Initiation à la compression d'image

(Sans faire trop de maths)

Table des matières

1 rappels sur une image

1.1 Courte introduction :

Ce document s'adresse aux néophytes et non informaticiens. Il a pour objet d'éclaircir un peu la notion de format de fichier image, et de compression d'image. J'ai fait quelques raccourcis volontaires, le but n'étant pas de réécrire un programme de compression !

1.2 un fichier informatique

C'est une bête suite de codes rangés sous la forme de bits, ou, plus pratique à utiliser, d'octets (1 octet = 8 bits). Un octet se représente avec deux caractères. Ci-dessous, il y a donc 16 octets en largeur et 16 en hauteur. Un octet peut valoir entre 0 et 255. Si on souhaite stocker de plus grandes valeurs, on les stocke sur plusieurs octets.

```
FF D8 FF E1 28 88 45 78 69 66 00 00 49 49 2A 00
08 00 00 00 0F 00 00 01 03 00 01 00 00 00 C0 0F
00 00 01 01 03 00 01 00 00 00 D0 0B 00 00 02 01
03 00 03 00 00 00 C2 00 00 00 03 01 03 00 01 00
00 00 01 00 00 00 06 01 03 00 01 00 00 00 02 00
00 00 10 01 02 00 14 00 00 00 C8 00 00 00 12 01
03 00 01 00 00 00 01 00 00 00 15 01 03 00 01 00
00 00 03 00 00 00 1A 01 05 00 01 00 00 00 DC 00
00 00 1B 01 05 00 01 00 00 00 E4 00 00 00 1C 01
03 00 01 00 00 00 01 00 00 00 28 01 03 00 01 00
00 00 02 00 00 00 31 01 02 00 1E 00 00 00 EC 00
00 00 32 01 02 00 14 00 00 00 0A 01 00 00 69 87
04 00 01 00 00 00 20 01 00 00 B4 01 00 00 10 00
10 00 10 00 4E 69 6B 6F 6E 20 43 4F 4F 4C 50 49
58 20 50 37 37 30 30 00 C0 C6 2D 00 10 27 00 00
C0 C6 2D 00 10 27 00 00 41 64 6F 62 65 20 50 68
```

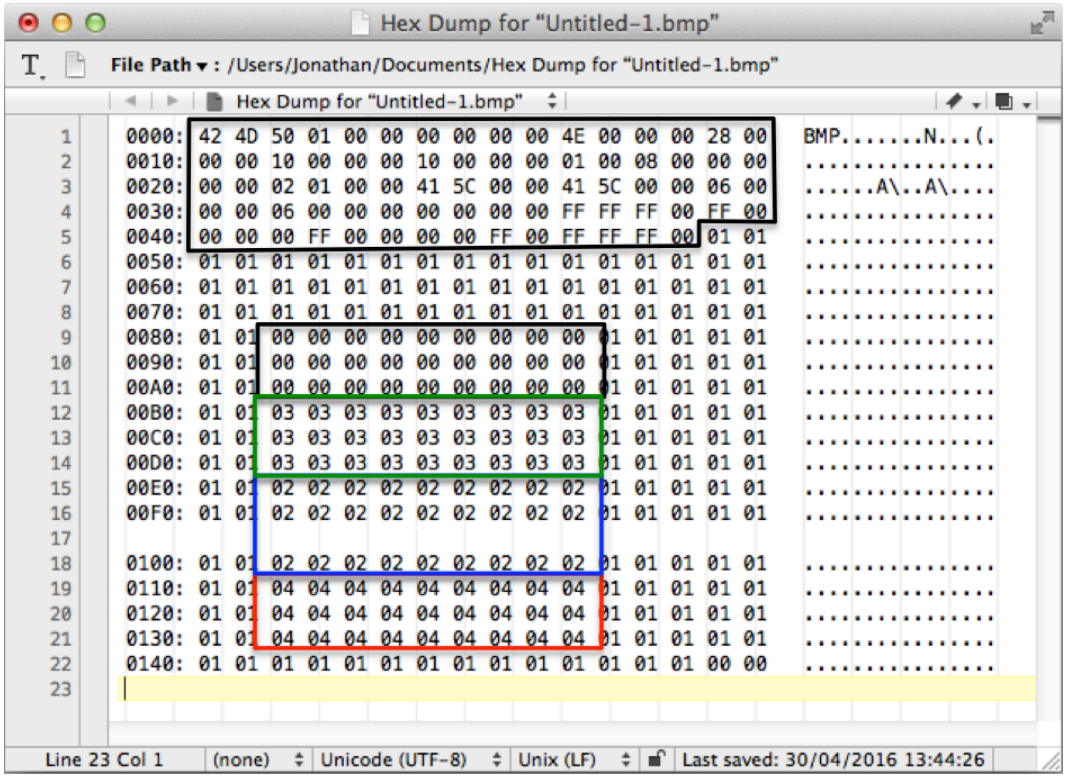
rappel : en mémoire vive d'un ordinateur, 1k octets = 1024 octets, et donc 1M octets = 1 048 576 octets. Ce n'est pas vrai dans les supports de stockage (disque dur, clée USB) où 1Mo = 1000 Ko. Donc un fichier de 1Mo sur disque prendra moins de 1Mo en mémoire vive ! arg !

1.3 une image

L'image ne coupe pas à la règle elle est donc aussi stockée sous la formes d'octets en mémoire.

Voici une image. Elle fait 16 pixels par 16 pixels, et 256 couleurs :

Ci contre, la même image vu depuis l'intérieur du fichier au format « bmp »:



On retrouve encadré en noir l'entête, puis encadré en rouge, bleu, vert, et noir, les rectangles de couleur. On note que l'image est à l'envers.

