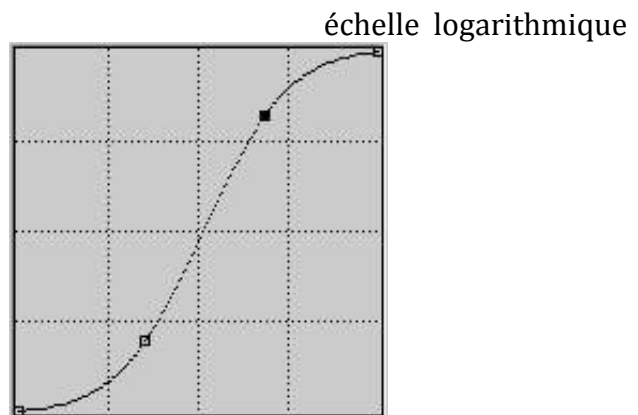
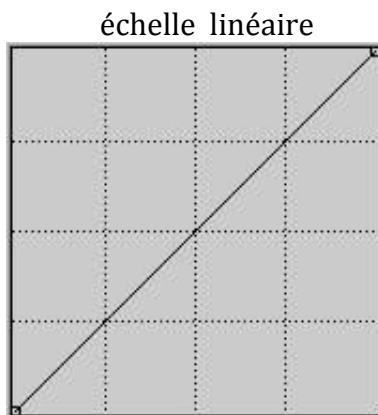
Les méthodes à base de quantification permettent une compression en réduisant la longueur des mots binaires et le nombre de bits du signal. On utilisera par exemple une quantification sur 8 bits au lieu de 10 bits pour réduire le poids de l'objet dans un rapport de 25%.

L'imprécision de la quantification (bruit de quantification) sera alors d'autant plus élevée.

Pour une même valeur de l'EQM, il est possible de réduire encore le nombre de niveaux de quantification, en adoptant un quantificateur non uniforme.

Le cas le plus concret consiste à utiliser une échelle logarithmique au lieu d'une échelle linéaire. En effet l'œil a une réponse aux écarts de lumière qui est quasi-logarithmique. C'est-à-dire qu'il est sensible à des écarts constants d'environ 1%.

Donc plus on se situe dans les basses lumières, plus l'oeil devient sensible à des différences de lumière très faibles. Cette notion est facile à comprendre, prenons l'exemple d'une pièce éclairée seulement par une bougie, si on en allume une deuxième on perçoit nettement l'augmentation de l'intensité lumineuse. Par contre sur une plage en plein soleil, il est évident que l'on allumera une ou deux bougies sans percevoir la moindre variation de luminance.



En quantification vectorielle, on travaille sur des petits blocs de l'image qui sont remplacés par des valeurs issues d'un dictionnaire, et non sur des pixels isolés. On pourra ainsi identifier un bloc de valeurs qui se répète à différents endroits.

b) Sous-échantillonnage

On peut délibérément choisir de ne pas conserver la totalité des éléments échantillonnés. Cela se traduit par une perte de détail des éléments considérés. Nous avons utilisé ce système pour la vidéo, dans le cas du sous-échantillonnage 4:2:2 par exemple.



Image de base

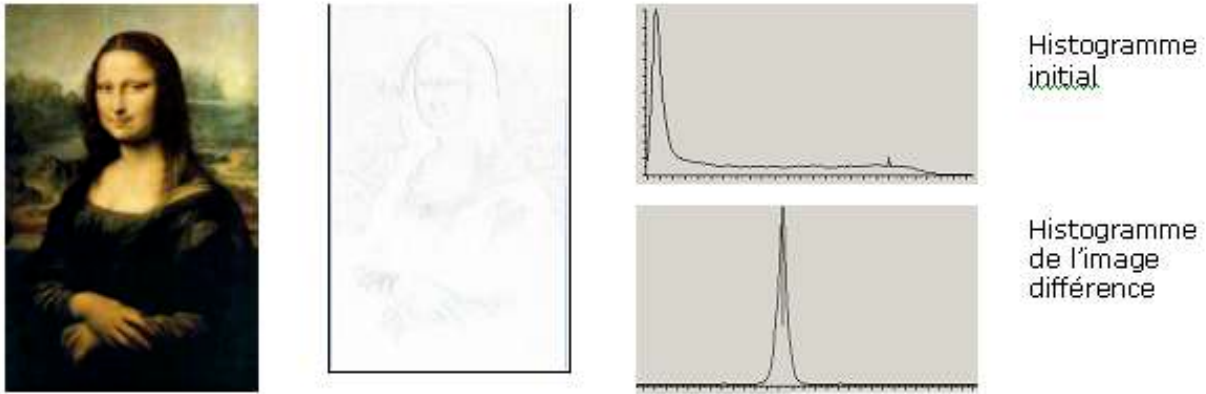


image sous échantillonnée

Il va de soit que les détails de l'image sous-échantillonnée sont dégradés, elle devient flou.

c) Prédictions

Les méthodes de prédiction reposent sur la grande corrélation des pixels voisins en image, ou sur les échantillons successifs d'un signal. Pour prédire de nouveaux échantillons, on utilise des coefficients pondérant les échantillons qui servent à « prédire » le pixel codé. On parle dans ce cas de prédiction spatiale. C'est une application concrète de la corrélation spatiale des éléments d'une image.



