Anto59290

Zeste de savoir

La compression JPEG

12 août 2019

Table des matières

Dans cet article nous allons découvrir ensemble les différentes étapes de la compression JPEG. C'est un format d'image que nous utilisons tous les jours et connaitre son mode de fonctionnement permet de mieux comprendre ses avantages et ses défauts. Sachez que le JPEG dispose d'un tas d'options qui permettent d'obtenir des résultats plus ou moins efficaces en fonctions de l'image. Nous allons nous placer ici dans le cas le plus simple en ignorant ces options. Il existe très peu de logiciels proposant l'ensemble des options et beaucoup se contentent de l'algorithme "classique".

Cet article utilise des notions de bases d'informatique et des notions avancées de mathématiques (fonctions trigonométriques, matrices, transformées, etc.) . Il est possible de comprendre les principes de la compression JPEG sans en comprendre les formules, alors lancez vous !

FIGURE 0. – Représentation des différentes qualités de compression

1. Vocabulaire

Afin de permettre une meilleure compréhension de l'article il me faut définir le taux de compression. Celui-ci est le rapport de poids du fichier non compressé sur le poids du fichier compressé. Ainsi un fichier natif de 12 Mio compressé en un fichier de 3 Mio correspond à un taux de

compression de 4 :1. Le fichier compressé est à un poids 4 fois plus faible que le fichier natif. Un taux de compression élevé est souvent recherché, sans que cela ne soit au détriment de la qualité.

Les données informatiques sont généralement stockées de façon binaire : à l'aide de bit valant 0 ou 1. Ces bits sont souvent regroupés par multiples de 8.

Train de Bits	Valeur décimale
00000001	1
00000010	2
00000011	3
00000100	4
...	...
11111101	253
11111110	254
11111111	255

On peut voir que quand un entier est codé à l'aide de 8 bits sa plage de représentation est de 0 ;255. Si l'on souhaite manipuler des nombres plus grands ou des nombres décimaux il est généralement nécessaire d'augmenter la profondeur d'encodage, jusqu'à 16 bits par exemple. Ainsi un entier codé sur 16 bits pourras être compris entre 0 et 65 535.

2. Image et prérequis

Afin de mieux comprendre la compression JPEG nous aurons besoin d'une image. Plaçons-nous dans le cas le plus courant d'une image couleur, celle-ci possède trois canaux : le rouge, le vert et le bleu et chaque canal de chaque pixel est codé sur 8 bits. Ainsi un pixel est composé de trois valeurs entières (R ; V ; B) chacun comprises entre 0 et 255. Par exemple un pixel rouge sera représenté par les valeurs (255 ; 0 ; 0) et le triplet (255 ; 255 ; 0) sera jaune, c'est à dire un mélange de rouge et de vert.

Malheureusement pour nous le format JPEG ne travaille pas en RVB mais dans un autre espace de couleur : YC_bC_r. Dans cet espace la luminance Y (la "luminosité") d'un pixel est séparée de la couleur. Cette dernière est représentée grâce à deux canaux C_b (la différence au bleu) et C_r (la différence au rouge).

? Heu, moi ça m'allait bien le RVB, pourquoi ils n'ont pas fait comme tout le monde ?

Il y a en fait de nombreuses raisons a l'utilisation de YC_bC_r dans la compression JPEG. Premièrement il s'agit d'un espace de travail historique qui fait fonctionner la télévision française depuis ses débuts. Ce choix permet donc une compatibilité des systèmes images très simple. De

plus en travaillant dans cet espace il est très simple de passer d'une image en nuances de gris à une image couleur (comme cela a été le cas pour la télévision).