On peut en déduire que dans cette image, le rouge est codé « 04 », le bleu « 02 », le vert « 03 », le noir, « 00 » et le blanc « 01 ».

Les couleurs sont prédéfinies dans une table et stockée dans l'entête du fichier

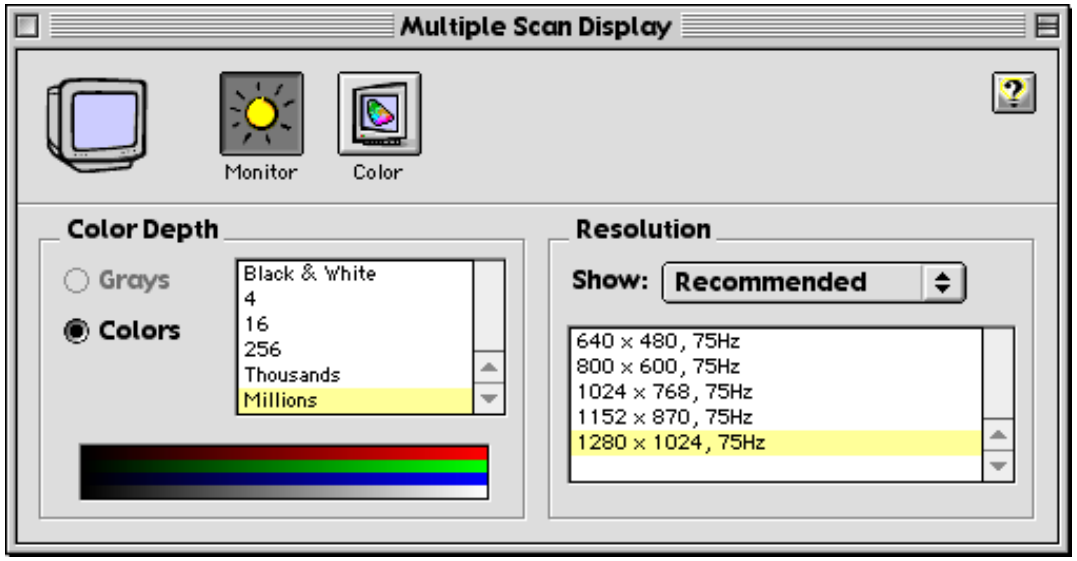
En plus des couleurs, l'entête doit contenir d'autres informations, comme par exemple la largeur et la hauteur de l'image. En effet, comme l'image est stockée sous forme d'octets les uns derrières les autres, on ne sait pas quand commence la ligne suivante.

L'avantage de cette façon d'enregistrer le fichier sur le disque, c'est que c'est facile à programmer et ça ne demande pas de calculs. On envoie tout le contenu de la mémoire dans un fichier, et c'est stocké.

1.4 Codage des couleurs

Avec un octet par pixel, on peut coder 256 couleurs. Pas terrible. Pour pouvoir afficher plus de couleurs, on utilise donc plus d'octets par pixel, ou bien, dit autrement, plus de bits par pixel (1 octet = 8 bits). Ça s'appelle la *profondeur* de l'image.

L'affichage des écrans est aujourd'hui 1 octet par couleur (R, V, B), soit 3 octets par pixel. On appelle ça l'affichage 24 bits. Ça permet d'afficher environ 16 millions de couleurs différentes.

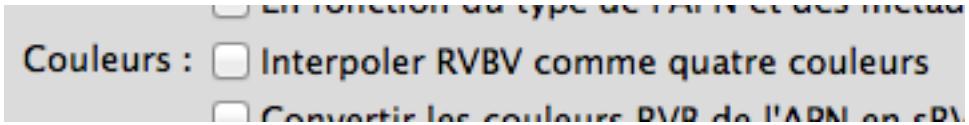


Les images destinées à être regardées sur un écran sont donc codées avec, au maximum, 3 octets par pixel. Il n'y a aucun intérêt à en mettre plus.

Par contre, les appareils photos sont capables de voir plus de couleur que les écrans ne peuvent en afficher. Pourquoi ? Parce qu'augmenter la dynamique ne sert à rien si on n'augmente pas la *profondeur* des capteurs…

De capteurs 8 bits par couleurs, on passe donc à des capteurs 10, 12 ou 14 bits par couleurs…

Et en plus, parfois, il y a 2 cellules vertes pour une rouge et une bleue (RVBV). Du coup avec tout ça, il n'y a plus 3 octets par pixel, mais jusqu'à ***8 octets (64 bits) par pixel à la sortie du capteur***…



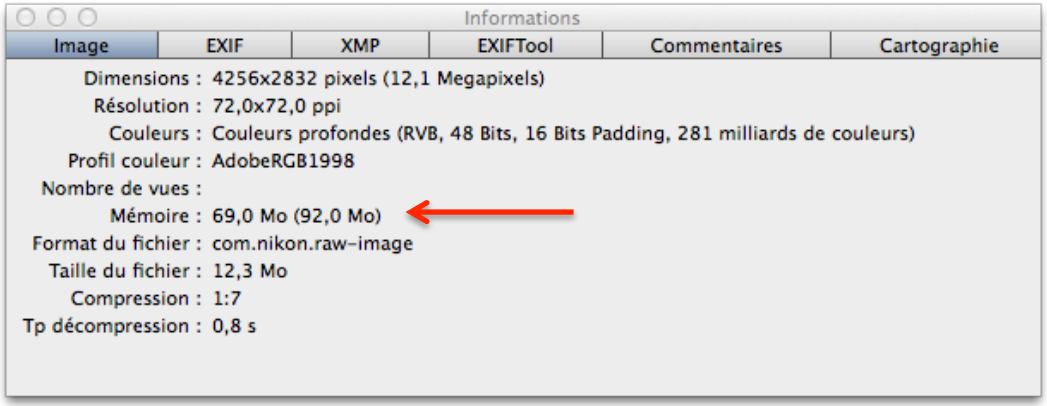
1.5 Taille d'une image en mémoire et sur disque.

