



**Palais des Congrès - Porte Maillot - Paris**

**Séminaires Formation  
Les nouveaux savoir-faire**

# Compresser les images

Sylvain Renard

La compression des données a pris aujourd'hui une importance considérable dans des domaines variés : télévision, musique, télédétection, imagerie médicale,... Le graphiste utilise depuis longtemps (parfois sans le savoir) diverses méthodes de compression implémentées dans les logiciels du commerce. On se propose ici, en se limitant au cas particulier des images fixes, de faire l'inventaire des méthodes disponibles, d'en faire comprendre le principe et de tracer les grandes lignes du futur prévisible dans ce domaine.

## Sommaire

[Prologue](#)

[Introduction](#)

[Les méthodes de compression sans perte](#)

- [Un exemple simple](#)
- [Les méthodes statistiques](#)
- [Les algorithmes de type dictionnaire](#)

[Les méthodes de compression avec pertes](#)

- [L'algorithme JPEG](#)
- [La compression fractale](#)
- [La compression par ondelettes.](#)

[Le format JPEG 2000](#)

## PROLOGUE

Claude Shannon, le fondateur de la théorie de l'information avait l'habitude de faire jouer à un petit jeu de société quand il était invité quelque part. Il prenait un livre au hasard, l'ouvrait au hasard, commençait à lire un paragraphe et s'arrêtait. Il demandait ensuite à l'assistance de deviner une à une les lettres suivantes. L'assistance se débrouillait bien et trouvait la lettre dans environ 75 % des cas. Shannon en déduisait que la langue anglaise possède un taux de redondance de 75 %.

Quand nous manipulons du texte, les caractères que nous utilisons n'ont pas la même probabilité d'apparition. De plus il a une structure interne forte (la grammaire). Quand le mot arbre est au pluriel on peut aisément prédire la lettre qui suit le « e » final.

Quand nous travaillons avec de la musique, la distribution des probabilités d'apparition des sons n'est pas uniforme non plus.

Quand nous manipulons des images, elles possèdent également des régularités, elles ne sont pas « aléatoires ». Quand on considère la photo d'une maison jaune avec une porte bleue, la probabilité pour un pixel bleu d'avoir ses « voisins » bleus n'est pas la même que d'avoir ses « voisins » jaunes !

Bref, la majorité des données que nous traitons ont un ordre interne, même s'il n'est pas apparent, c'est à dire une distribution non uniforme de certains symboles ou séquences de symboles.

C'est cette caractéristique qui incite à compresser les données et c'est elle qui permet, souvent, de réussir.

## INTRODUCTION

La compression des données est un vaste sujet qui a fait l'objet de nombreux ouvrages et articles; elle donne lieu aujourd'hui à de nombreuses recherches en raison des enjeux économiques sous-jacents. Elle est utile en PAO mais elle est une des conditions d'existence du multimédia.

Les domaines mathématiques et informatiques dont la connaissance est nécessaire pour comprendre ces problèmes sont nombreux et complexes : calcul intégral, algèbre linéaire, géométrie fractale,

théorie de l'information, théorie des ondelettes, théorie des probabilités,... On comprendra facilement également que la compression des données a quelque chose à voir avec le fonctionnement de notre système visuel et avec la construction des algorithmes.

On se contentera donc dans ce bref exposé, d'ouvrir un champ de réflexion en indiquant souvent sommairement quelles sont les méthodes utilisées aujourd'hui pour compresser les données et quelles sont celles qui ont des chances de s'imposer demain.

Ce petit exposé est composé de trois parties :

- **Les méthodes réversibles (sans perte).**

Je traiterai d'abord un exemple très simple mais qui peut être entièrement expliqué et appliqué ici. J'aborderai ensuite les méthodes de codage statistique (algorithmes de Huffman et de Shannon-Fano) pour terminer par les méthodes dites « à dictionnaire » (algorithme LZW)

- **Les méthodes irréversibles (avec pertes).**

J'exposerai relativement précisément la compression JPEG en raison de son importance dans nos métiers et je donnerai quelques vues sur l'utilisation des fractales et des ondelettes dans la compression de données.

- **Le format JPEG 2000**

J'expliquerai pourquoi il est aujourd'hui nécessaire d'adopter une nouvelle norme et je donnerai quelques explications sur ce que sera le format JPEG 2000, appelé à remplacer le format JPEG actuel.

Avant d'aller plus loin définissons d'abord la compression. Nous dirons que nous avons compressé un fichier si nous parvenons à réduire le nombre de digits binaires nécessaires pour l'enregistrer.

On mesure l'efficacité de la compression par le taux de compression :

$$\sigma = \frac{\text{Nombre de digits binaires utilisés par l'image originale}}{\text{Nombre de digits binaires utilisés pour l'image compressée}}$$

$\sigma$  est toujours plus grand que 1 et nous souhaitons qu'il soit le plus grand possible.

Il est plus difficile de mesurer la qualité de la compression.

Essayons. Supposons que nous compressions un fichier composé des données  $n_1, n_2, n_3 \dots$ . Quand on décompresse le fichier on récupère des données  $\tilde{n}_1, \tilde{n}_2, \tilde{n}_3 \dots$ .