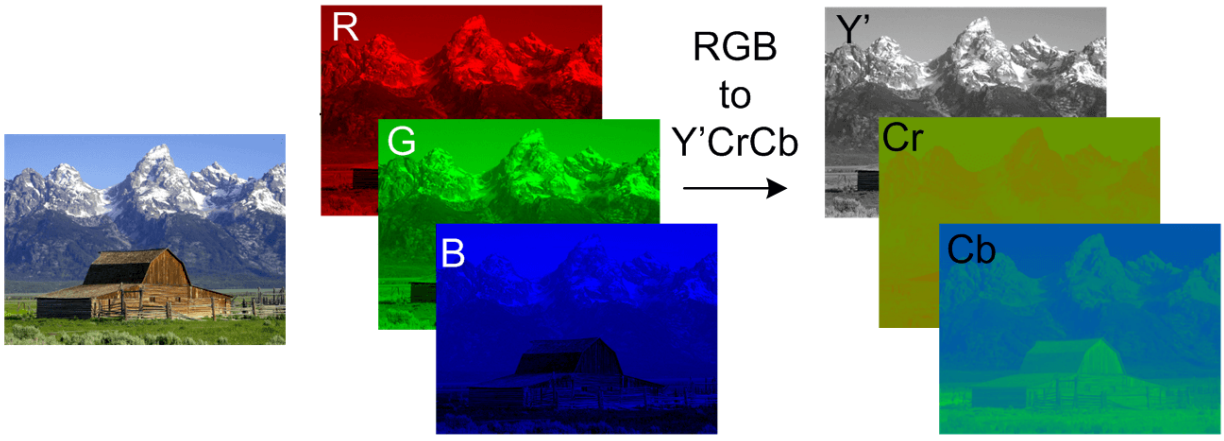

FIGURE 2. – Illustration d'un changement d'espace de couleur pour une image. Image créé à partir [du domaine public](https ://commons.wikimedia.org/wiki/File :CCD.png)

Ensuite ce choix correspond à des raisons physiologiques, le corps humain est bon pour capter des différences de luminosité, mais mauvais pour capter les différences de couleurs. Séparer ces informations va donc nous permettre de leur appliquer une compression différente, destinée à la vision humaine. Enfin le but de l'opération est aussi décoreller les informations. En effet les signaux R, V et B auront une évolution similaire dans l'image, on dit qu'ils sont fortement corrélés. En revanche la luminance (Y) et les signaux de chrominance (C_b et C_r) n'ont pas beaucoup de rapport entre eux, on dit qu'ils sont décorellés. Ce phénomène de décorrélation va nous permettre d'obtenir un traitement plus efficace avec moins de répétitions dans l'image compressée.

Pour ces raisons l'espace YC_bC_r est très souvent utilisé en compression d'image. La première étape de la compression JPEG est le passage de l'espace RVB à l'espace YC_bC_r, cela se fait très simplement en appliquant une petite formule de changement d'espace colorimétrique pixel par pixel.

$$Y = \left[\frac{R + 2V + B}{4}\right]$$

$$C_b = B - V$$

$$C_r = R - V$$

2.1. Sous-échantillonnage

Maintenant que l'on s'est amusé à séparer les couleurs de la luminance, nous allons pouvoir commencer a les traiter séparément. Si nous voulons obtenir un taux de compression plus élevé, le JPEG nous donne la possibilité de supprimer des informations de couleur. On ne parle pas ici de passer l'image en noir et blanc mais de réduire la répétition entre les informations de couleurs. En effet, notre système visuel est beaucoup plus sensible aux variations de lumière qu'à celles de

couleur, on peut donc y voir l'occasion de supprimer quelques infos *superflues* au passage. Cette opération s'appelle le sous-échantillonnage de chrominance et correspond à une diminution de la résolution spatiale, c'est-à-dire à la diminution de la quantité d'information sur une même surface. Pour réduire la quantité d'informations nous allons supprimer les paires (C_b ;C_r) de certains pixels de l'image. Pour ce faire il faut choisir un mode de sous-échantillonage, cela nous permet de savoir les informations nous allons garder et celles que nous allons supprimer. Cette étape est optionnelle dans le processus de compression JPEG.

Dans la norme JPEG il existe trois modes de sous échantillonnage, le mode 4 :2 :0, le 4 :2 :2 et le 4 :4 :4. Le premier nombre représente la largeur d'un bloc pour le traitement de la chrominance. En JPEG cette largeur est toujours de quatre. Le deuxième nombre représente le nombre d'échantillons de chrominance dans la première ligne et le deuxième nombre dans la deuxième ligne.