Il est donc facile de calculer la taille de l'image. Comme c'est une matrice, il suffit de faire :

Taille de l'image = Largeur * hauteur * nombre d'octets par pixel.

Sur le D700 par exemple, les images sortent en 4 canaux de 14 bits mais sont enregistrées en 4 canaux de 16 bits (c'est plus simple à générer le fichier).

Ce qui fait 4256 * 2832 * 8 octets = 96 423 936 octets soit 92 Mo en RVBV (69 en RVB).



Le D800 fait 36,1 mega pixel, ce qui fait, avec 8 octets par pixels, 290 Mo la photo. Gloups !

Heureusement on sait compresser les images !

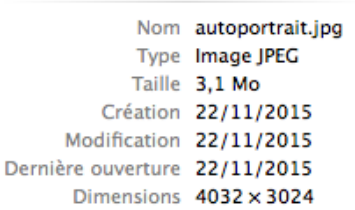
Mon image …

4032 pixels * 3024 pixels * 3 couleurs * 1 octets/couleur = 36 578 304 octets

soit … 34,9 Mo

En mémoire vive dans photoshop :

34,9 Mo… on retrouve bien le poids réel de l'image !



Sur le disque : 3,1 Mo (en jpg). Le fichier sur disque est donc **compressé**. On gagne ici un rapport 11 !

2 Compression sans perte

2.1 Le plus ancien exemple connu de compression sans perte :

A ·–	J ·–––	S ···	1 ·————
B –···	K –·–	T –	2 ··–––
C –·–·	L ·–··	U ··–	3 ···––
D –··	M ––	V ···–	4 ····–
E ·	N –·	W ·––	5 ·····
F ··–·	O –––	X –··–	6 –····
G ––·	P ·––·	Y –·––	7 ––···
H ····	Q ––·–	Z ––··	8 –––··
I ··	R ·–·	0 —————	9 ————·

Le code Morse.

Samuel Morse avait « senti » sans donner d'explication, que plus un message élémentaire (un caractère) revient souvent, plus on a intérêt à ce qu'il soit court… En anglais, le E, le T sont très fréquents, le O, le J, le Q plus rare. On note que tous les chiffres ont la même fréquence…

La *compression statistique* est née.

Ensuite, on peut se dire : tiens tiens, il y a des suite de caractères qui reviennent souvent ! Comme par exemple le « en » en français. Il y a aussi des mots, voir des suites de mots qui reviennent souvent. Si on mettait un code unique pour ces suites de caractères ?

Et voilà qu'on a grandement optimisé notre compression !

Ça s'appelle la compression LZ, utilisée dans les fichiers « ZIP » en informatique.

Pour compresser les dialogues de Tintin, il faut des codes cours pour « Saperlipopette », « Moulins à gaufres », etc… Le gain est d'autant plus intéressant que le mot est long et qu'il revient souvent.

2.2 Dans une image

Dans une image, le message élémentaire est le pixel, dont la valeur est sa couleur. Pour une image destinée à être affichée, il y a 256 niveaux différents pour chaque couleur (R, V, B). On commence donc par compter l'occurrence de chacun des niveau, et on fait comme le code morse : le niveau de couleur qui revient le plus souvent est celui qui prendra le moins de place. Pareil pour des séquences de pixels qui se suivent.

Forcément après ça, on a plus 3 octets par pixel. Il y a des pixels qui font moins de 3 octets, et des pixels qui font plus que 3 octets. Mais à la fin, on gagne :

Exemple d'une image au format BMP et ensuite compressée sans perte au format ZIP :

mon image…

Nom	autoportrait.bmp
Type	Image bitmap W…
Taille	36,6 Mo
Création	21/11/2015
Modification	aujourd'hui 00:04
Dernière ouverture	aujourd'hui 00:04
Dimensions	4032 × 3024

Nom	autoportrait.bmp.zip
Type	Archive dans un…
Taille	29,9 Mo
Création	aujourd'hui 00:05
Modification	aujourd'hui 00:05
Dernière ouverture	aujourd'hui 00:05

… mais pas beaucoup…

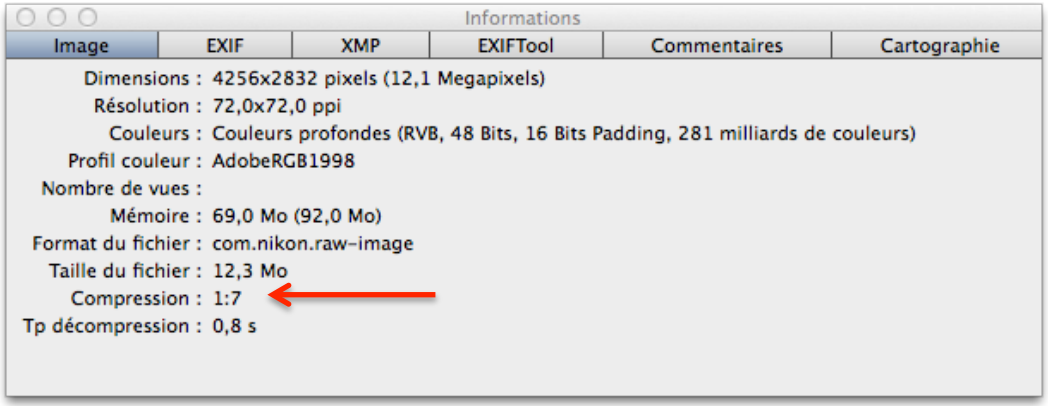
Pourquoi ?