Si  $\tilde{n}_1 = n_1, \tilde{n}_2 = n_2, \tilde{n}_3 = n_3 \dots$  la compression est sans pertes. Si certaines valeurs ont été modifiées, il y a perte et il faut essayer de mesurer l'écart entre l'image originale et celle que l'on récupère après compression.

Pour chaque valeur on va faire la différence  $\tilde{n}_i - n_i$  qui peut être positive ou négative. Pour que les différences ne s'annulent pas, on va prendre le carré de chaque différence et on va en faire la moyenne pour l'ensemble des données, par exemple pour tous les pixels d'une image. Nous venons de définir l'erreur quadratique moyenne (EQM).

(pour les matheux, on a :

$$EQM = \frac{1}{N} \sum_{i=1}^{i=N} (\tilde{n}_i - n_i)^2$$

pour une suite de N données)

L'inconvénient de cette méthode de calcul et d'ailleurs de toutes les méthodes « simples » d'évaluation de la qualité d'une compression, c'est qu'elles ne tiennent pas compte de la manière dont nous percevons une image, en particulier elle ne tiennent pas compte de la répartition spatiale de l'erreur (zones plus ou moins importantes de l'image). En fait, il faudra souvent faire des tests avec des images différentes et des « observateurs » différents pour comparer les méthodes de compression.

## LES METHODES DE COMPRESSION SANS PERTE

### Un exemple simple.

Étudions d'abord un exemple très simple.

Supposons que nous devions essayer de compresser la série d'octets suivante :

```
0001 0100   0011 1111   0101 0101   0101 0101   0101 0101
0101 0101   1110 1111   0000 1111   1111 1111   1111 1111
1111 1111   1111 1111   1111 1111   0110 0111   0111 0000
0111 0000   1010 1111   0000 0000   0001 1111   0001 1111
0001 1111
```

Elle compte 21 octets. (Vous pouvez l'interpréter comme une suite de caractères ou comme les niveaux de gris d'une suite de pixels, cela n'a pas d'importance pour ce qui nous préoccupe ici).  
Nous allons coder cette chaîne en utilisant les répétitions d'octets pour gagner de la place.

Commençons par écrire un octet de signalisation, pour le repérer, nous l'écrivons en gras.

**0000 0010**

Convenons que le premier bit indique si l'octet de donnée qui suit se répète, si c'est le cas le bit sera mis à 1 (ce n'est pas le cas ici), si ce n'est pas le cas il sera mis à 0. Les 7 bits qui suivent indiqueront le nombre de répétitions dans le premier cas ou le nombre d'octets sans répétition qui suivent (ici on a écrit 0000 0010 car on va faire suivre l'octet de signalisation par deux octets différents qui ne se répètent pas).

Notons qu'avec ces 7 bits, nous pouvons coder 127 répétitions. On voit l'économie réalisée si beaucoup d'octets sont identiques (un masque dans Photoshop...).

Notre chaîne codée commence donc par :

**0000 0010** 0001 0100 0011 1111

Ça commence mal ! Nous avons utilisé 3 octets pour en coder 2, mais ça va s'améliorer. Continuons. Les 4 octets qui suivent sont identiques, ce que nous indiquons par l'octet de signalisation :

**1000 0100**

Le premier bit à 1 annonce une répétition, la valeur 0000100 (4 en binaire) indique le nombre de répétitions.

Nous pouvons compléter notre chaîne codée :

**0000 0010** 0001 0100 0011 1111 **1000 0100** 0101 0101

Vous vérifierez qu'en continuant nous obtenons la chaîne suivante qui code l'ensemble du message :

**0000 0010** 0001 0100 0011 1111 **1000 0100** 0101 0101  
**0000 0010** 1110 1111 0000 1111 **1000 0101** 1111 1111  
**0000 0001** 0110 0111 **1000 0010** 0111 0000 **0000 0010**  
1010 1111 0000 0000 **1000 0011** 0001 1111

Cette chaîne fait 19 octets soit 2 de moins que la chaîne initiale, nous avons un taux de compression de  $21/19 = 1,1$

Le décodage ne pose aucun problème à condition que le décodeur soit informé de la méthode utilisée, il interprétera dans ce cas le premier octet comme un octet de signalisation et tout ira bien...

Cette méthode est séduisante parce qu'elle est facile à comprendre mais elle est malheureusement peu efficace. Elle n'est plus utilisée seule mais seulement comme méthode complémentaire.

## Les méthodes statistiques.

L'idée est la suivante : nous avons l'habitude de coder toutes les valeurs que nous avons à stocker avec le même nombre de bits, généralement un ou plusieurs octets. Nous utilisons un octet par caractère quand nous travaillons avec du texte, un ou deux octets par pixel et par couche quand nous travaillons avec des images. Est-ce la meilleure solution ? La théorie nous apprend que non.

