

COMPRESSION D'IMAGES

GUILBERT Augustin, MARTINANT Valentin, OZDEMIR Serdar

13 juin 2017

Table des matières

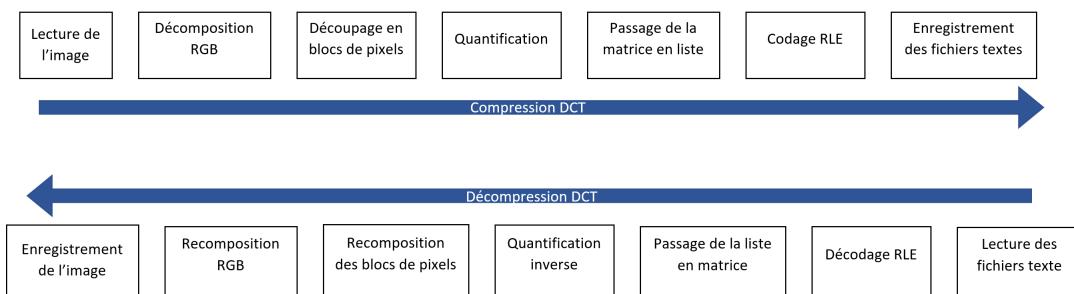
1 Abstract	3
2 Objectifs et types de compression d'images	3
3 Compression JPEG	4
4 Résultats obtenus	7
5 Bibliographie	9

1 Objectifs et types de compression d'images

Les images peuvent être codées « naïvement » : il s'agit d'enregistrer une matrice contenant pour chaque pixel : la valeur de celui-ci si l'il s'agit d'une image en noir et blanc, les valeurs rouge, verte et bleues si c'est une image en couleur. Ce format est simple à mettre en œuvre et facile à décoder, cependant il est très gourmand en stockage. Par exemple pour image couleur de taille 500*500 (taille moyenne), il faut près de 75000 octets soit 600000bits ! Ce qui peut s'avérer problématique pour le stockage de grandes quantités d'images, mais aussi lors des transferts. C'est pourquoi il est intéressant de compresser les images, c'est-à-dire diminuer leur poids. Deux types de compression se présentent à l'utilisateur. D'une part, la compression sans perte, qui permet de conserver une netteté des traits irréprochable. Cette compression est utilisée pour des images nécessitant une très grande précision, tel que les schémas, les dessins techniques ou les balayages médicaux. D'autre part, la compression avec perte privilégie par définition l'économie de poids à la qualité. Ce qui est utile dans le cas de transmission bas débit, ou pour un stockage de masse d'images, cependant la qualité de l'image est dégradée. Le format JPEG (Joint Photographic Expert Group) s'avère être un bon compromis entre une qualité d'image très correcte et une bonne compression. La compression JPEG s'appuie sur la DCT (Discrete Cosine Transform). Nous avons réalisé à l'aide du logiciel de programmation Python une compression de type JPEG.

2 Compression JPEG

La compression JPEG permet une bonne qualité car elle utilise la DCT. En effet, la DCT permet de traduire l'information spatiale des pixels en une information fréquentielle. Or, les variations rapides d'intensités, traduites par les hautes fréquences étant rares, et l'œil humain les percevant mal, elles sont supprimées au cours de la compression. Une image à laquelle on applique la compression JPEG suit le traitement suivant.



La formule de la Transformée en Cosinus Discrète (DCT) se définit de la manière suivante :

$$C(i)(j) = \frac{1}{\sqrt{2N}} \left[\sum_{1 \leq x \leq N-1} \sum_{1 \leq y \leq N-1} p(x, y) \cos \left[\frac{(2x+1)i\Pi}{2N} \right] \cos \left[\frac{(2y+1)j\Pi}{2N} \right] \right]$$

