Projet Scientifique Informatique

Format Pivot

Nous avons pris la décision de représenter notre image sous la forme d'une matrice de pixels. Ce format permet une manipulation intuitive des données. De plus de nombreux algorithme de traitement d'image utilise ce format ce qui facilite notre documentation. Intégrer une classe pixel permet de faciliter l'ensemble des fonctions grâce à des méthodes de classe sur ces dernières.

Conversion JPEG

Nous avons choisi de créer et de conserver les pixels `YCbCr` sous 3 matrices de double. Une pour `Y`, une deuxième pour `Cb` et une troisième pour `Cr`. Nous l'avons fait sous cette forme pour que les calculs et l'intégration avec `Huffman` soient plus simple.

En effet, utiliser le type `double` permet de mener les calculs avec plus de précision et ainsi de limiter l'erreur. Le calcul réalisé lors de la DCT est obligatoirement en double à cause de `sqrt` (), et de même avec la quantification. Ainsi, il vaut mieux tout réaliser en `double` et à la fin convertir si besoin.

Utiliser 3 matrices distinctes permet d'agir avec plus d'agilité sur les 3 composantes `YCbCr` . Cela permet de s'occuper d'une variable puis d'une autre. Aussi, le code gagne en visibilité et est bien plus clair. Enfin, bien que nous n'ayons pas réussi à enregistrer l'image en format `JPEG`, nous pensons qu'il est plus simple d'enregistrer l'image en l'ayant sous forme de 3 matrices différentes qu'en une matrice de pixels.

source : JPEG — Wikipédia (wikipedia.org) et les pages Wikipédia associées aux étapes.

Enregistrer JPEG

Pour enregistrer l'image, nous avons choisi d'utiliser une `struct` pour le `header`, composé de tableaux de byte afin de pouvoir écrire directement après dans le fichier. Chaque

tableau de byte représente un segment du header (`SOF0`, `APP0`, `DHT`...) tandis que nous avons créer une série de méthodes `set_...()` pour préparer le header à l'écriture. Nous avons fait de même pour la fin du fichier (`EOI`) qui est représenté par un tableau de byte.

Source : https://youtu.be/sb8CQ9knDgI?si=wkZDP_XoRGeK8-Wh, https://web.archive.org/web/20120403212223/http://class.ee.iastate.edu/ee528/Reading material/JPEG_File_Format.pdf

Interpolation bilinéaire

Lors de la rotation chaque pixel de l'image ne correspond pas à un pixel de l'image initiale. Afin d'optimiser le rendu on réalise une interpolation bilinéaire avec les 4 voisins pour obtenir le pixel le plus pertinent possible.

https://www.iro.umontreal.ca/~mignotte/IFT6150/Chapitre7_IFT6150.pdf

Lecture des images avec bits de remplissages

Certaines images bmp on un offset bien supérieur et pour des raisons d'optimisation matériel des bits de remplissage sont ajoutés à l'image. On doit donc les supprimer à la lecture car le nombre de bytes constituant l'image n'est plus égal au nombre de bytes entre l'offset et la fin du fichier. Pris en charge via get_image(). Désactivé par défaut car erreur avec certaines fonction.

https://www.gladir.com/LEXIQUE/FICHIERS/bmp.htm

Stéganographie

Pour obtenir une meilleure conservation de couleurs lors de l'encodage et décodage de l'image cachée on utilise une simple compression des bytes de l'image à cachée sur un certain nombre de bits. L'implémentation actuelle de le stéganographie nous permet de choisir les chaînes de couleurs et la quantité de bits par chaîne à utiliser pour cacher notre image
On peut aussi cacher jusqu'à trois images dans une image, en perdant les informations de couleurs