Il est en effet plus efficace d'utiliser des codes plus courts pour des valeurs fréquentes et de réserver des codes plus longs pour les valeurs moins fréquentes. L'intérêt de cette méthode dépendra bien entendu de la distribution des valeurs à coder telle qu'on peut l'étudier en construisant l'histogramme des valeurs à coder. Pour une image, Photoshop sait faire ça très bien comme vous le savez.

En fait l'examen des histogrammes de quelques images comme celles que nous utilisons journalièrement est assez encourageant : il y a toujours dans les images naturelles des variations de fréquence significatives et même malheureusement dans les bruts de scan des valeurs inutilisées.

On va donc s'intéresser aux VLC (Variable Length Code), on dit aussi au codage entropique.

Nous voyons bien l'intérêt de la méthode mais la mise en œuvre est un peu plus délicate. Prenons un petit exemple.

Supposons que nous ayons à traiter un message qui ne contient que 4 caractères que nous nommerons A, B, C et D. et que les fréquences de ces caractères dans notre message soient les suivantes :

A : 60 %

B : 30 %

C : 5 %

D : 5 %

On peut imaginer le code suivant :

A : 0    B : 00    C : 01    D : 10

Il nous permettra assurément de gagner de la place mais si nous recevons le message suivant : 00001, comment l'interpréter ?

4 A ?    2 B ?    2 A et 1 B ?    1 B et 2 A ? Ça ne marche pas ! Nous venons de découvrir le problème de la synchronisation.

Comment le résoudre ?

Utiliser des caractères séparateurs ? c'est contradictoire avec l'objectif de compression.

La solution c'est le **VLC préfixé**.

Un code est préfixé s'il n'est le début d'aucun autre. Le VLC que nous avons tenté d'utiliser tout à l'heure n'était pas préfixé.

Revenons à notre exemple et essayons de corriger.

A : 0    B : 10    C : 110    D : 111 Ça marche ! Est-ce la solution optimale ? Comme nous l'avons obtenue par tâtonnement, on n'en sait rien mais rassurez-vous la théorie permet d'affirmer que oui.

Nous allons maintenant prendre un exemple un peu plus complexe et présenter une méthode permettant de déterminer un VLC efficace (même s'il n'est pas toujours optimal).

Supposons que nous voulions transmettre le message suivant :

### **LE PRESIDENT EST ENTRE DANS LA SALLE**

Il compte 36 caractères et occupe dans un logiciel comme XPress ou WordPerfect 36 octets.

Nous allons compter les occurrences des différents caractères de l'alphabet utilisés et les classer par fréquences décroissantes, nous obtenons la liste suivante :

E : 7

Espace : 6

L : 4

S : 4

N : 3

T : 3

A : 3

R : 2

D : 2

P : 1

I : 1



complexe mais qui parvient au même résultat, celui de **Huffman**.

Ces méthodes sont utilisées directement dans les photocopieurs et elles interviennent en PAO comme méthodes complémentaires. Il y a par exemple un codage de Huffman dans la méthode de compression JPEG.

## Les algorithmes de type dictionnaire

Les méthodes statistiques, en particulier l'algorithme de Huffman, ont été très employées jusqu'en 78. C'est alors que sont apparues les méthodes de type dictionnaire .

Nous avons commencé, tout à l'heure, à compresser en tenant compte des répétitions d'octets mais dans les images que nous manipulons ce sont souvent des séquences d'octets, des motifs qui se répètent. Pourquoi ne pas utiliser ces répétitions pour compresser les données ?

Prenons un petit exemple pour faire comprendre l'intérêt de la méthode.

Combien la langue française comporte-t-elle de mots de 10 lettres ? environ 10 000.

Combien peut-on faire de mots différents de 10 lettres avec les 26 lettres de l'alphabet ?

$26^{10}$  soit  $1,4 \cdot 10^{14}$  ce qui fait tout de même 140 000 milliards de mots. Autrement dit, les mots de 10 lettres qui ont un sens représentent 1 quatorze milliardième de l'ensemble des mots de 10 lettres.

Il est donc très judicieux si l'on veut coder efficacement ces mots, d'en dresser une liste et de les numéroter, on dit plutôt de les indexer.

## Les algorithmes LZ\*\*

C'est en 1977 que Jacob Ziv et Abraham Lempel ont publié un article qui est à la base de tous les algorithmes à dictionnaire que nous utilisons actuellement.

En 1984, l'américain Terry Welch améliore l'algorithme précédent et dépose un brevet. C'est la naissance de l'algorithme LZW. Il est plus performant que les méthodes statistiques. Il sert de base à la norme V42bis du CCITT que nous utilisons dans nos modems.

Le principe de l'algorithme LZW, c'est d'utiliser un dictionnaire dynamique qui contient des motifs du fichier traité.

L'inconvénient des algorithmes à dictionnaire, c'est qu'il est à priori nécessaire de transmettre le dictionnaire avec les données, ce n'est pas le cas avec LZW. Le dictionnaire est construit à la compression et reconstruit à la décompression.

Ce qui est également intéressant, c'est qu'il est inutile de lire et d'étudier le fichier avant de le compresser. La compression s'effectue au fur et à mesure de la lecture, le compresseur détectant au fur et à mesure les séquences qui se répètent.

