Chapitre IV

Concepts de base de la compression de données

1. Introduction

La compression de données (texte, son, image, …etc.) est nécessaire surtout pour :

- La diminution de leur taille pour pouvoir les enregistrer sur un support de petite capacité.
- Ainsi, depuis quelques années, le développement de réseaux et la démocratisation d'internet font que la compression est toujours d'actualité. En effet, seul un transfert rapide des données permet d'avoir une vitesse d'affichage élevée des pages Webs.

2. Définition

La compression de données est la procédure par laquelle nous réduisons le volume de données à transmettre ou à stocker sans une perte cruciale de l'information. La compression de données est un processus de détection et d'élimination de la redondance dans un ensemble de données. Son but principal est d'atteindre une réduction maximale de volume de données, en préservant les informations significatives après reconstruction.

La méthode de compression dépend intrinsèquement du type de données à compresser : on ne compressera pas de la même façon une image qu'un fichier audio...

3. Types de compressions et de méthodes

3.3. Compression sans perte

Le résultat après décompression est complètement équivalent à l'originel (compression réversible). Ces méthodes de compression sont utilisées pour les données informatiques dont l'exactitude est cruciale (Exemple : données textuelles, programmes…). Normalement, ces algorithmes n'aboutissent qu'à des valeurs faibles de compression.

3.3.1. Codage de Huffman

David Huffman a proposé en 1952 une méthode statistique qui permet d'attribuer un mot de code binaire aux différents symboles à compresser (pixels ou caractères par exemple). La longueur de chaque mot de code n'est pas identique pour tous les symboles :

- les symboles les plus fréquents sont codés avec mots de code courts,
- les symboles les plus rares reçoivent de plus longs codes binaires.

On parle de codage à longueur variable (*VLC* ; *Variable Code Length*) préfixé.

Exemple

Soit une source qui prend 04 valeurs différentes A, B, C et D, avec les probabilités fournies par la *table 1* ci-dessous. Un codage de type fixe des valeurs nécessiterait évidemment 2 bits.

Symbole	A	B	C	D
Probabilité	0.10	0.20	0.40	0.3
Codes	00	01	10	11

Table 1 – *Probabilités et codes associés aux quatre valeurs.*

L'algorithme de Huffman fournit l'arbre binaire représenté sur la *figure 1*. La table de codage résultante est reprise dans la *table 2*.

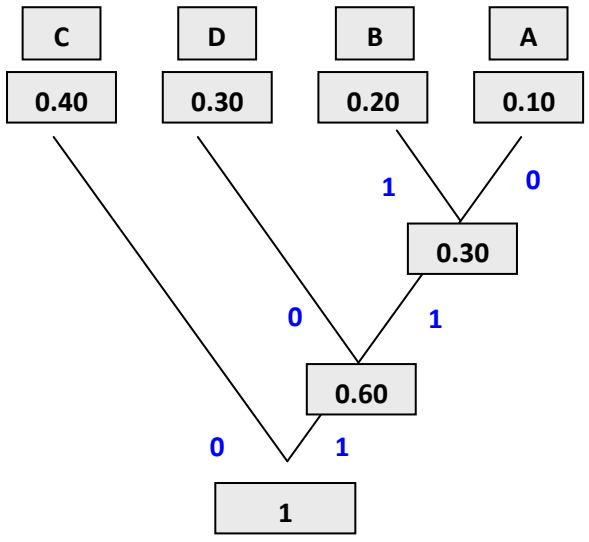


Figure 1 – *Illustration de l'algorithme de Huffman.*

Probabilité	0.1	0.2	0.4	0.2
Symbole	A	B	C	D
Code de Huffman	110	111	0	10

Table 2 – *Code de Huffman de la table 1.*

3.3.2. Codage RLE

Le codage RLE (*Run Length Encoding*) est basé sur le principe de compter le nombre de caractères identiques successifs et à transformer cette succession en indiquant le nombre de répétitions de l'information à répéter ainsi qu'un caractère spécial informant de la présence d'une répétition.

Cette méthode n'a d'intérêt que pour les données contenant souvent des répétitions de plus de trois caractères. Sinon elle peut même augmenter la taille des données source.