Parce qu'il y a beaucoup de couleurs. Du coup, chaque couleur revient peu de fois, et donc les codes des couleurs grandissent trop vite. Et il n'y a presque jamais de séquences qui se retrouvent, puisque les pixels d'une même couleur apparente ne sont en fait pas les mêmes.

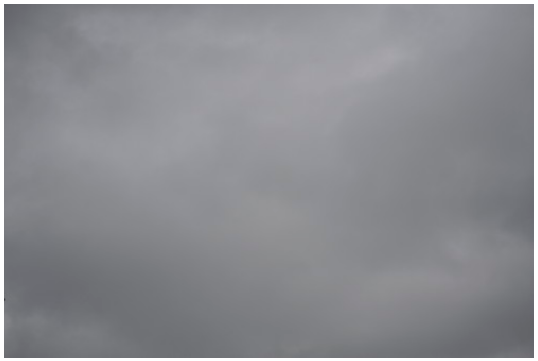
Mais dans tous les cas, ça ne vaut vraiment pas le coup de s'en priver puisque **la compression est sans perte**. C'est ainsi que les formats « non compressés » proposent en général une compression sans perte.

2.3 Le format RAW

C’est ainsi que le format RAW peut, ou non, être compressé.

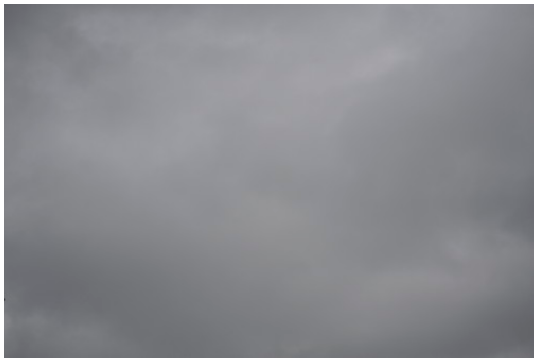


Exemple : le D700 propose le format RAW compressé sans perte. L’image générée par le capteur pèse toujours 92 Mo (ou 69 en RVB) mais elle est compressée sans perte à 1 : 7 ce qui fait que le fichier ne pèse « que » 12,3 Mo (exemple ci-dessus).



.NEF non compressé :

25 Mo. Comme l’image en mémoire pèse 96 Mo, c’est forcément déjà compressé !



.NEF compressé sans perte :

13,1 Mo

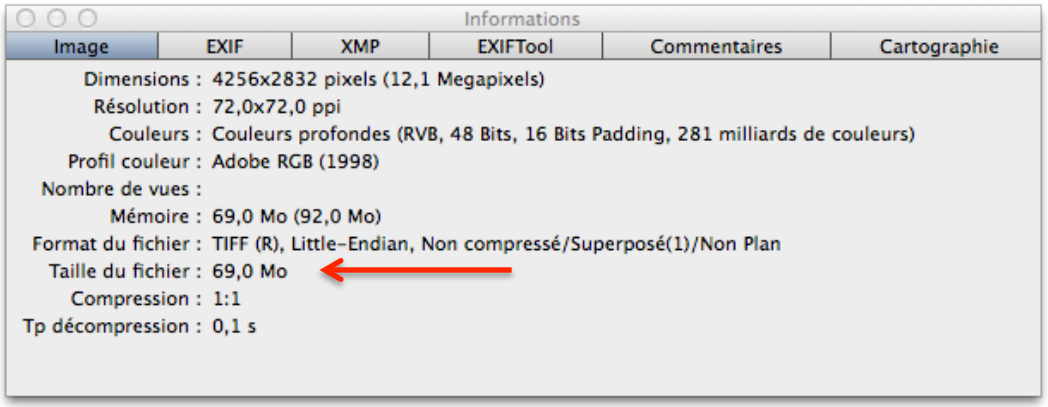
Attention les apparences sont trompeuses : plus de détail, et pourtant…