La première étape est la lecture de l'image qui consiste à la convertir en une matrice de dimension 3 (longueur, largeur, nombre de couleurs). Cette étape est suivie dans le cas d'une image en couleur par la décomposition de cette matrice en trois matrices contenant chacune les valeurs des pixels pour une couleur donnée. Il faut ensuite découper cette matrice en blocs carrés, pour des raisons de complexité de calcul on se limitera à une taille de 8, bon compromis entre temps de calcul et précision. Cette opération peut nécessiter le rajout de lignes et de colonnes si le nombre de lignes et de colonnes n'est pas un multiple de 8, les colonnes et lignes rajoutées sont alors des copies de la dernière ligne ou colonne propre à l'image, ainsi elles n'interfèrent pas dans la DCT. Ces blocs de matrices subissent alors un changement de base, permettant de faire apparaître les variations sous forme de fréquences. Mais cela reste encore une opération très coûteuse : il faudrait parcourir chaque matrice et calculer une somme double pour chaque coefficient.

Pour éviter cela, on peut concevoir cette formule comme un changement de base avec la matrice de passage définie de la manière suivante :

$$P(i)(j) = \begin{cases} \frac{1}{\sqrt{2N}} & if \quad i = 0 \\ \sqrt{\frac{2}{N}} \cos\left[\frac{2 * j + 1}{2N}\right] & if \quad i > 0 \end{cases}$$

Ce qui nous donne, une fois calculée numériquement :

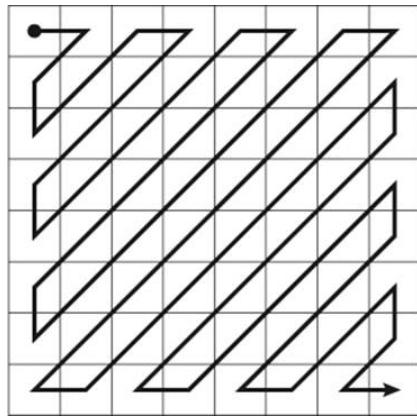
$$P = \begin{pmatrix} .3536 & .3536 & .3536 & .3536 & .3536 & .3536 & .3536 & .3536 \\ .4904 & .4157 & .2778 & .0975 & -.0975 & -.2778 & -.4157 & -.4904 \\ .4619 & .1913 & -.1913 & -.4619 & -.4619 & -.1913 & .1913 & .4619 \\ .4157 & -.0975 & -.4904 & -.2778 & .2778 & .4904 & .0975 & -.4157 \\ .3536 & -.3536 & -.3536 & .3536 & .3536 & -.3536 & -.3536 & .3536 \\ .2778 & -.4904 & .0975 & .4157 & -.4157 & -.0975 & .4904 & -.2778 \\ .1913 & -.4619 & .4619 & -.1913 & -.1913 & .4619 & -.4619 & .1913 \\ .0975 & -.2778 & .4157 & -.4904 & .4904 & -.4157 & .2778 & -.0975 \end{pmatrix}$$

On réalise alors le changement de base suivant la formule suivante : $P.M.P^{-1}$, la matrice de passage n'étant calculée qu'une fois, on réduit la complexité de $O(n^4)$ à celle des fonctions numpy.dot() utilisées dans notre programme pour multiplier plus efficacement

Arrive alors l'étape de la quantification qui consiste à diviser les coefficients de l'image par ceux d'une matrice de quantification : on ne conserve que les coefficients les plus importants : l'arrondi à la partie entière permet d'en réduire une bonne partie à 0. Nous avons retenu celle ci à un niveau de quantification de 10

$$Q_{10} = \begin{bmatrix} 80 & 60 & 50 & 80 & 120 & 200 & 255 & 255 \\ 55 & 60 & 70 & 95 & 130 & 255 & 255 & 255 \\ 70 & 65 & 80 & 120 & 200 & 255 & 255 & 255 \\ 70 & 85 & 110 & 145 & 255 & 255 & 255 & 255 \\ 90 & 110 & 185 & 255 & 255 & 255 & 255 & 255 \\ 120 & 175 & 255 & 255 & 255 & 255 & 255 & 255 \\ 245 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \end{bmatrix}$$

La Matrice obtenue après quantification est ensuite parcourue puis ses éléments ajoutés à une liste selon le schéma suivant :



Cette liste contenant beaucoup de 0 consécutifs, elle subit une RLE (Run-Lenth Encoding), qui consiste à indiquer le nombre de coefficients de même valeur consécutifs et la valeur de ce coefficient. Enfin, ces listes sont enregistrées en tant que fichiers textes qui représentent l'image une fois compressée.

