Formats d'images

1 Introduction

Lorsque nous utilisons un ordinateur ou un smartphone l'écran constitue un élément principal de l'interaction avec la machine. Les images sont donc au cœur de l'utilisation de ces objets, et leur qualité, leur taille et les moyens de les manipuler constituent des éléments importants de l'interaction homme machine. Comme pour la plupart des objets numériques, il existe de nombreux formats de fichiers d'image. Les 5 formats les plus utilisés en informatique sont : `BMP`, `TIFF`, `GIF`, `JPEG` ou `JPG`, et `PNG`.

BMP (Windows BitMaP) : Ce format est l'un des premiers qui a été utilisé en informatique. Il est simple car chaque pixel est représenté par un caractère. Il ne compresse pas les images ce qui le rend très volumineux. Il n'est quasiment plus utilisé de nos jours.

TIFF (Tagged Image File Format) : Ce format est orienté vers les professionnels (imprimeurs, publicitaires...). Il permet d'obtenir une image de très bonne qualité, mais sa taille reste volumineuse, même si elle est inférieure à celle des fichiers `BMP`, car les fichiers `TIFF` peuvent être compressés (compression LZW).

GIF (Graphics Interchange Format) : Ce format est un des standards d'internet. Les fichiers sont de petite taille, car les images ne contiennent que 256 couleurs. Ce format permet d'avoir des parties d'image transparentes.

JPEG (Joint Photographic Expert Group) : Ce format offre plusieurs taux de compressions, qui affectent la qualité de l'image. Il est devenu le standard des formats d'image sur internet, car on obtient des images plus petites grâce à la compression. Ceci permet de les charger rapidement, même avec une connexion bas débit.

PNG : Portable Network Graphic C'est le futur standard pour internet, car il autorise la transparence comme le format `GIF` et autorise 16.7 millions de couleurs. Il permet aussi la compression de données.

On rencontre également le format de fichier `RAW` qui n'est pas un standard, mais qui désigne les fichiers créés par les appareils numériques réflexes ou bien les scanners. Ce format se caractérise par le fait qu'il est quasiment constitué des informations « brutes » captées par l'appareil.

Il est possible de classer les formats d'images en deux grandes familles :

- Les images dites « matricielles » (*BitMap*), peuvent représenter des photos, des dessins, des plans etc ... Ces images en couleur ou noir et blanc sont constituées de tous les points (*pixels*) de l'image. Agrandir ou rétrécir ces images pose certains problèmes afin de déterminer les pixels de la nouvelle image. Par contre, elles sont souvent d'une grande qualité car elles contiennent toute l'information.
- Les images dites « vectorielles », sont formées de courbes qui définissent des formes à l'aide d'équations. Même pour des formes complexes, les images vectorielles nécessitent moins de place en mémoire que les images BitMap. De plus elles supportent parfaitement le changement d'échelle.

La figure 1 montre une image vectorielle d'un abricot qui n'est pas détériorée par le changement d'échelle, alors que l'image matricielle du même objet ne passe pas correctement à l'échelle.

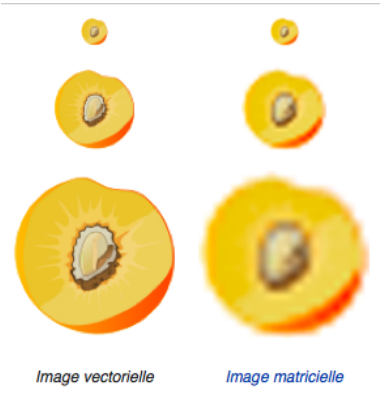


FIGURE 1 – À gauche une image vectorielle d'un abricot et à droite l'image matricielle du même objet.

2 Les images matricielles

Parmi les images représentées de manière matricielle, il y a des formats d'images compressés et des formats non compressés.