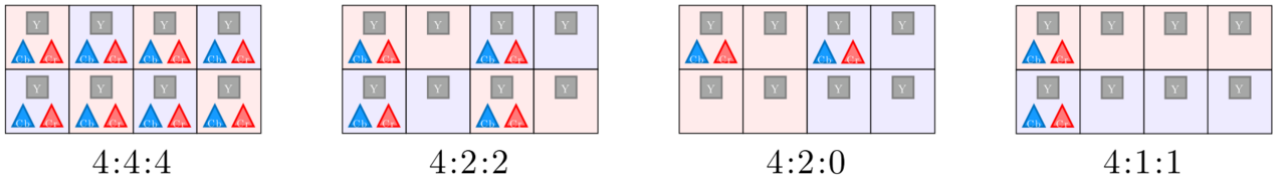

FIGURE 2. – Différents modes de sous échantillonnage de la chrominance - Ellande , Licence CC BY SA 4.0

Si l'on regarde ce schéma on commence par le mode 4 :4 :4 qui correspond à notre image d'origine sans altération. Si l'on regarde les quatre colonnes et les deux lignes, on trouve dans chaque case une information de luminance (Y) et deux informations de chrominances (C_r et C_b). Toute les informations sont bien là, l'image d'origine n'a pas été modifiée, le résultat sera visuellement meilleur, mais le taux de compression moins bon.

Dans le mode 4 :2 :2 nous avons bien toutes les informations de luminance mais on conserve seulement deux informations de C_r et deux informations de C_b par ligne. D'où le nom 4 :2 :2. Les pixels *sans informations* sont affichées lors du décodage avec une moyenne de la valeur de gauche et de droite. On perd en finesse de couleur, mais le taux de compression grimpe en flèche.

Dans le mode 4 :2 :0, on retire toute la chrominance de la deuxième ligne. Pour retrouver des informations approximatives de chrominance on procédera par moyenne lors du décodage. On a bien fait le ménage dans les informations, mais la qualité visuelle peut en pâtir.

i

Le mode 4 :1 :1 du schéma n'est jamais utilisé dans la norme JPEG. Sachez qu'il existe plein d'autres modes de sous échantillonnage comme le 4 :2 :1, le 3 :1 :1 ou des modes exotiques comme le 3 :1,5 :0 !

À partir de ce point, nous ne travaillons plus que sur un seul canal (la luminance). Le traitement appliqué est le même quelque soit le canal, mais ils sont traités indépendamment.

2.2. Découpage en blocs

L'opération suivante consiste à découper notre image en blocs de 8x8 pixels. Ces blocs forment des matrices qui seront traitées séparément dans la suite de la compression. L'avantage de cette technique est que l'on travaille sur des matrices assez petites (64 valeurs) ce qui permet un traitement rapide. Le nombre de matrices à traiter est en revanche assez important.

Les plus malins d'entre vous se demandent déjà comment l'algorithme procède quand l'image possède des dimensions qui ne sont pas des multiples de 8. Dans ce cas il faut ajouter des pixels supplémentaires pour obtenir les dimensions voulues. On ajoute au choix des pixels noirs ou on réalise une recopie des pixels existants. Cette étape génère parfois des petits défauts de compressions sur le pourtour de l'image (on parle d'artefacts de compression). La technique de recopie des pixels tends a diminuer ces artefacts mais demande un peu plus de temps de calcul.

La phase de préparation de l'image est terminée, nous allons maintenant travailler sur des blocs de 8x8 pixels. Jusqu'ici nous n'avons pas vraiment altéré notre image, sauf à l'étape de sous-échantillonage (en dehors du cas 4 :4 :4), il s'agit de la préparation à la suite de la compression qui elle se passe dans le domaine fréquentiel .

3. Le domaine fréquentiel

3.1. La transformée

Nous travaillons maintenant sur un bloc de 8x8 pixels, si on ne considère uniquement la luminance (Y) ce bloc pourrait être le suivant.

