27 Mars
Giraldo Alvaro Alonso

modification : delann_a & sharma_s 28/03/2013

modification : heftma_r 28/03/2013

modification : renaud_d 29/03/2013

CORRECTION PARTIEL CODO 2014

I - Entropie et ordre des données

Soit le fichier F = {BCADADBCBCADBCAD}

1 - Calculez l'entropie d'ordre 1 de ces données. A cet ordre le fichier est-il compressible ?

La formule de l'entropie est : F(S) = $-\sum_{i=1}^{n} P_i \log_b P_i.$

Rappelons également que : $\forall x \in \mathbb{R}_+^* \quad \log_a(a^x) = x \log_a(a) = x$

A = 4/16 B = 4/16 C = 4/16 D = 4/16

F(S) = - 1/4 x log2(1/4) x 4 = -1 x (-2) = 2

=> Le fichier n'est pas compressible car les caractères sont équiprobables.

2 - Existe t-il une entropie d'ordre supérieur qui révélerait une autre organisation des données ?

On retrouve la même organisation à l'ordre 2, en effet :

BC = 4/8 AD = 4/8.

Néanmoins à l'ordre 4 on obtient :

BCAD = 3/4 ADBC = 1/4

Dans cet ordre là, le fichier est compressible.

II - Compression LZW

* : caractère réservé sur 8 bits, prévenant le décompresseur qu'il doit ensuite lire les données sur 9 bits

d, o, *, 256, 256, r, e, m, i, 260, 256, 262, 260, 264, o

Le dictionnaire par défaut du décompresseur est consituté des 256 caractères ASCII. Le fichier se termine par un caractère de fin dont on ne tiendra pas compte.

1 - Décompressez ces données.

Reçu	Ecrit	Dico	@	phrase
d	/	/	/	/
o	d	do	256	d
*	/	/	/	/
256	o	od	257	do
256	do	dod	258	dodo
r	do	dor	259	dododo
e	r	re	260	dododor
m	e	em	261	dododore
i	m	mi	262	dododorem
260	i	ir	263	dododoremi
256	re	red	264	dododoremire
262	do	dom	265	dododoremiredo
260	mi	mir	266	dododoremiredomi
264	re	rer	267	dododoremiredomire
o	red	redo	268	dododoremiredomirered
FIN	o	/	/	dododoremiredomireredo

2 - Quel est le taux de compression de cette séquence.

To = 22 x 8 = 176 bits

Tc = (3 x 8) + (12 x 9) = 132 bits

=> Gain de 44 bits

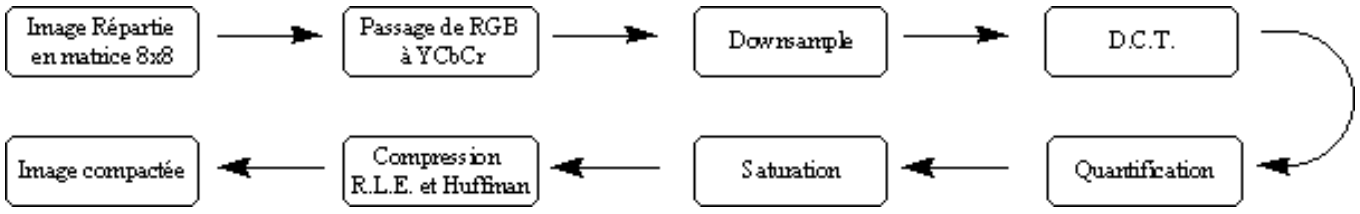
3 - Dans le cadre d'une application où aucun caractère réservé n'est possible (un flux binaire codé sur 8 bits par exemple), quelle solution suggérez-vous pour s'en passer ?

/*renaud_d*/

On commence le codage sur 9 bits, en laissant le poids fort à 0, quand on arrive à l'emplacement ou doit y avoir un marqueur, on passe le poids fort à 1. Et on continue sur 10 bits.

III - Compression JPEG

Fonctionnement général du JPEG (compression)



1 - Quel est le role de la DCT (transformée en cosinus) dans l'algorithme de compression JPEG?