Les méthodes de prédiction interviennent également dans le cas des séquences d'images (vidéo) : la prédiction est alors temporelle.

Nous étudierons ce type de compression avec la famille MPEG.

d) A base de transformation

Ces méthodes sont utilisées sur les signaux acoustiques comme sur les images. Ces transformations s'opèrent sur des blocs. Ces transformations linéaires s'écrivent mathématiquement à l'aide d'une matrice de passage, qui fournit à partir des échantillons du signal, des composantes transformées, appelés encore coefficients transformés.

Le matriçage RVB $\leftrightarrow$ YUV est un exemple de transformation.

Après transformation, les blocs perdent leur nature initiale et leur sens psychophysique. La transformation est donc provisoire, et doit être réversible.

On montre qu'il est possible de trouver des transformations linéaires qui, quelles que soient les valeurs initiales, celles des coefficients transformés deviennent très différentes les unes des autres.

Si ces valeurs sont très grandes pour certaines, l'annulation des plus petites n'entraîne qu'une distorsion réduite : la compression s'effectue par l'abandon d'un certain nombre de ces composantes, les plus faibles.

Les composantes du bloc transformé n'ont pas nécessairement une importance psychophysique proportionnelle à leur valeur. Mais il existe des transformées qui garantissent :

- une grande disparité entre les valeurs de ses composantes et donc une diminution de l'entropie.
- une importance psychophysique réduite pour les plus petites des composantes transformées.

La DCT (Discret Cosin Transform) ou Transformée en cosinus discrète, présente ces avantages, plus quelques autres que n'ont pas d'autres transformées (coefficients réels, parité ...) comme la transformée de Fourier.

La distorsion introduite par cette méthode est issue de l'annulation des composantes les plus petites et de la quantification des autres : les composantes obtenues sont des nombres réels, même si les grandeurs initiales sont des entiers.

L'avantage des méthodes par transformation tient dans les taux de compression obtenus. En revanche, la méthode nécessite de travailler sur des vecteurs, c'est à dire des blocs dans une image. La complexité de l'algorithme implique une taille réduite de ces blocs : dans les utilisations normées (JPEG, MPEG), les blocs comprennent 8x8 pixels.

A fort taux de compression, des discontinuités apparaissent dans les images aux frontières des blocs.

Ces méthodes se prêtent bien à des affichages progressifs, ou à des résolutions variables. Dans l'algorithme JPEG, la transformation est donc associée à une quantification et un codage entropique selon le schéma suivant :

JPEG =	transformation (DCT)	+	quantification	+	codage
	Diminution Entropie		perte information		exploitation
	(concentration Info)				redondance

e) Fractales

Les concepts d'images fractales, créés et développés par Iterated Systems, reposent sur l'idée qu'une image peut être décrite à partir d'un ensemble de motifs répétitifs et des transformations géométriques affines.



A et A' peuvent être appairés après symétrie, B et B' après une mise à l'échelle.

Contrairement aux autres techniques de compression habituelles, la compression fractale ne tente pas de réduire le nombre de couleurs (format gif) ou de compresser de manière classique les octets composant l'image.

Le format FIF se base sur le principe qu'une image n'est qu'un ensemble de motifs identiques en nombre limité, auxquels on applique des transformations géométriques (rotations, symétries, agrandissements, réductions).

L'image affichée est donc générée par un ensemble de transformations, non plus point par point, mais sous forme de formules mathématiques (et de références à celles-ci) qui, interprétées par l'algorithme adéquat, reconstituent par leur assemblage l'image initialement compressée.

Ce procédé permet donc au codage d'une image de demeurer indépendant de la taille (largeur, hauteur ou encore espace disque occupé initialement) et offre de forts taux de compression : de 1:10 jusqu'à 1:50 en moyenne !

La compression est effectuée soit par voie logicielle, soit par voie matérielle (pour les systèmes les plus optimisés).

Le codec correspondant est fortement dissymétrique. La décompression est extrêmement rapide (de l'ordre de la seconde), ce qui rend ce type d'images très intéressant pour les interfaces Web.

La compression est en revanche plus lente (de 30 s à 5 min suivant l'équipement). Il est donc conseillé d'adjoindre des cartes de compression spécifiques en cas de production d'un grand nombre d'images.

La netteté de l'image diminue progressivement lorsque le taux de compression augmente, mais aucun effet de mosaïque n'apparaît (rappelons qu'on ne stock plus des informations sur les points mais sur les formes géométriques).

Lors d'un zoom, par exemple, l'image fractale peut devenir floue mais ne pixelise pas. Ceci est dû au fait que lors de l'agrandissement, ce ne sont pas les pixels qui sont élargis, mais toute l'image qui est recalculée mathématiquement.

De surcroît plus l'image se conforme à des modèles géométriques, moins le taux de compression influera sur la qualité finale.