139144149153155155155155144151153159156156156156150155160163158156156156159161162160160159159

Comme vu dans la partie précédente les informations sont codée sur 8 bits par canal, elles peuvent donc être contenues entre un intervalle de 0 à 255 ($2^8 - 1$). Ces valeurs représentent la luminosité des différents pixels dans un bloc de l'image.

![L'image représentant le bloc de luminance que nous allons traiter (agrandi). On voit que l'on est sur une toute petite partie d'une image. On peut remarquer que les variations sont assez faibles.](/media/galleries/3582/f9abf915-aff3-4358-aab6-db564a6cc790.png)

Pour la suite du traitement nous souhaitons travailler dans l'espace fréquentiel. Cet espace va nous permettre d'isoler les fréquences les plus utiles pour le cerveau humain (les basses fréquences) des fréquences auxquelles nous sommes peu sensible (hautes fréquences). Pour cela nous disposons d'un super outil mathématiques : la transformée en cosinus discrète. :magicien :

$DCT_{u,v}$ est la valeur du coefficient à la position u, v dans la matrice de fréquence.

$Pixels_{x,y}$ est la valeur de la luminance du pixel à la position x, y dans la matrice de luminance.

Bon, ok quelques explications s'imposent. Nous souhaitons passer nos données du domaine spatial au domaine fréquentiel. De cette façon les valeurs de la matrice ne seront plus l'intensité des pixels mais l'intensité des fréquences. L'outil mathématiques qui nous permet cela est la transformée en cosinus discrète. Cette opération est de l'ordre des transformées, **elle ne modifie donc pas les données mais seulement la manière de les représenter.** On pourrait ainsi imaginer une transformée qui permet de passer de l'écriture numérique à l'écriture toute lettre. On transformerait ainsi "10" en "dix". Les données ne sont pas modifiées, seulement la façon de représenter.

Ce type de transformée part du principe que tout signal discret peut être écrit sous la forme d'une somme de cosinus. Ainsi si on considère un signal à une dimension on obtient la formule suivante.

$$DCT_u = \frac{1}{2} \sum_{x=0}^{7} Pixels_x \cos\left[\frac{(2x+1)u\pi}{16}\right]$$

Avec cette formule le calcul de DCT est assez complexe, dans notre exemple, le résultat de la transformée est le suivant.

1260−1−12−52−2−31−23−17−6−3−300−1−11−9−220−1−10−7−2011000−1−1120−1112020−111−1-

Bien que cette matrice soit similaire à la précédente et qu'elle représente les même données (rappelez-vous nous n'avons fait que transformer) elle présente une différence fondamentale : elle se trouve dans l'espace des fréquences. Ainsi les coefficients représentent l'amplitude de chaque fréquence et non plus l'amplitude lumineuse de chaque pixel.

On peut remarquer plusieurs choses de cet exemple :

- La première valeur de la matrice est la plus élevée. On l'appelle composante DC ou composante continue. Elle représente le signal continu du bloc de pixel, dans notre cas la moyenne de luminosité. - Les plus grosses valeurs sont regroupées dans le coin haut gauche de la matrice. Il s'agit des coefficients de faibles fréquences, qui représentent les zones faiblement détaillées de notre image (les aplats). En effet une image naturelle (une photo par exemple) comporte généralement beaucoup d'aplats locaux et peu de bordures très contrastés. Gardez en tête que l'on travaille sur une toute petite portion de l'image (un bloc de 8x8). - Les coefficients ont une valeur qui peut dépasser 255 ou être négatifs, il faut donc passer d'un encodage sur 8 bits à une profondeur d'encodage plus élevée (16 bits en général). - Comme dit précédemment cette opération est en théorie non destructrice, nos données pourraient être parfaitement intactes. Mais le calcul informatique avec des virgules flottantes se fait à une précision finie, ce qui introduit de nombreuses approximations.