Nous n'explicitons pas cet algorithme mais nous retiendrons deux points :

- d'une part, il s'agit d'un algorithme sophistiqué et « pilotable ». Il est possible par exemple de remettre à zéro le dictionnaire ou de modifier sa taille.
- d'autre part, on peut combiner LZW et algorithme de codage statistique. C'est ce que font des programmes comme ARJ ou PKZIP qui utilisent LZW puis Shannon-Fano.

## LES METHODES DE COMPRESSION AVEC PERTES

Le principe est clair : nous acceptons une dégradation de l'image décompressée - indiscernable à l'œil ou suffisamment faible pour être acceptable - en contrepartie d'un taux de compression beaucoup plus intéressant.

Il existe différentes approches de la compression avec perte. L'une d'entre elle, représentée par l'algorithme JPEG, a occupé le devant de la scène depuis quelques années en raison de ses performances et de la facilité de son implémentation sur micro. Cette situation est trompeuse et d'autres méthodes déjà connues (ondelettes et fractales en particulier) vont s'imposer dans les années qui viennent. Il est difficile de prédire ce qui interviendra ensuite mais d'autres voies de recherche sont explorées (utilisation des réseaux neuromimétiques par exemple).

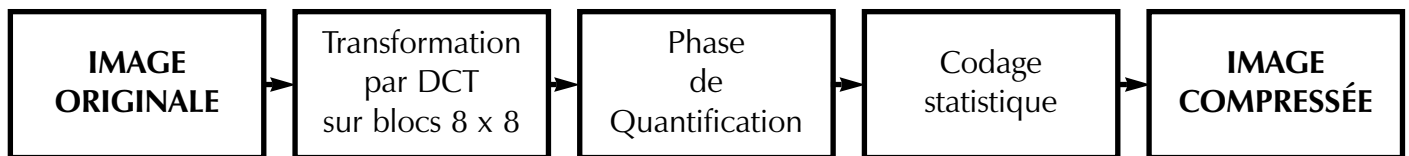
### **L'algorithme JPEG**

Nous donnerons d'abord une idée relativement précise de l'algorithme JPEG.

Ce nom (Joint Photographic Expert Group) provient du groupe d'experts internationaux qui a établi, en 1991, la norme que nous utilisons actuellement. En fait, la DCT (Discret Cosin Transform) qui est au cœur de la méthode a été proposée en 1974 par le professeur Rao de l'université du Texas en 1974.

La norme JPEG de 91 décrit le format des données compressées et le schéma de codage et de décodage. Les algorithmes de compression sont proposés mais n'ont pas de valeur normative.

Voyons d'abord le schéma de principe :



Nous allons maintenant suivre ce qu'il advient d'un bloc de 64 pixels qui a été extrait d'une image, j'emprunte cet exemple à Xavier Marsault (ouvrage cité en bibliographie).

Prenons le bloc suivant :

```
100 155 131 116 151 135 131 211
120 135 127 88 155 131 155 179
120 135 151 100 179 116 155 167
120 155 151 108 191 112 155 179
135 151 135 120 167 112 179 179
120 151 155 151 151 116 179 179
135 151 167 167 151 151 167 171
120 151 179 151 151 131 155 167
```

Nous allons maintenant soustraire 128 de chaque valeur et appliquer la transformation DCT. Le résultat est le suivant :

145	-84	34	-69	4	-66	-35	72
-45	-28	28	19	10	-54	5	15
0	-2	-8	-15	-9	0	30	-41
9	-14	15	-11	5	8	-12	-32
1	1	3	-11	7	-23	-4	0
18	4	-17	-10	4	-10	7	-10
-5	1	-7	-20	1	-1	-3	5
3	1	1	9	2	7	2	-2

Notons qu'à cette étape, aucune information n'a été perdue. Le bloc original peut être reconstitué sans aucune perte en utilisant la DCT inverse et en ajoutant 128 à chaque terme.

Quel est l'intérêt de cette transformation ?

On observe que les coefficients de forte valeur absolue sont situés en haut et à gauche, l'importance des coefficients pour la reconstitution de l'image diminue quand on se déplace en diagonale du haut à gauche vers le bas à droite.

Nous allons passer à l'étape suivante, celle de la quantification. De quoi s'agit-il ? De transmettre des valeurs approximatives de la matrice précédente en se donnant un pas de quantification. Prenons un exemple.

Supposons qu'une variable puisse prendre les valeurs 1, 2, 3, 4, 5,.... Si l'on prend un pas de quantification de 3 on ne gardera que le quotient de la division euclidienne de la valeur par 3. C'est très simple !

On va remplacer :

0, 1 et 2 par 0

3, 4 et 5 par 1

6, 7 et 8 par 2

9, 10 et 11 par 3 et ainsi de suite.

C'est à cette étape que nous allons perdre de l'information et nous allons la perdre d'une manière « astucieuse » parce que le pas de quantification dont dépend la précision de l'image restituée va dépendre de la position de la valeur dans la matrice.