2.1 Les formats non compressés

Certains formats d'image comme les formats `BMP` (*Windows BitMaP*), `PPM` (*Portable PixMap file format*), `PGM` (*Portable GrayMap file format*) et `PBM` (*Portable BitMap file format*), n'effectuent pas de compression, c'est-à-dire que tous les pixels sont présents dans l'image. L'avantage de ces formats est la qualité des images car sans compression il n'y a pas de perte de qualité. Par contre cela donne des fichiers de grande taille. Par exemple, une image `BMP` de 800×600 pixels occupe 1,37 Mo.

Nous allons détailler les formats `PPM`, `PGM` et `PBM` car ils sont simples à expliquer. Par contre ils sont très volumineux puisque les pixels sont représentés par des caractères ASCII qui occupent 1 octet chacun.

Le format PBM a été défini par Jef Poskanzer dans les années 1980, pour des images monochromes codées en texte ASCII. Puis en 1988, il a développé les formats PGM (pour les images avec des niveaux de gris) et PPM (pour les images en couleurs). Ces fichiers sont construits sur la même base :

- Le nombre « *magique* » du format (P1 pour les images en noir et blanc, P2 pour celles en niveaux de gris et P3 pour les images en couleurs).
- Un caractère d'espacement qui peut être un espace, une tabulation ou une nouvelle ligne.
- La largeur de l'image (codée en caractères ASCII).
- Un caractère d'espacement qui peut être un espace, une tabulation ou une nouvelle ligne.
- La hauteur de l'image (codée en caractères ASCII).
- Pour les format PGM et PPM, un caractère d'espacement qui peut être un espace, une tabulation ou une nouvelle ligne, suivi d'un nombre représentant le nombre de couleurs utilisées.
- Un caractère d'espacement qui peut être un espace, une tabulation ou une nouvelle ligne.
- Les données binaires de l'image :
  - L'image est codée ligne par ligne en partant du haut.
  - Chaque ligne est codée de gauche à droite.

De plus, toutes les lignes commençant par # sont ignorées.

PBM : ces images utilisent le caractère 1 pour les pixels noirs et 0 pour les pixels blancs, comme l'indique l'image de la figure 2.

PGM : ces images sont constituées de caractères qui représentent des niveaux de gris dont la valeur maximale inférieure à 65536 (codée en caractères ASCII) est fixée dans l'image. Sur la figure 3, il y a 15 niveaux de gris possible. Le noir est codé par 0 et le blanc par la valeur maximale.

PPM : Cette fois les pixels sont colorés. Il faut indiquer pour chaque pixel 3 valeurs qui indiquent la proportion de rouge, de vert et de bleu. La valeur maximale des couleurs, inférieure à 65536 (codée en caractères ASCII), est fixée dans l'image, sur l'exemple de la figure 4 cette valeur vaut 255.

```
P1
#  Image de la lettre "J"
6 10
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
1 0 0 0 1 0
0 1 1 1 0 0
0 0 0 0 0 0
0 0 0 0 0 0
```

(a) Résultat (b) Code de l'image

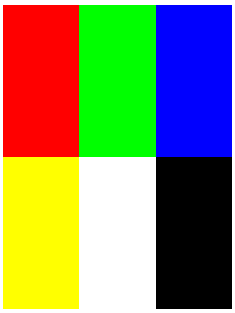
FIGURE 2 – Image PBM représentant la lettre « J » 2a et son code 2b

(a) Résultat

```
P2
# Affiche le mot "LOOP"
24 7
15
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  3  0  0  0  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  7  0  0 11  0  0 11  0  0 15  0  0 15  0
0  3  0  0  0  0  0  7  0  0  7  0  0 11  0  0 11  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  7  0  0 11  0  0 11  0  0 15  0  0  0  0
0  3  3  3  3  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15  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
```

(b) Code de l'image

FIGURE 3 – Image PGM représentant de l'image « LOOP » 3a et son code 3b



```
P3
# par 3 colonnes et 2 lignes,
 3 4
# ayant 255 pour valeur maximum
255
255   0   0     0 255   0     0   0 255
255   0   0     0 255   0     0   0 255
255 255   0   255 255 255     0   0   0
255 255   0   255 255 255     0   0   0
```

(a) Résultat (b) Code de l'image

FIGURE 4 – Image PPM représentant 6 pixels colorés 4a et son code 4b

2.2 Les formats compressés

Les formats d'image représentant l'ensemble des pixels sont très volumineux. Il existe plusieurs mécanismes de compression de données qui réduisent la taille des fichiers. Il y a deux grandes familles : les méthodes de compression d'image sans perte et celles avec perte.