[[information]] | En pratique la DCT est peu utilisée sous cette forme et elle remplacée par la *Fast Fourrier Transform* (ou transformée de fourrier rapide). Cette autre transformée est

optimisée pour les blocs dont les dimensions sont des puissances de deux, comme les blocs 8x8. En fonction des conditions cet algorithme peut être jusque cent fois plus rapide.

La quantification

Le but de cette étape est d'adapter la compression de l'image à la vision humaine. En effet notre système visuel est bon pour détecter des variations de luminosité sur une grande surface. En revanche il est médiocre pour détecter précisément des changements importants de luminosité sur une petite surface.

Mathématiquement, nous allons diviser chaque coefficient de la matrice par une constante. Ces constantes sont elles même stockées dans une matrice nommée la matrice de quantification. L'œil est plus sensible aux basses fréquences, qui sont stockées dans le coin haut gauche de la matrice de coefficients de la DCT. Les coefficients de la matrice de quantification sont donc plus faibles dans cette zone afin de mieux conserver les informations importantes pour notre système visuel. Il n'existe pas de formule générale pour calculer la matrice de quantification, car celle-ci est laissée à l'appréciation de l'auteur de l'encodeur. Ainsi Adobe Photoshop utilise peut-être un algorithme pour générer cette matrice différent de Gimp.

[[information]] | Afin de simplifier l'algorithme il est courant que les mêmes tables de quantifications soient souvent réutilisées. Le résultat obtenu est légèrement moins bon, mais la compression est plus rapide car il ne faut pas recalculer de table, il est juste nécessaire de l'adapter à la qualité visuelle souhaitée.

On peut prendre par exemple la matrice de quantification Q suivante.

16&11&10&16&24&40&51&61 \ 12&12&14&19&26&58&60&55 \ 14&13&16&24&40&57&69&56
\ 14&17&22&29&51&87&80&62 \ 18&22&37&56&68&109&103&77 \ 24&35&55&64&81&104&113&92
\ 49&64&78&87&103&121&120&101 \ 72&92&95&98&112&100&103&99 \end{pmatrix} $$

On remarque que les coefficients sont beaucoup plus élevés en bas à droite de la matrice, il s'agit de la zone de fréquence où l'œil est le moins sensible. La division des coefficients de la DCT par les coefficients de la matrice de quantification va produire des nombres réels, que nous allons arrondir afin de ne travailler qu'avec des entiers. Cette étape est destructive est fait perdre des informations de manière non réversible. C'est à cette étape qu'intervient la plus grosse perte de données dans le processus de compression JPEG.

On fait un exemple avec le premier coefficient de la matrice :

$$\text{round}\left(\frac{1260}{16}\right) = \text{round}\,(78,75) = 79$$

Le résultat des divisions est le suivant.

$$790 - 100000 - 2 - 1000000 - 1 - 10000000000000000000000000000000000000000000000$$

On peut constater que :

- De nombreux coefficients ont une valeur nulle. Si on regarde la moitié inférieure droite des matrices on peut voir que les coefficients DCT étaient assez petits et qu'ils ont été divisé par des coefficients de quantifications assez élevés. L'arrondi nous donne donc une valeur nulle. - La valeur la plus élevée dans la matrice est relativement faible, on peut donc de nouveau stocker les coefficients avec une profondeur de 8 bits.

Codage

Lecture du bloc

A ce point de la compression nous avons réussi a générer beaucoup de zéros dans le bloc, l'information à coder est beaucoup plus redondante, ce qui va nous permettre d'obtenir un codage plus efficace. Nous allons voir comment tirer parti de cette redondance.

La première étape est de parcourir le bloc en zig-zag et de stocker les informations dans ce nouvel ordre. Le but de cette manœuvre est de regrouper un maximum de coefficients nuls ensemble afin d'améliorer le taux de compression.