Nous allons prendre un pas relativement petit pour les valeurs

importantes (en haut à gauche) et prendre un pas de plus en plus grand au fur et à mesure qu'on descend vers le bas et la droite de la matrice.

L'ensemble des pas qui vont être utilisés constituent ce que l'on appelle une matrice de quantification.

Certaines ont été construites en fonction de critères psycho-visuels, nous allons en fabriquer une avec une petite formule :

$$Q(i,j) = 1 + (1 + i + j) \times Fq$$

(pour les matheux, on indiquera une formule un peu plus complexe, permettant d'obtenir un grand nombre de matrice différentes :  $Q(i,j) = 1 + (1 + \mu (i^n + j^n)) \times Fq$ . Par souci de simplification, on a pris ici  $\mu = n = 1$  et  $Fq = 5$ . On peut bien entendu compliquer...)

Nous prendrons  $Fq = 5$ . Il s'agit d'un facteur de qualité, c'est celui que vous modifiez quand vous choisissez la qualité de la restitution dans un logiciel comme Photoshop.

Voici la matrice de quantification que nous obtenons :

6	11	16	21	26	31	36	41
11	16	21	26	31	36	41	46
16	21	26	31	36	41	46	51
21	26	31	36	41	46	51	56
26	31	36	41	46	51	56	61
31	36	41	46	51	56	61	66
36	41	46	51	56	61	66	71
41	46	51	56	61	66	71	76

Nous allons maintenant diviser les valeurs de la matrice de données par les valeurs de la matrice de quantification. On obtient cette matrice :

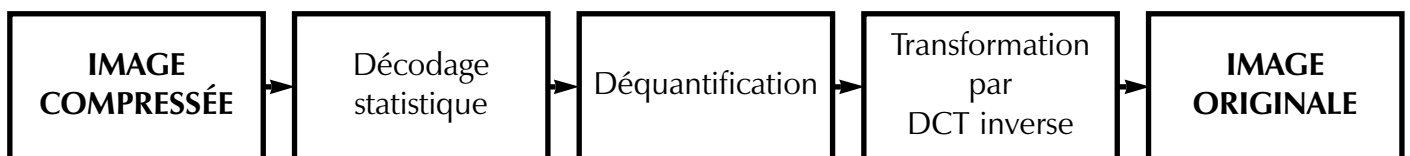


On utilise une méthode de compression sans perte qui peut être celle de Huffman, c'est le cas habituel ou bien encore un algorithme de compression arithmétique qui est breveté par IBM.

Nous avons terminé, restera à indiquer au décodeur quelle matrice de quantification a été utilisée.

Nous allons suivre la décompression du bloc que nous venons de compresser.

Voici le schéma général :



Après le décodage statistique nous retrouvons la matrice :

24	-7	2	-3	0	-2	0	1
-4	-1	1	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	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

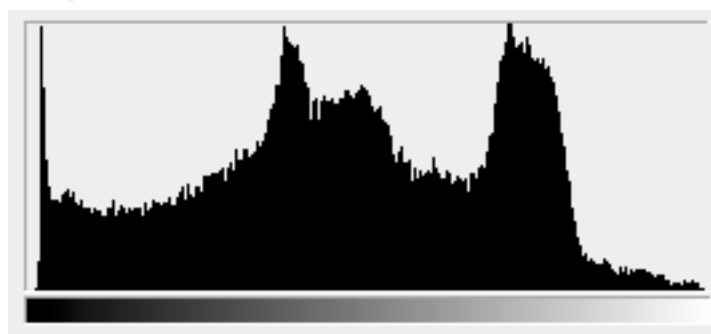
Nous allons la déquantifier en multipliant chaque terme par le coefficient de la matrice de quantification correspondant. On obtient :

144	-77	32	-63	0	-62	0	41
-44	-16	21	0	0	-36	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

Nous allons maintenant appliquer la DCT inverse et ajouter 128 à chaque terme, ce qui permet d'obtenir le bloc décompressé :

112	145	137	107	149	130	139	124
114	145	139	110	149	131	141	183
117	145	143	116	151	133	144	181
121	145	148	124	152	135	148	179
126	145	153	132	154	137	152	177
130	145	158	139	155	139	156	175
133	145	162	145	157	141	159	173
135	146	164	148	157	142	161	172

Nous pouvons également faire une approche plus pratique de la compression JPEG. Prenons une image réelle avec un histogramme tel que nous avons l'habitude d'en voir :



Enregistrons cette image en JPEG avec trois réglages différents dans la fenêtre que nous obtenons en faisant « enregistrer sous » :

- Qualité maximale (réglage 10)
- Qualité moyenne (réglage 6)
- Qualité basse (réglage 0)

Les taux de compression obtenus sont :

$$\sigma_{10} = 1,8$$

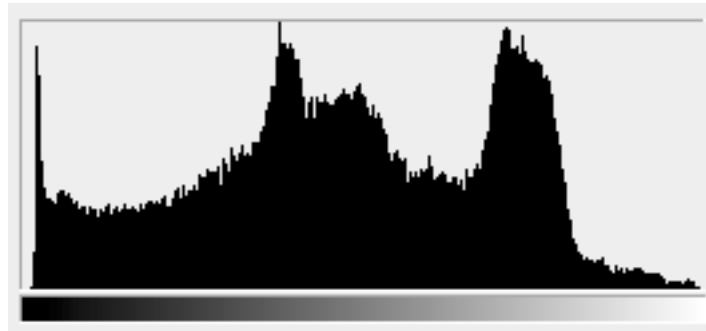
$$\sigma_6 = 8,2$$

$$\sigma_0 = 23$$

L'examen des 3 images décompressées à l'écran permet d'abord d'observer leur très bonne qualité.