Exemple

1. AAABBREEEGGG = 12 octets, après codage; #3A#2BR#3E#3G = 13 octets $\Rightarrow$ inutile.

2. 0000001111100000 = 16 octets, après codage; #60#51#50 = 9 octets.

3.4. Compression avec perte

Par opposition à la compression sans pertes, la compression avec pertes (*Lossy Compression*) se distingue par l'élimination de quelques informations, ce qui permet d'avoir de meilleurs taux de compression.

Le résultat doit être le plus proche possible des données originelles. Etant donné que ce type de compression supprime des informations contenues dans les données à compresser, on parle généralement de méthodes de compression irréversibles.

3.4.1. Modulation différentielle

Le principe de la codification DPCM (Differential Pulse Code Modulation) consiste à faire une prédiction numérique du signal à coder. De cette façon, on code la différence entre chaque échantillon et la valeur respective estimée pour cet échantillon.

3.4.2. Quantification uniforme

Pour simplifier, on suppose pour cela que la variable X est bornée, et prend ses valeurs dans un intervalle ***I*** entre $\pm x_{\max}$. Une quantification uniforme consiste à découper ***I*** en M = 2^N sous-intervalles de taille constante, notée q. Le pas de quantification *q* peut s'exprimer en fonction des valeurs extrêmes $\pm x_{\max}$ par :

$$q = \frac{2x_{\max}}{2^N} \qquad (1)$$

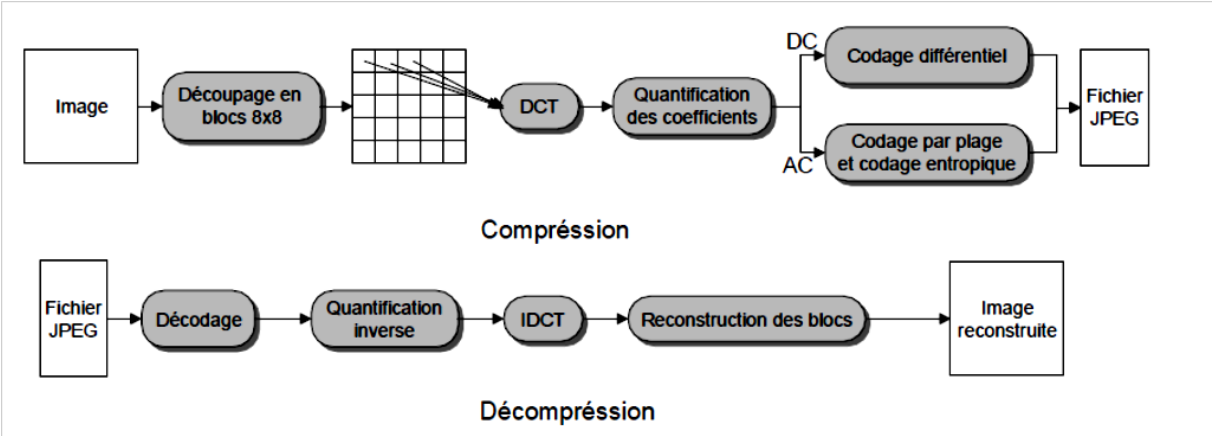
où N est le nombre de bits de quantification.

6. la méthode JPEG :

Le sigle JPEG veut dire « Joint Photographic Experts Group », il représente actuellement le standard de compression avec perte le plus utilisé pour les images naturelles. Cette norme de compression mondiale d'images fixes est apparue à la fin des années 80.

La transformée choisie pour la norme est la transformée en cosinus discrète appelée DCT, elle n'est pas appliquée sur toute l'image mais sur des blocs de l'image de taille fixes (8 x 8). Suivent ensuite une étape de quantification des coefficients DCT et un codage de ceux-ci.

La phase de décompression se résume par la succession inverse des étapes nécessaires µa la compression. Le but des prochains paragraphes est de présenter de manière plus précise les différents blocs de ce diagramme.