12,6 Mo

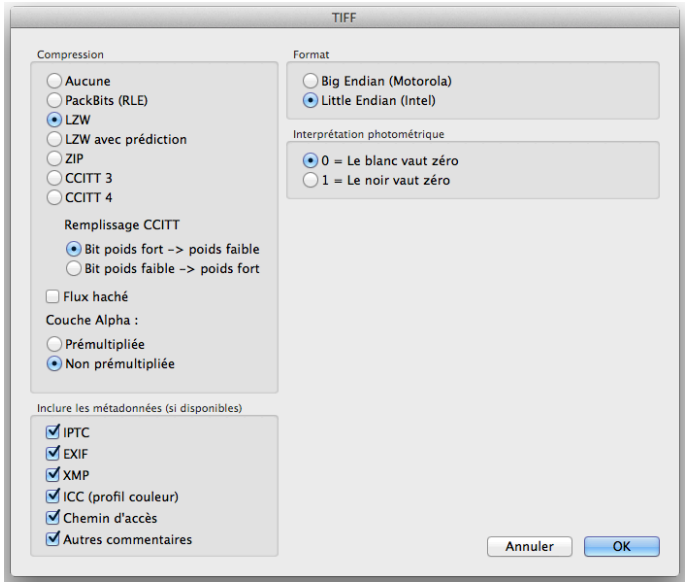
2.4 Le format TIFF

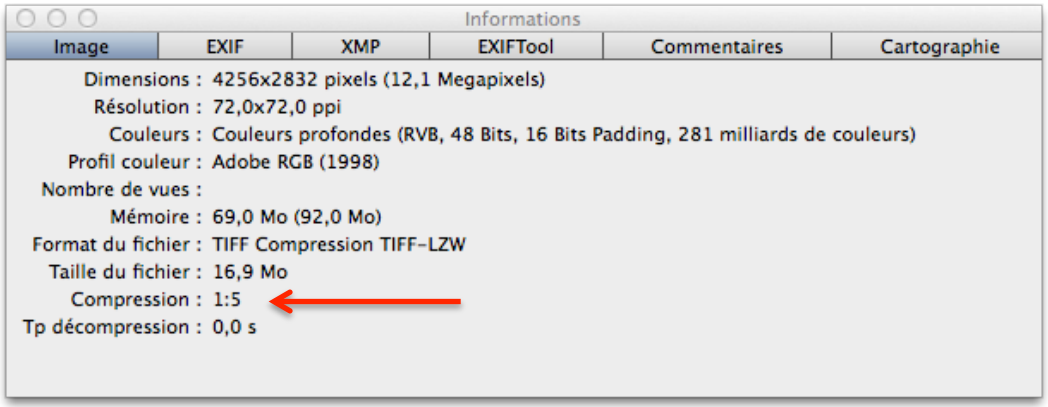
Le format Tiff est le seul format universel, non propriétaire et qui permet de stocker des images avec plus de 8 bits par couleurs, et sans perte.

Si j'enregistre cette image au format TIFF :



Mais le format TIFF peut reconnaître nativement un grand nombre de compressions et d'optimisations différentes… sans perte. C'est donc un excellent fichier d'échange pour les photos qui doivent conserver leurs qualités équivalentes au RAW.





3 Compression avec algorithme de compression d'image « avec perte »

3.1 Principes de base

Le principe de base de la compression d'image est simple : on optimise l'image en perdant des informations, puis on applique une compression sans perte. Enfin, c'était vrai jusqu'au JPeG 2000…

L'optimisation de l'image peut se faire de plusieurs manières, manières qui sont fortement liées à la puissance de calcul des machines. C'est ainsi que les formats de compression ont évolué pour suivre les capacités des machines.

Quelques grands principes :

- plusieurs images en 256 couleurs sont plus faciles à compresser qu'une même image en 16 millions de couleurs, parce que la fréquence d'apparition d'un pixel est de 1/256 au lieu de 1/16 millions. On commence donc par séparer les couleurs, ce qui crée plusieurs images N&B.
- L'œil humain est plus sensible à la luminosité qu'aux différences de teinte et de saturation. Il est donc judicieux de ne pas travailler sur des images par composante de couleur, mais plutôt par composante TSL (teinte, saturation, luminosité). Les images composantes pour la teinte et la saturation peuvent être dégradées (profondeur, résolution) davantage. En divisant la résolution par 2, on gagne un facteur 4…

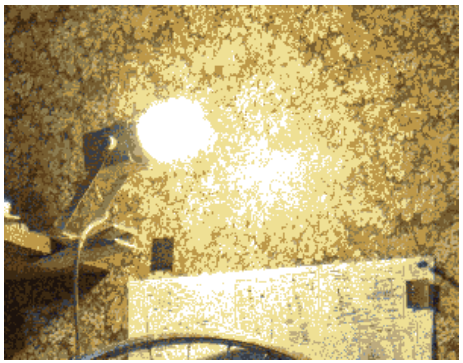
image au format TSL (teinte, saturation, luminosité)

Par ailleurs, Pour répondre au besoin de stockage d'informations autres que l'image elle même, la plupart des formats d'images peuvent contenir des informations autres que l'entête, qui est purement technique (taille de l'image, couleurs, etc…). Ce sont les zones EXIF, IPTC par exemple. Ces informations ne supportent pas la perte, elles sont donc stockées non compressées.

3.2 Le format GIF

Au début de l'ère internet, le format GIF a permis d'afficher des images sur des pages internet avec des modems 56k… sur des ordinateurs qui tournaient à 100MHz ou moins.

Le principe du GIF est très simple : on diminue le nombre de couleurs, et ensuite on applique une compression statistique. Il s'agit donc bien d'une **compression sans perte** ! En effet, la perte est faite au moment du choix des couleurs. Si l'image comporte peu de couleurs, alors elle n'est pas du tout dégradée. Le format GIF est donc très adapté pour les schémas, par exemple.

Fichier original non compressé (24 bits)	256 couleurs	16 couleurs
		
En BMP : 36,7 Mo	En BMP : 12,3 Mo	En BMP : 12,3 Mo
Impossible en GIF, limité à 8 bits	En GIF : 7,4 Mo	En GIF : 3,2 Mo

Les gros points forts du GIF sont :

- ne demande que peu de ressource au processeur (il y a 20 ans ça comptait !)
- permet des effets de transparence et des animations.
- Pas de perte pour des images avec peu de couleurs.