En observant l'image de plus basse qualité à un fort grossissement, on voit apparaître très nettement l'effet de mosaïque, typique du JPEG. Encore faut-il noter qu'il sera beaucoup moins accentué si nous imprimons l'image, les pixels étant convertis en points de trame au moment du flashage. Le choix du facteur de qualité lors du calcul de la résolution d'acquisition est dans ce domaine déterminant.

L'examen de l'histogramme de l'image décompressée en qualité maximale :



permet de constater que les modifications, modestes, concernent tous les niveaux de gris.

On peut essayer d'étudier statistiquement l'erreur résultant de la compression avec Photoshop.

Fabriquons d'abord une image à deux calques. Le calque de fond correspond à l'image originale, le calque placé au-dessus correspond à notre image décompressée après une compression de qualité maximale.

En utilisant le mode « différence », nous pouvons créer une image

qui apparaît complètement noire à l'écran. Les niveaux de ses pixels correspondent à la valeur absolue de la différence entre les deux calques. Examinons son histogramme :



On peut mesurer ici combien les modifications introduites par la compression sont faibles : la moyenne, dans notre exemple, est à 0,6, la médiane à 1. Aucun écart n'est supérieur à 2, 13 seulement sont égaux à 2.

La même opération répétée avec l'image compressée au plus fort taux donne l'histogramme suivant :



Malgré l'importance de la compression, la moyenne et la médiane sont à 4. On notera que la méthode employée ici, intéressante parce que « graphique », ne permet pas de calculer l'EQM.

Ajoutons que l'application d'un seuillage ou d'une courbe adaptée permet de visualiser la répartition spatiale de l'erreur.

## La compression fractale

Vous connaissez tous les images fractales. ce sont des images caractérisées par leur autosimilarité. Elles sont générées à partir d'une équation et de quelques paramètres. On peut bien entendu construire de très nombreuses images fractales.

Si l'on veut compresser une telle image, la méthode semble s'imposer : transmettre la formule qui l'a générée.

Ce raisonnement mène tout naturellement à se poser une question plus difficile mais aussi plus intéressante. Étant donné une image

quelconque, est-il possible de trouver une formule ou quelques formules qui permettent de reconstruire cette image ?

On démontre que oui. Chaque image possède une formule qui permet de la reconstruire. C'est un théorème d'existence mais nous n'avons aucune méthode qui nous permette de déterminer cette formule.

C'est seulement en 1986 qu'un chercheur de l'Institut d'Atlanta, Michael Barnsley, a proposé une méthode qui permet d'approcher cette formule d'une manière utile à la compression.

En pratique, pour des taux inférieurs à 50, la compression fractale donne des résultats plutôt inférieurs au JPEG. Pour des taux plus importants, c'est l'inverse. La compression JPEG donne des images peu ou pas reconnaissables tandis que les images résultant de la compression fractales restent lisibles.

Comment ça fonctionne ?

On découpe l'image en blocs-parents carrés de 16 pixels de côté et on les découpe à leur tour en 4 blocs-fils de 8 pixels de côté.

Pour chaque blocs, on va calculer son attracteur de manière approximative, c'est à dire un couple de fonctions qui appliqué itérativement à un bloc quelconque permet de converger vers le bloc...

Pour améliorer la méthode on compare l'attracteur de chaque bloc-fils avec les attracteurs des blocs-parents. Si ça marche, on codera la référence du bloc correspondant et non plus l'attracteur ce qui prend moins de place.

C'est pour cela que l'on dit parfois que la compression fractale permet de coder une image par elle-même.

En conclusion, on peut dire que cette méthode fait encore l'objet de recherches actives, elle permet un taux de compression intéressant mais elle reste malheureusement très lente dans la phase de compression.

## **La compression par ondelettes**

Les ondelettes c'est d'abord une théorie mathématique récente d'analyse du signal, développée dans les années 80. On peut considérer qu'il s'agit d'une extension de l'analyse de Fourier.

On a un signal continu et on le décompose en une série de nombres qui décrivent des courbes qui s'additionnent pour reconstruire le signal. L'intérêt de cette théorie est au départ l'analyse des signaux et elle a déjà de nombreuses applications.

La différence entre l'analyse de Fourier et les ondelettes, c'est que l'analyse de Fourier utilise uniquement des sinusoides alors que dans la décomposition en ondelettes on utilise des fonctions plus complexes que l'on déforme. C'est un outil plus souple mais plus complexe.