2.3 Compression sans perte

Les méthodes les plus importantes de compression d'image sans perte sont :
- la méthode du codage des répétitions (comme LRE), utilisée sur les premiers scanners et télécopieurs ;
- le codage par quadtree ;
- le codage entropique (comme le code de Huffman) ;
- les algorithmes à dictionnaire adaptable tels que LZW.

LRE : Le premier système présenté est le codage par plages, appelé *LRE* pour « *run-length encoding* » en anglais. Cet encodage consiste, pour chaque suite de pixels de la même couleur, à coder uniquement le nombre de pixels puis la couleur de la séquence. Ainsi sur l'image noir et blanc représentée par la suite de pixels blancs (`W`) ou noirs (`N`) suivante :

```
WWWWWWWWWWWWBWWWWWWWWWWWWWWWBBBWWWWWWWWWWWWWWWWWWWWWWWBWWWWWWWWWWWW
```

Le codage est :

```
12W1B14W3B23W1B11W
```

En général le codage comporte moins de caractères (il y a bien compression). Toute fois cela n'est pas toujours le cas comme le montre l'image `WBWBWBWBWB` qui se code par `1W1B1W1B1W1B11W1B1W1B`. Ce codage est proche de celui utilisé dans l'activité.

Quadtree : Une autre technique souvent utilisée pour le codage d'images est une représentation sous forme d'arbre. Pour simplifier, les images sont supposées être en noir et blanc, carrées, et de côté 2^n.

La représentation d'une image par un quadtree repose sur deux principes :

- une image toute blanche ou toute noire est représentée uniquement par sa couleur ;
- une image multicolore se divise en quatre images carrées plus petites.

L'image de la figure 5 va être utilisée pour illustrer le principe de construction d'un quadtree. Les couleurs sont notées `B` pour blanc et `N` pour noir. Un noeud à quatre fils est noté `F(hg,hd,bd,bg)`, où `hg`, `hd`, `bg`, `bd` représentent respectivement la sous-image carrée en haut à gauche (1), en haut à droite (2), en bas à droite (3) et en bas à gauche (4), comme sur la figure 5.

L'image de la figure 5 n'est ni toute noire ni toute blanche, elle est donc divisée en quatre images de tailles égales. La première image en haut à gauche est elle toute blanche, donc elle sera repésentée par B. La seconde image en haut à droite n'est ni toute noire ni toute blanche donc elle est divisée en quatre images de tailles égales. La troisième image est elle aussi multicolore donc elle sera aussi découpée en quatre image. Enfin la dernière image est elle toute noire. Elle est donc représentée par N. Le même raisonnement est appliquée aux huit images restantes afin d'obtenir l'arbre suivant dont une représentation graphique est donnée dans la figure 6.