![Le parcours du bloc 8x8 en zig-zag - Domaine Public](https://upload.wikimedia.org/wikipedia/commons/th
$JPEG_ZigZag.svg.png$)

Rappelez-vous le bloc à traiter :

79&0&-1&0&0&0&0&0 \ -2&-1&0&0&0&0&0&0 \ -1&-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
\end{pmatrix} $$

Vous voyez l'idée ? En parcourant le bloc de cette façon on condense les informations intéressantes. Voici la partie non-nulle du bloc affichée en ligne avec et sans le zig-zag,

$$Normal = \begin{pmatrix} 79 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & & \end{pmatrix}$$

$$Zig - Zag = \begin{pmatrix} 79 & 0 & -2 & -1 & -1 & -1 & 0 & 0 & -1 \end{pmatrix}$$

Lisez chaque matrice à haute voix, vous pouvez vous rendre compte que la seconde solution va beaucoup plus vite ! Vous vous demandez surement ce qui advient de tous les zéros qui sont derrière. Pour signaler la fin du bloc utile (la fin de la partie non-nulle) on utilise un marqueur spécial le EOB (*End of block*). Quand le décodeur rencontre ce marqueur il faut remplir le reste du bloc avec des zéros.

3.2. Codage de Huffman

La suite des opérations consiste a appliquer un codage d'Huffman. Ce codage attribut des "mots" (une séquence binaire) à chaque couple rencontré, un couple étant formé de la valeur d'un coefficient de la matrice R (lue en zig-zag) et du nombre de zéros précédants cette valeur. Voici quelques exemples de couples dans notre cas : (79 ;0) (-2 ;1) (-1 ;2).

Plus le couple est rencontré souvent, plus le mot attribué par le codage d'Huffman est court. Ainsi une information présente de nombreuse fois sera stockée avec très peu de données. Pour attribuer ces mots le codage se base sur des tables écrites dans la norme JPEG. Il est aussi possible de construire une table codage spécifique a une image, mais le gain d'espace est généralement assez faible.

Le codage et la compression sont terminées, on résume tout ça dans la conclusion.

Voila pour la compression d'une image en format JPEG. En réalité il reste à écrire le résultat dans un fichier respectant la norme. Cela permet à n'importe quel programme de lire le JPEG correctement. Le décodage de l'image JPEG suit le même algorithme en sens inverse et ne présente donc pas grand intérêt.

FIGURE 3. – Processus de compression et décompression au format JPEG - Domaine Public

Si on résume les différentes étapes on a donc :

- Changement d'espace de couleur
- Sous-échantillonnage de chrominance (optionnel)
- Découpage en bloc
- Utilisation de la DCT
- Quantification et arrondi
- Lecture en Zig-zag
- Codage de Huffman

On peut aussi préciser que la compression JPEG **dégrade systématiquement l'image**. Si on regarde de plus près cela se passe à trois différents niveaux :

- Le sous-échantillonnage de couleur (optionnel)
- La précision du calcul lors de la DCT
- L'arrondi de quantification

Ainsi même avec un algorithme très précis, l'arrondi provoquera une petite perte de données.

4. Taux de compression et résultat visuel

En général on parle d'un taux de compression obtenu de 10 :1 sans changements notables à l'œil nu. Bien sur ce résultat dépend du type d'image et des options choisies.

FIGURE 4. – Image compressé avec un taux de 143 :1 - ToyToy Licence CC BY-SA 3.0

La compression JPEG peut fournir des taux de compression beaucoup plus élevé, de l'ordre de 150 :1, mas avec une qualité visuelle très mauvaise. Le défaut majeur de l'algorithme est son utilisation de blocs de 8x8 non chevauchants. Cela est visible directement par la présence de ces blocs dès que le taux de compression devient élevé. D'autres algorithmes comme le JPEG 2000 ont cherchés a résoudre ce problème en n'utilisant pas le découpage en blocs et en remplaçant la DCT par une autre transformée.

5. Pour aller plus loin

Voici quelques sujets d'ouverture pour compléter la lecture :

— Se plonger dans l'algorithme MP3 qui utilise le même genre de procédés.
— Approfondir avec l'algorithme de JPEG 2000 (dites au revoir à la DCT) : en résumé (EN) ou en détails (EN)

Merci à backmachine, NuX, Aabu, A-312, amael, hobi1, blo yhg, motet-a et The-Aloha-Protocol pour la relecture et les conseils. Merci informaticienzero à pour la validation