Quand on enregistre une image en utilisant les ondelettes, on divise sa résolution par deux et on code l'information perdue par des coefficients d'ondelettes. On commence donc à coder les détails les plus fins (les hautes fréquences). On recommence l'opération autant de fois que nécessaire jusqu'à ce que l'image se réduise à 1 pixel. A chaque étape l'image est « lissée » et les détails perdus sont codés en coefficients d'ondelettes.

Dans le langage du traitement du signal on dit qu'on applique un filtre passe-bas et qu'on sous-échantillonne le résultat.

Comme on ne perd pas d'information avec cette méthode, on peut toujours reconstruire l'image. A une résolution donnée on ajoute l'information codée avec les ondelettes à cette étape et l'on obtient une image à la résolution supérieure.

En conclusion on retiendra les points suivants :

- Cette méthode n'entraîne pas d'effet de mosaïque.
- L'algorithme est plus simple et plus souple que JPEG et donc plus rapide.
- Il est possible d'avoir des images très compactes (de l'ordre du ko !).
- Une image compressée par les ondelettes peut être décompressée de deux manières différentes : sa résolution est fixe mais sa taille augmente progressivement, sa taille est fixe mais sa résolution augmente progressivement.

Peut-être ceux qui connaissent la théorie des ondelettes ne perçoivent-ils pas le rapport entre l'algorithme qui vient d'être sommairement exposé et la théorie. En fait ce que nous venons d'expliquer est une application de la transformation en ondelettes rapide. Le rapport n'est pas si évident puisque cette manière de traiter les images avait été découverte indépendamment de la théorie des ondelettes (algorithmes pyramidaux de Burt et Adelson).

## LE FORMAT JPEG 2000

Il faut d'abord rappeler que le format JPEG actuel a été développé il y a plus de 10 ans et qu'il est basé sur les technologies des années 70 (le groupe JPEG a commencé à travailler en 1987 et la norme est parue en 1994).

Les technologies ne sont plus les mêmes aujourd'hui car la recherche a évolué et d'autre part, les besoins ne sont plus les mêmes non plus. Il faut répondre au développement de la photo numérique, du web et du commerce électronique par exemple.

On a aujourd'hui besoin :

- d'un format ouvert permettant l'**échange** d'images **entre des logiciels et des équipements différents**,
- d'un format utilisant des **techniques de compression modernes** dans le but de transmettre des contenu plus riches en haute résolution,
- d'un format qui permette d'obtenir, à partir du même fichier, la **même image à des résolutions différentes** pour tenir compte de la bande passante disponible (accéder à l'image avant qu'elle ne soit complètement téléchargée sur internet est un avantage important, de même que la possibilité d'utiliser le même fichier pour une vignette d'aperçu et l'image à haute résolution),
- d'un format qui permette l'**incorporation de métadonnées** répondant à différents objectifs (identification des détenteurs des droits, description du sujet pour indexation, indications sur les conditions de prise de vue pour une meilleure appréciation, profil ICC pour une meilleure restitution des couleurs...)

Un tel programme est difficile à réaliser. Le groupe de travail ISO JPEG 2000 y travaille depuis 1998.

On peut tout de même donner un premier aperçu de ce que devrait être le format JPEG 2000 en présentant 4 de ses caractéristiques principales :

- Il sera basé sur la technologie des **ondelettes**. Le résultat pratique de ce choix devrait être une amélioration modeste mais réelle du taux de compression (de l'ordre de 20 %) et une qualité supérieure de restitution par rapport à ce que permet le JPEG actuel.

Les coefficients d'ondelettes sont plus facile à stocker que les blocs résultants de l'application de la DCT. L'apparition fâcheuse des blocs nécessaires à l'application de la DCT ne pourra plus se produire.

Le choix des ondelettes est aussi le seul qui permette de concilier compression avec et sans pertes.

- Il permettra à l'utilisateur de déterminer jusqu'à quel point les détails de l'image l'intéressent en lui donnant le **choix de la résolution de restitution** (on dit en anglais « level of interest access »). La récupération sans pertes de l'image devrait être possible.

Contrairement à ce qui se passe avec un format comme FlashPix, une image enregistrée avec les ondelettes n'est pas constituée de différents niveaux de résolution mais d'un flux continu d'informations qui reconstitue l'image peu à peu. On peut imaginer différentes implémentations de la fonction de contrôle de l'utilisateur final (je clique quand j'en ai vu assez, je clique sur un bouton « stop »...)

- Il permettra une meilleure restitution des couleurs grâce à l'utilisation des **profils ICC**. On pourra aussi bien enregistrer des images RVB que des images CMJN. L'espace colorimétrique par défaut devrait être sRVB.

- Il permettra l'enregistrement d'un ensemble important et structuré de **métadonnées** qui devraient pouvoir être modifiées sans réécrire l'ensemble du fichier.

Il n'est pas possible aujourd'hui de dire quand la norme JPEG 2000 sera publiée, ce sera sans doute en 2001. Encore faut-il ajouter que les industriels du secteur ont investi des sommes considérables pour développer les logiciels et les matériels qui utilisent la norme JPEG actuelle. Il faudra probablement plusieurs années pour que la nouvelle norme supplante l'ancienne.