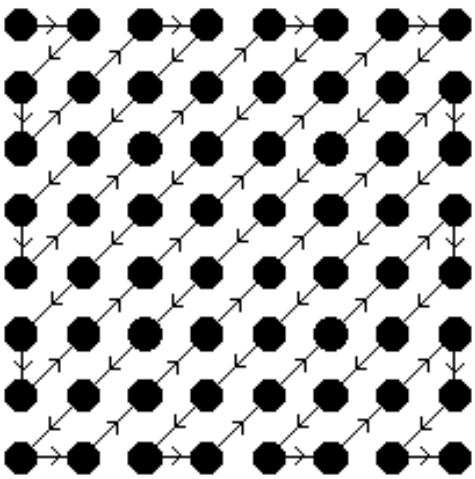
La D.C.T. (*Discrete Cosine Transform*) est une transformée de Fourier dont on a remplacé les sinus par des cosinus. Plus on s'éloigne de l'origine, plus les fréquences correspondantes sont élevées. Or la plupart des images sont composées d'informations de basses fréquences. Donc quand on s'éloigne de l'origine, on trouve des coefficients faibles et qui en plus sont peu importants pour la vision. Ainsi le force de la DCT et de "choisir" les éléments importants de l'image (ce qui semble difficile à faire sur une image non transformée).

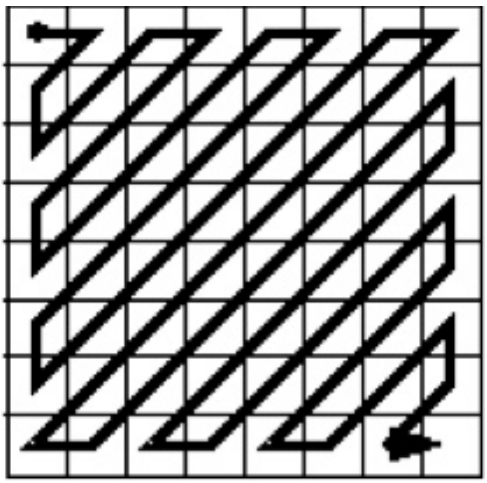
2 - Après un rappel du rôle de la quantification dans la compression JPEG, expliquez la logique de la séquence zigzag lors de la linéarisation du spectre DCT quantifié

Cette étape est la principale étape de perte de données. En effet le principe de la quantification est de créer une matrice de quantification et de diviser chaque terme de la matrice obtenue par la D.C.T. par son terme correspondant de la matrice de quantification

Après la D.C.T. et la quantification l'image n'a pas été réduite. C'est donc pendant la phase de compression que l'on va obtenir un gain de place.

Le codage R.L.E. (*Run Length Coding*) est une méthode de compression très simple. Son principe est de remplacer une suite de caractères identiques par le nombre de caractères identiques suivi du caractère. Par exemple AAAAAAA sera remplacer par 7A. Ici comme dans la matrice issue de la quantification il y a beaucoup de 0, on va appliquer le codage R.L.E. aux 0. Ainsi il faut obtenir les suites de 0 les plus longues possibles. On va donc lire la matrice en *zigzag*.





3 - Après compression d'une image I avec une matrice de quantification Q puis décompression, on obtient une image I'.
I' diffère t-elle de I?
On compresse ensuite I' avec le même algorithme et la même matrice de quantification Q. Après décompression on obtient l'image I"
I" diffère t-elle de I' ?

ps: cette question ne tombera pas au partiel d'après le prof

4 - Pour quelle raison principale JPEG utilise t-il le codage des couleurs YUV et non RVB

Le système RGB n'est pas le mieux adapté pour appliquer le codage J.P.E.G. car l'œil humain est plus sensible à la luminance (ou luminosité) qu'à la chrominance d'une image. Or la luminance est présente dans les trois couleurs rouge, vert , bleu. Ainsi on transforme les composantes RGB de l'image en une composante de luminance notée Y et en deux composantes de chrominance notées U et V.

Ainsi une perte de donnée sur les deux composantes U et V à peu d'effet visible sur l'image, c'est pour cela qu'on effectue le sous échantillonnage (Downsample) sur ces deux composantes.

sources : http://xavier.chassagneux.free.fr/tipe/tipe.htm