Et les gros défauts :

- limité en nombre de couleurs (forcément !)
- L'utilisation n'est pas libre (appartient à Unisys qui apprécie les royalties…)
- Taux de compression limitée dés qu'il y a du détail (limites de la compression LZ)
- Ne supporte pas les informations EXIF, IPTC…

Mais : en N&B 256 (niveaux de gris), ça peut donc être un choix qui se tient. Toujours la même photo pèse 6MO en JPG qualité maximum (et donc avec des pertes légères), et 10,5 Mo en GIF… **sans pertes**.

3.3 Le PNG

Le PNG a été créé pour contourner le problème de propriété du GIF. Il en reprend donc les grands principes, avec en plus :

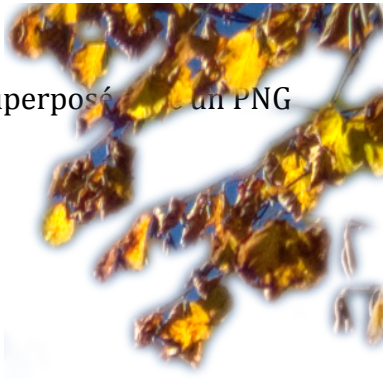
- Possibilité d'avoir des transparents fondus grâce à une 4ème couche de « couleur » dont l'intensité représente la transparence.
- Possibilité de dépasser 256 couleurs, mais la compression s'en ressent vivement. Donc possibilité d'avoir une image sans perte.
- Données EXIF prises en charges

Le gros défaut est le même que le GIF, à savoir :

- Taux de compression limitée dés qu'il y a du détail (limites de la compression LZ)

Exemples d'une image avec une couche de transparence par dessus une autre image :

Ici un exemple de texte superposé [illegible] un PNG



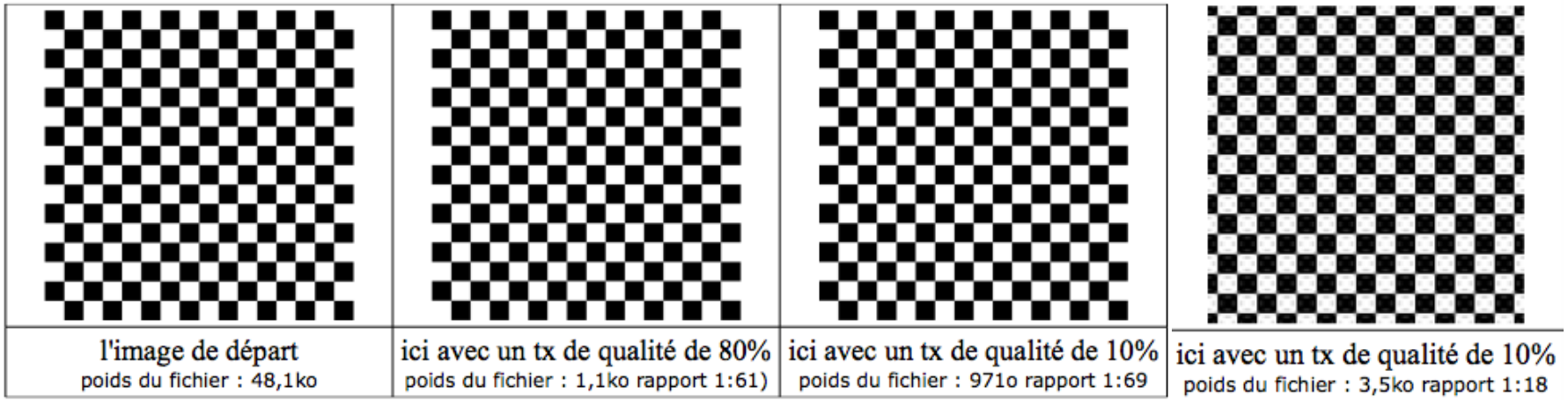
3.4 le JPeG (.jpg ou .jpeg)

Le format JPeG était initialement destiné à l'envoi de fax. Autrement dit, le JPeG, c'est un peu le moyen Age de la compression d'image.

Son principe est probablement le plus compliqué de toutes les compressions d'images :

- On découpe l'image en matrice de 8 x 8 pixels. Ceci est à l'origine de l'effet mosaïque.
- On fait une Transformée de Fourrier sur laquelle on applique un filtre passe bas (ce qui a pour conséquence de supprimer les variations brutales de contraste, à ce niveau se situe la perte). Le réglage de 1 à 12 de qualité du JPeG lors de l'enregistrement correspond à la valeur de ce filtre.
- On applique une compression statistique (sans perte)

Expérience amusante : si on crée une matrice de cases faisant chacune 8x8 pixels, on peut obtenir une compression maximum avec une qualité « médiocre », sans que l'image soit dégradée. Mais si on décale les carrés, c'est le drame :



l'image de départ poids du fichier : 48,1ko	ici avec un tx de qualité de 80% poids du fichier : 1,1ko rapport 1:61)	ici avec un tx de qualité de 10% poids du fichier : 971o rapport 1:69	ici avec un tx de qualité de 10% poids du fichier : 3,5ko rapport 1:18

Les gros points forts du JPeG sont :

- ultra répandu
- permet une bonne compression avec une bonne qualité d'image (petite taille de fichier)

Et les défauts :

- effet mosaïque et flou sur les transitions brutales
- pas de possibilité de compression sans perte.