## CONCLUSION

La compression des données est appelée à prendre un rôle encore plus important en raison du développement des réseaux et du multimédia.

Son importance est surtout due au décalage qui existe entre les possibilités matérielles des dispositifs que nous utilisons (débits sur Internet, sur Numéris et sur les divers câbles, capacité des mémoires de masse,...) et les besoins qu'expriment les utilisateurs (visiophonie, vidéo plein écran, transfert de quantités d'information toujours plus importantes dans des délais toujours plus brefs). Quand ce décalage n'existe pas, ce qui est rare, la compression permet de toutes façons des économies.

Nous ne nous sommes intéressés au cours de ce bref exposé qu'aux méthodes de compression des images fixes. Des recherches très actives se poursuivent évidemment dans le domaine des images animées et du son.

Les méthodes déjà utilisées couramment dans nos métiers sont efficaces et sophistiquées (Huffman, LZW, JPEG). La norme JPEG 2000 constituera un progrès important avec l'utilisation des ondelettes mais ne sera qu'une étape.

Les méthodes du futur sauront sans doute s'adapter à la nature des données à compresser et utiliseront l'intelligence artificielle.

# BIBLIOGRAPHIE

**BURKE HUBBARD Barbara.** *Ondes et ondelettes. La saga d'un outil mathématique.* Pour la science. Diffusion Belin. Collection «Sciences d'avenir». *Un excellent petit ouvrage sur un sujet nouveau et difficile, qui a obtenu le prix d'Alembert 1996. L'auteur fait un réel effort de vulgarisation pour faire comprendre une théorie mathématique pleine de promesses dans le domaine de la compression des données en particulier. Permettant plusieurs niveaux de lecture, cet ouvrage s'adresse à un large public.*

**GUILLOIS Jean-Paul.** *Techniques de compression des images.* Collection informatique. Hermes, Paris, 1996. *Excellent ouvrage (niveau mathématique requis bac + 2) qui décrit dans le détail les méthodes actuelles de compression des données. On pourra compléter son information avec l'ouvrage de Xavier Marsault que nous citons par ailleurs.*

**MARSAULT Xavier.** *Compression et cryptage des données multimédias.* Deuxième édition revue et augmentée. Collection «Traité des Nouvelles Technologies. Série Informatique». Hermes, Paris, 1995. *Ouvrage très intéressant pour ceux que ne veulent pas s'en tenir à des généralités sur le sujet (niveau mathématique requis bac+2). Il traite de l'analyse quantitative et qualitative de l'information, des algorithmes de compression statistiques et dictionnaire, des normes JBIG, JPEG et MPEG, de la compression fractale, de la cryptographie à clés secrètes et révélées, de la crypto-compression et de la compression de l'information génétique. (fichiers sources en C)*

**NELSON Mark.** *La compression de données. Textes. Images. Sons.* Collection Science informatique. Traduction de Hervé Soubara. Dunod Tech, Paris, 1993. *Ouvrage classique sur le domaine (codage de Shannon-Fano et de Huffman, méthodes à base de dictionnaire, compression destructive, JPEG,...). Manquent les développements les plus récents (ondelettes, fractales,...).*

**LECOMTE Daniel, COHEN Daniel, BELLEFONDS Philippe de et BARDA jean.** *Les normes et les standards du multimédia.* Dunod, 1999. *Un ouvrage récent très sérieux qui apporte beaucoup d'informations sur des sujets d'actualité.*



**Sylvain Renard**

Après des études de lettres, de mathématiques et d'informatique, Sylvain Renard a enseigné le français avant de créer son entreprise de communication. Il s'est ensuite tourné vers le conseil et la formation dans le domaine de la PAO.

Il intervient aujourd'hui dans des centres de formation et dans de nombreuses entreprises pour animer des stages de formation continue.

Il a décidé il y a quelques années de se former aux techniques du multimédia et de les enseigner. Créateur de « **La Tour des ruses** », site consacré à la PAO et au multimédia qui regroupe de nombreux professionnels passionnés de l'image numérique, Il a réalisé le site du mensuel « Alternatives Économiques » ([www.alternatives-economiques.fr](http://www.alternatives-economiques.fr)).

Il participe à l'animation du club Photoshop.

**Formations assurées** : XPress, InDesign, Illustrator, Photoshop, ImageReady, Painter, Acrobat, Golive, conception et réalisation de sites web...

**Domaine d'intervention en production** : édition, sites web, mise en place de solutions de paiement sécurisé, hébergements de sites et prestations complémentaires, travaux spéciaux (transcodage, conversion XPress-HTML, travaux multilingues)...

### **Sylvain Renard**

58, bd Henri Dunant. Bât. A.

91100 Corbeil-Essonnes.

Tél. : 01 64 96 02 31.

Mobile : 06 60 45 19 73

Mel : [srenard@ruses.com](mailto:srenard@ruses.com)

Site : [www.ruses.com](http://www.ruses.com)