La décompression suit le chemin inverse :

Tout d'abord, une fonction lit et transforme en liste les fichiers textes contenant l'image compressée, puis ces listes subissent une RLE inverse permettant d'obtenir 64 coefficients. On transforme ensuite cette liste en une matrice 8×8 selon le schéma diagonal défini plus haut dans le sens inverse. Les coefficients de la matrice sont ensuite multipliés par ceux de la matrice de quantification, puis on réalise le changement de base inverse. Les blocs de 64 pixels sont alors rassemblés, puis les colonnes et lignes rajoutées sont supprimées afin de donner à la matrice les dimensions de l'image d'origine.

Finalement, les matrices des différentes couleurs sont assemblées puis cette matrice finale est enregistrée au format image pour donner une image finale de même taille que l'originale.

3 Résultats obtenus

Lors de la compression, plus le niveau de quantification de l'image est faible, plus l'image est compressée, est donc la qualité dégradée. Nous avons testé différents niveaux de quantification sur une image afin de déterminer leur impact sur la compression.

Niveau de quantification	Taille originale (en Mo)	Taille compressée (en Mo)	Taux de compression
10	6,22	1,08	5,7
20	6,22	1,26	4,9
30	6,22	1,44	4,3
40	6,22	1,60	3,9
50	6,22	1,76	3,5
60	6,22	1,90	3,3
70	6,22	2,14	2,9
80	6,22	2,56	2,4
90	6,22	3,39	1,8

Taux de compression en fonction du niveau de quantification choisi

Ainsi, la compression de l'image reste imperceptible pour un niveau de quantification supérieur à 50, en effet, un niveau de quantification trop faible entraîne l'uniformisation des pixels de chaque carré de 64 pixels, ce qui dégrade amplement la qualité de l'image. Voici les images obtenus pour des niveaux de compression de 90, 50, et 10.



Image originale



Niveau de compression de 90



Niveau de compression de 50



Niveau de compression de 10

Les compressions avec pertes officielles peuvent avoir des taux de compressions allant de 3 à 100, ce qui est beaucoup plus important que ce que nous réalisons, car nous obtenons un taux maximal de 9 pour des images de très mauvaise qualité. En effet, notre programme n'est pas optimal pour plusieurs raisons : les fichiers textes de la compression sont codés en tant que caractères qui plus est avec des caractères séparateurs et non pas en hexadécimal, privilégié normalement, ce qui permettrait un gain de place, le temps de compression est élevé.

4 Bibliographie

http://lmrs.univ-rouen.fr/Vulgarisation/JPEG/jpeg-DCT.html

http://etud.insa-toulouse.fr/floane_sa/BEmultimedia/index.php?Dct

http://igm.univ-mlv.fr/dr/XPOSE2013/La_compression_des donnees/jpeg.html

http://www-ljk.imag.fr/membres/Valerie.Perrier/SiteWeb/node9.html

https://www.math.cuhk.edu.hk/lmlui/dct.pdf

Image compression and the discrete cosine transform, Ken Cabeen and Peter Gent

5 Abstract

Currently, everybody stores a lot of pictures on numerous devices, like computers or smartphones. Nevertheless, people needs to store more and more pictures, so pictures must be compressed. A compressed picture is lighter than a non-compressed one, but quality is damaging during this process. JPEG format is one of the best because it combines a good compression rate and a good quality. JPEG compression principle is based on DCT which transform pixels information into frequencies. High frequencies mean fast intensity changes, well fast intensity changes are occasional in pictures, and human eyes don't detect it. So, these frequencies are deleted during the compress to create a lighter picture.