`F(B,F(B,B,N,F(N,B,B,B)),F(F(N,B,B,N),B,F(B,B,N,B),B),N)`

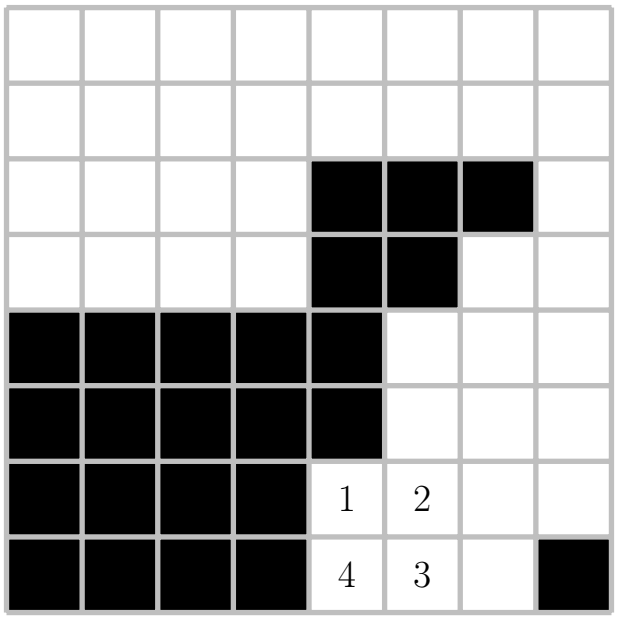


FIGURE 5 – Image noir et blanc utilisée pour construire un quadtree.

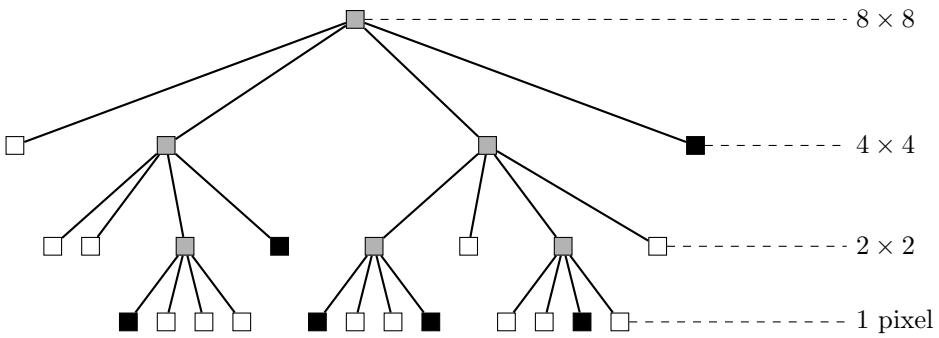


FIGURE 6 – Arbre correspondant au quadtree associé à la figure 5

Code de Huffman Il existe également des codes à longueur variable comme le code de Huffman. Dans ce cas (comme pour le Morse), les caractères les plus fréquents sont codés de manière plus concise que ceux qui sont rares. Par exemple, en Morse la lettre E est codée par un `'.'` (un seul symbole) alors que la lettre Y est codée par `'-.--'` (4 symboles). Ainsi, le code d'un message quelconque sera en moyenne plus court que si l'on avait utilisé un code de longueur fixe. Mais cela introduit une problématique nouvelle : puisque la longueur des codes des différents caractères n'est pas toujours la même, comment déterminer où s'arrête le code d'un caractère et où commence le code du suivant ? Le code de Huffman répond à cette question grâce à la propriété suivante : le code de chaque caractère est choisi de telle sorte qu'il n'est jamais le début du code d'un autre caractère (on dit que le code de Huffman est un *code préfixe*).

Le code de Huffman repose sur la construction et l'utilisation d'un *arbre de Huffman*. Un tel arbre représente un dictinnaire. Il est construit en se basant sur la fréquence de chaque caractère dans le message à coder, et indique comment coder ce message. On appelle *fréquence* d'un caractère dans un texte le nombre d'occurrences de ce caractère dans le texte considéré. Les caractères à coder sont portés par les feuilles de l'arbre et le code associé à un caractère est déterminé à partir du chemin entre la racine et la feuille portant ce caractère.

Dans l'exemple de la figure 7 un arbre de Huffman servant au codage du message "`hello leo`" est présenté.