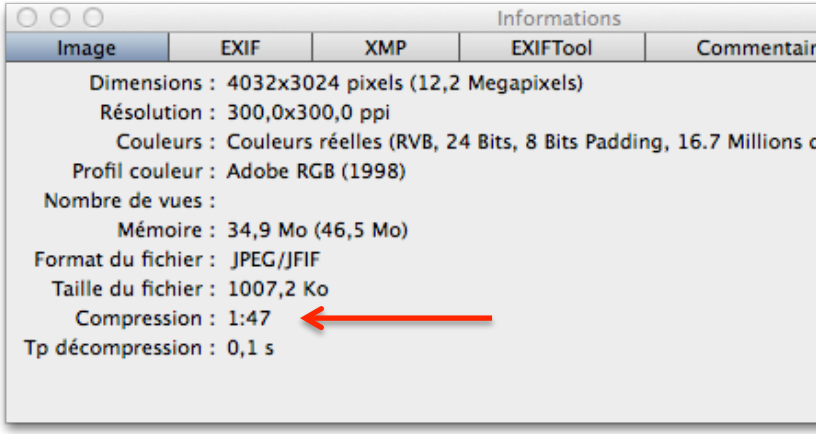
Voici un magnifique paysage (normal, c'est moi qui l'ai pris) :

Sa taille est 4032 * 3024 pixels, 24 bits, soit un bon gros 35 Mo en mémoire RAM. Je fais mon site Internet, et je veux que mon image fasse 1Mo… Observons un échantillon (ici brut de fonderie, à la compression de WORD près)

Compressé pour un atteindre de 1Mo en JPeG, ça fait ça : une belle mosaïque de bleus, et cet effet de « halo » autour des transitions brutales.

L'effet mosaïque, c'est le matriçage en 8x8 pixels. Le halo, c'est le passe bas. Trop bien le JPeG, y'a tout ce qu'on aime pas.

(chaque petit carré fait bien 8x8 pixel)



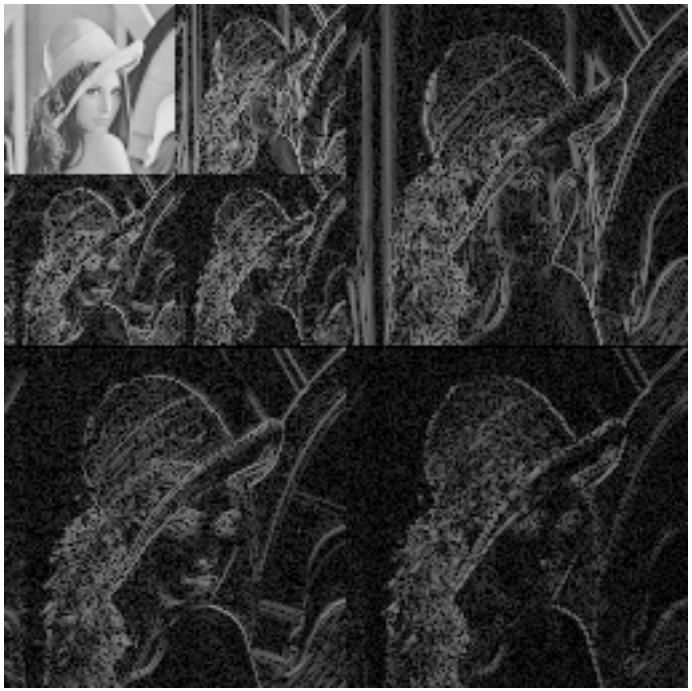
3.5 JPeG 2000 (ou JP2)

Par exemple utilisé sur le Geoportail

Il n'est pas reconnu sous Windows donc n'a jamais été largement diffusé. Il est cependant possible d'ouvrir des fichiers JPeG 2000 sous Windows avec des utilitaires. Gimp ne le prend pas en charge, XNView non plus. Il est nativement reconnu sur Mac depuis sa création dans les années 90… QuickTime le prend en charge, et donc tous les logiciels qui utilisent QuickTime pour ouvrir des images (Word sur Mac par exemple pour écrire ce rapport).

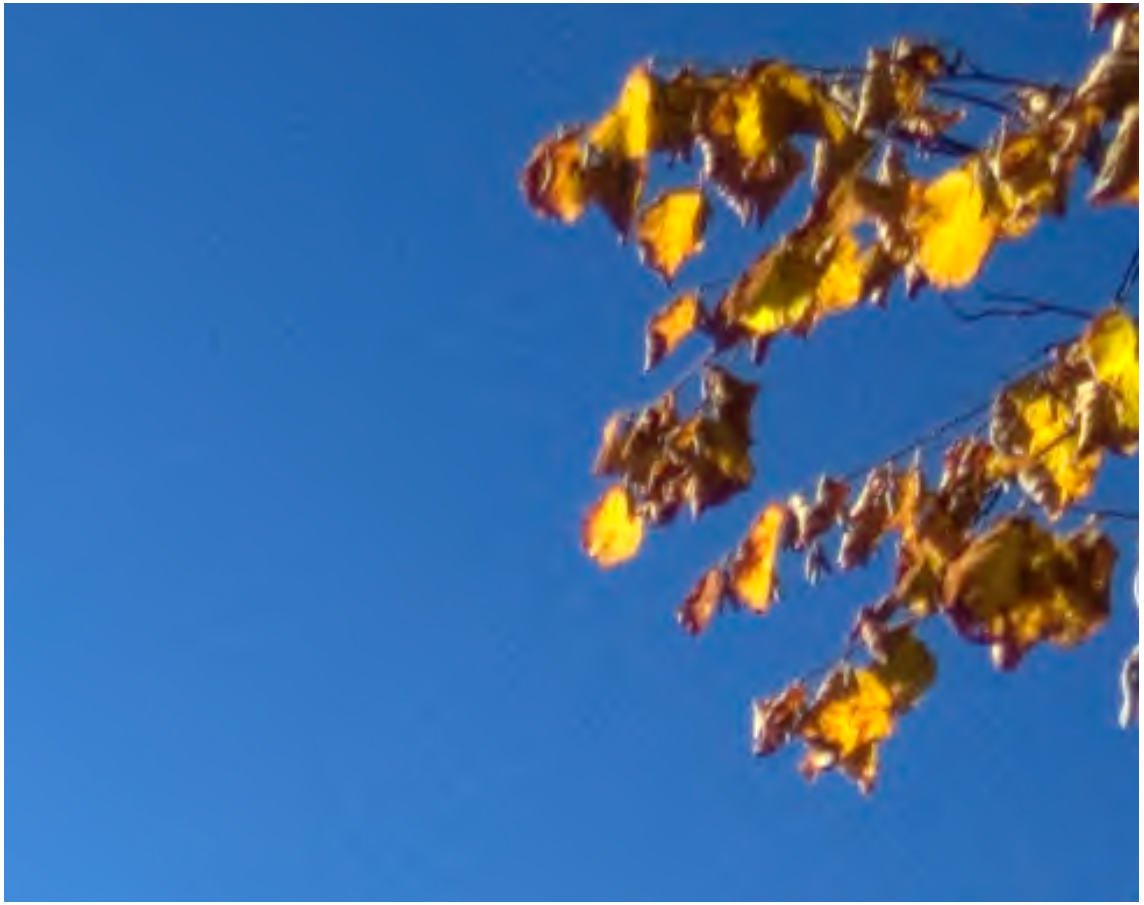
Le principe de compression est :