6.1. La transformation en cosinus discrète

La formule qui permet de transformer une image *I*(x; y) en sa transformée *T*(u; v) est la suivante :

$$T(u,v) = \frac{2}{N}C(u)C(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\cos\left[\frac{\pi}{2N}u(2x+1)\right]\cos\left[\frac{\pi}{2N}v(2y+1)\right]I(x,y) \qquad (2)$$

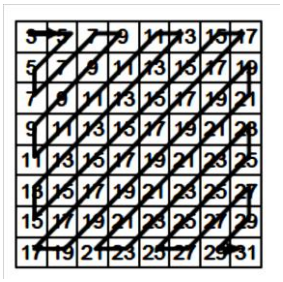
Avec $C(0) = 1/\sqrt{(2)}$ et $\forall \alpha \neq 0\ C(\alpha) = 1.$

La transformée inverse de la DCT a pour équation :

$$I(x,y) = DCT^{-1}(T(u,v)) = \frac{2}{N} C(x)C(y) \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \cos\left[\frac{\pi}{2N} u(2x+1)\right] \cos\left[\frac{\pi}{2N} v(2y+1)\right] T(u,v) \quad (3)$$

Pour répondre à la sensibilité de l'œil, le pas de quantification utilisée est diffèrent suivant la position du coefficient dans le bloc : les basse-fréquences sont affectées d'un pas de quantification plus faible que les hautes fréquences. Ce mode de quantification entraine une accumulation de coefficients nuls dans le coin inferieur droit du bloc DCT.

Afin de faciliter le codage par place des coefficients DCT, il est important de choisir un sens de parcours des coefficients qui permette d'aller des coefficients les plus important vers les coefficients faibles et souvent nulles. Cette opération s'effectue en utilisant un chemin de parcourt en Z, appelé « zig-zag scan ». Le parcours est également représente sur la figure ci-dessous.



Parcours en zig-zag des coefficients quantifiés

6.2. Codage différentiel du coefficient DC

La somme des pixels d'un bloc, représentée par le coefficient **T(0,0)**, est souvent très importante, il est alors plus avantageux d'utiliser un codage différentiel. Ce codage consiste a coder la différence entre le coefficient DC d'un bloc et le coefficient DC du bloc précédent.

6.3. Codage par plage des coefficients AC

Les coefficients AC sont parcourus en zig-zag et ensuite codés par plages afin d'exploiter l'important nombre de coefficients haute-fréquence égales à 0.

6.4. Codage de Huffman

Le codage de Huffman est ensuite utilisé pour coder le résultat du codage par plage ainsi que le codage différentiel des coefficients. Le fichier ainsi crée est le fichier compressé jpg.

7. performances des algorithmes de compression d'images

Le rapport de compression : le paramètre le plus utilisé pour tester la puissance d'un compresseur, noté *CR* (*Compression Ratio*) est le rapport entre la taille de données originelles (avant compression) et la taille de données retenues. Il est défini par la formule suivante :

$$CR = \frac{\text{Nombre de bits originels}}{\text{Nombre de bits retenus}} \tag{4}$$

Le facteur CR n'est pas le seul paramètre utilisé dans les algorithmes de compression, d'autres paramètres peuvent être utilisés, comme

- le taux de compression
- le nombre de bits par pixel (*bpp*).

L'erreur de reconstruction :

Soit *I* une image de dimension M×N et $\hat{I}$ l'image reconstruite correspondante après décompression, l'erreur RMSE est calculé par la formule suivante :

$$RMSE = \sqrt{\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left[I(i,j) - \hat{I}(i,j)\right]^2} \tag{5}$$

avec *i* et *j* représente la position du pixel dans l'image.

La mesure du *SNR* en décibel (*db*) est calculée par l'expression suivante :

$$SNR = 10log_{10}\left(\frac{\sum_{i=1}^{M}\sum_{j=1}^{N} I^2(i,j)}{\sum_{i=1}^{M}\sum_{j=1}^{N}\left[I(i,j) - \hat{I}(i,j)\right]^2}\right) \tag{6}$$

Pour les images 8-bits, le *PSNR* est calculé par l'expression suivante :

$$PSNR = 20log_{10}\left(\frac{255}{RMSE}\right) \tag{7}$$

Le temps de calcul est aussi un paramètre très important, surtout pour la manipulation de données en temps réel.