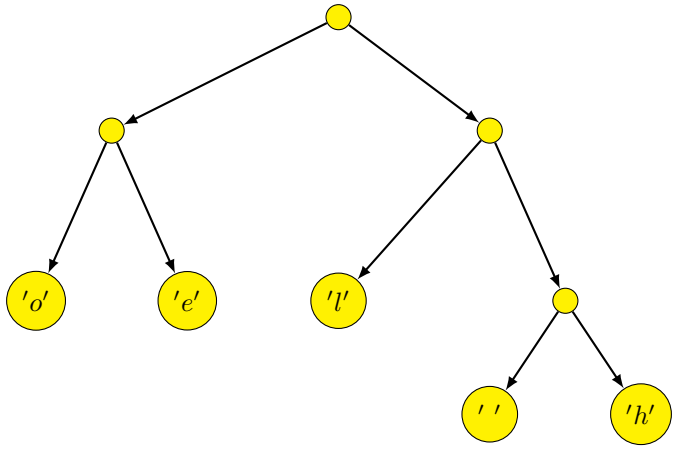


FIGURE 7 – Exemple d'arbre de Huffman.

On convient que la branche de gauche d'un arbre (en bas sur le dessin) correspond à un 0 et celle de droite à un 1. Ainsi la lettre `'o'` est codée par 11 tandis que la lettre `'h'` est codée par 000. Le message entier, "`hello leo`", sera codé par 00010010111001011011.

LZW (Lempel-Ziv-Welch) : Cette méthode est un algorithme de compression de données sans perte créé en 1984 par Terry Welch. C'est une amélioration de l'algorithme LZ78 inventée par Abraham Lempel et Jacob Ziv en 1978. Il est utilisé dans les formats d'image numérique `GIF` ou `TIFF`. L'idée de cet algorithme est de construire une table de traduction à partir des informations à compresser. Pour décompresser l'image il suffit uniquement de connaître l'image compressée qui contient toutes les informations nécessaires pour retrouver l'ensemble des données initiales. Cette méthode plus complexe construit aussi un dictionnaire comme dans la méthode de Huffman.

2.4 Compression avec perte

Il existe de nombreuses techniques de compression avec perte. Nous présentons uniquement et de manière simplifiée la méthode suivie par le format `JPEG` (*Joint Photographic Experts Group*). Le processus de compression et de décompression JPEG comporte six étapes principales représentées dans la figure 8. Le processus de compression est irréversible, puisqu'il y a perte de données.

Transformation de couleurs : L'image est transformée d'un modèle RVB (Rouge/Vert/Bleu) vers un modèle de type YCbCr (chrominance/luminance).

Sous-échantillonnage : cette étape diminue la résolution pour l'information de chrominance par rapport à l'information de luminance.

Découpage : L'image est découpée en sous-blocs de 64 (8x8) pixels.

DCT (Discrete Cosine Transform) [1] est une variante de la transformée de Fourier. Cette opération est complexe et prend du temps, elle permet de classifier les zones de changements de contrastes afin de pouvoir identifier les zones ayant les mêmes couleurs. Cette opération est réversible.

1. En français : transformée en cosinus discrète

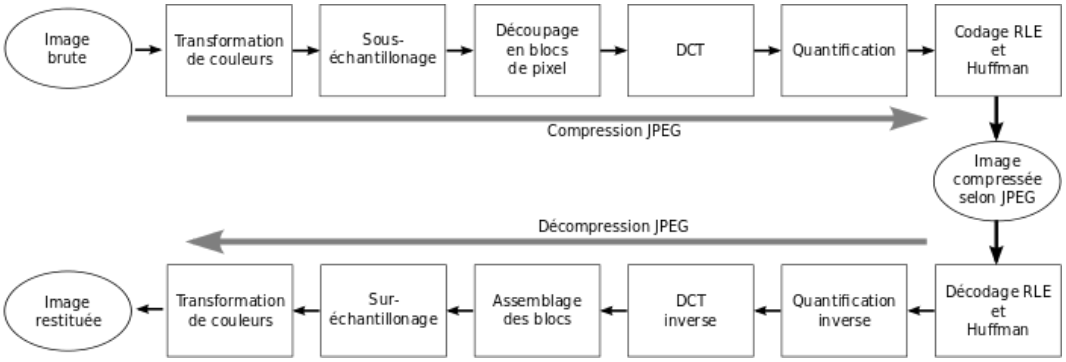


FIGURE 8 – Chaîne de compression et de décompression du format JPG.