- Dans un premier temps le même principe que JPeG… séparation des composantes TSL, optimisation.
- Ensuite, contrairement aux modèles archaïques de compression d'image ;-), on utilise une compression par ondelettes qui consiste « vu de loin » à découper les 3 images résultantes en « tuiles » que l'on découpe en les réduisant successivement (image ci dessous)
- Une petite compression LZ pour finir.



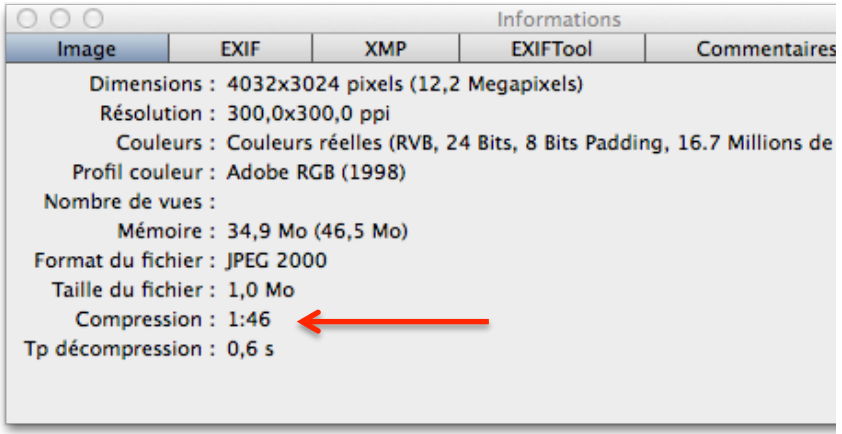
L'image originale et sa compression en « tuile »

(image piquée sur internet)



Objectif : avec la même image, atteindre un poids de 1Mo.

Le fichier fait rigoureusement le même poids sur le disque.



(on observe par ailleurs que le temps de décompression est 6 fois plus long !)

Les points forts du JPeG 2000 sont :

- Pour un rendu équivalent, un fichier de poids plus faible que le JPeG.
- La compression par ondelette offre également la possibilité d'une compression sans perte, plus performante que la compression LZ.

Et le défaut :

- Non reconnu sur les systèmes Window

4 Conclusion

4.1 Quelques comparaisons de compressions

	Format TIFF 8 bits / couleurs : 36,6 Mo Format TIFF 8 bits / couleurs, avec compression sans perte : 28Mo Format JPeG 2000 sans perte : 17,9 Mo Format JPeG 2000 sans perte : avec qualité 85 % : 3,9 Mo Format JPeG avec perte : avec qualité 85 % : 3,5 Mo (mais la qualité sera moins bonne) Format PNG : 31 Mo. Impossible de compresser ce fichier en GIF en conservant au moins du 24 bits.

(Ajouter d'autres images)

4.2 Et la vidéo :

Les flux analogiques vidéo « terrestre » à l'ancienne étaient codés en 8 bits par couleurs, avec 25 images par seconde et 720 × 576 pixels.

Les DVD sont (étaient ?) codés en MPG-2, ce qui correspond à un algorithme JPeG adapté dans le temps.

Aujourd'hui, tout le monde regarde des DIV-X, des MPG-4, ce qui correspond au principe de compression du JPeG 2000 ramené à la vidéo… la résolution est de l'ordre de 1920 x 1080 avec 50 ou 100 images par secondes… et ça passe par les mêmes canaux de diffusion.

Les photos numériques n'ont pas suivi le même parcours… C'est comme si on était resté au DVD.

4.3 Vraie conclusion

- Il ne faut pas dire « Le Raw c'est pas compressé » ;-)
- Le TIFF est un excellent format, pas assez utilisé.
- Qu'est qu'on attend pour sortir les photos au format JPeG 2000 ? Et bien… que Microsoft accepte de reconnaître ce format…

J'ai essayé de ne pas faire trop de calculs, de tout expliquer avec des mots. J'espère que ça n'a pas été trop compliqué !

Enjoy !

Jonathan