Quantification : c'est dans cette étape qu'a lieu la compression par perte de données. L'algorithme de compression utilise le fait que l'œil humain est assez sensible à la luminance (la luminosité) mais peu à la chrominance (la teinte) d'une image. Ainsi les pixels ayant une teinte proche sont uniformisés. Les couleurs sont donc modifiées afin d'être uniformes dans les zones ayant des teintes proches (ce que l'œil humain a du mal à distinguer).

Codages RLE et Huffman : ces deux codages réversibles sont appliqués pour obtenir l'image compressée finale.

3 Les images vectorielles

Il existe de nombreux formats de fichiers vectoriels, comme `Postscript`, `PDF`, `Adobe Flash`, `Illustrator` ou `SVG`. Aujourd'hui le format standard des images vectorielles est un format PostScript appelé l'`EPSF` pour Encapsulated PostScript File ce qui signifie PostScript encapsulé. Son extension est `.eps` ou `.epsf`.

Les images vectorielles sont moins volumineuses que les images matricielles et passent sans déformation à l'échelle. Ces images sont constituées uniquement d'équations, de formules qui permettent de tracer les courbes les constituant. Les courbes de Bézier sont un des types de courbes souvent utilisées dans ces images.

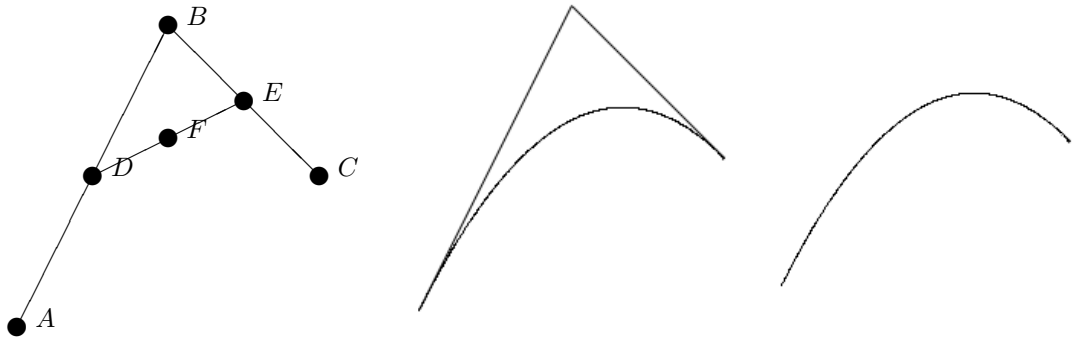


FIGURE 9 – Exemple de courbe de Bézier

Courbes de Bézier : Par souci de simplicité, nous présentons uniquement une version élémentaire des courbes de Bézier. La figure 9 représente un exemple de courbe de Bézier à droite et comment elle a été générée à partir des points A, B et C à gauche. La courbe de Bézier, de points de contrôle A, B et C, est obtenue en remplaçant la ligne brisée A, B, C par les deux lignes brisées A, D, F, et F, E, C telles que :

- D est le milieu de $[AB]$,
- E est le milieu de $[BC]$,
- F est le milieu de $[DE]$,

et en itérant ce processus de nombreuses fois. Le résultat est une courbe lisse, passant par A et C et tangente aux segments $[AB]$ et $[BC]$.

Le format `EPS` permet de décrire de telles courbes en utilisant un petit langage de programmation spécialisé (le langage `postcript`).

4 Conclusion

L'importance des images dans notre quotidien n'est plus à démontrer. Les éléves sont souvent fascinés par les images produites par un ordinateur comme les fractales. Pour un informaticien, les formats d'images ne sont que des fichiers de données. Ces données particulières de par leur structure sont plus à même d'être compressées. Ainsi de nombreux algorithmes de compression ont été développés pour les images, certains sans perte et d'autres avec perte, ces derniers utilisant le fait que l'œil humain n'est pas capable de distinguer certaines nuances et détails des images.