Dominic Bergeron 1405236
Jules Thuillier 1521088

Le 24/11/2016
A Montréal

Rapport TP2 - INF4710

Introduction aux technologies multimédia

A2016 - Travail pratique #2
Compression avec pertes (JPEG/MPEG)

Présentation du pipeline (toutes les étapes)

1. **Encodage**
    a. **Conversion RGB → YCrCB**
    On applique les formules de conversion des espaces de couleur pixel par pixel en itérant sur l'image de départ. Si le sub-sampling est activé, on ne remplit les canaux Cr et Cb q'une donnée sur quatre.
    On effectue un clamping de chaque valeur entre 0 et 255 afin de ne pas entrer de valeurs aberrantes dans l'image de sortie.

    b. **Découpage**
    Il s'agit ici d'une simple transformation de la façon dont sont stockées les données. On subdivise l'image en blocks de 8x8 pixels en utilisant 4 boucles imbriquées.

    c. **DCT**
    La Transformée en cosinus discrète permet de changer la représentation de l'image dans le domaine fréquentielle. Cette transformée nécessite de nombreux calculs complexes pour le processeur. Nous avons pu comparer nos résultats avec ceux d'OpenCV et avons remarqué une légère différence, bien qu'il s'agisse d'une compression en théorie sans pertes.

    d. **Quantification**
    La quantification permet de mieux la compresser l'image en éliminant les hautes fréquences, peu sensibles par l'œil humain. On divise chaque bloc de 8x8 pixels par les valeurs de la matrice de quantification.

    e. **Zigzag**
    Cette opération permet de transformer le format de stockage des blocks de 8x8 pixels sortis de la quantification de matrice 2D à un vecteur 1D. Cela est nécessaire pour appliquer l'encodage de Huffman par la suite.

    f. **Huffman**
    On applique ici le codage entropique de Huffman, qui est une compression sans pertes basé sur la fréquence de répétition des symboles.

2. **Décodage**

    a. **Huffman inverse**
    On décompresse en inversant le codage de Huffman

    b. **Zigzag inverse**
    On reconstruit les blocks de 8x8 pixels à partir des vecteur 1D.

c. **Quantification inverse**
La quantification inverse possède une implémentation similaire à la quantification, à la différence qu'on multiple les blocs de pixels par la matrice de quantification.

d. **DCT inverse**
L'algorithme de la DCT inverse est assez semblable à celui de la DCT, seulement avec des formules différentes. Nous avons aussi pu comparer nos résultats avec ceux d'OpenCV et avons encore une fois remarqué une légère différence.

e. **Découpage inverse**
Il s'agit de l'opération inverse du découpage. On prend les matrices de 8x8 pixels pour les remettre ensemble dans le format initial de l'image.

f. **Conversion YCrCb → RGB**
L'opération est très semblable à la conversion en YCrCb mais dans l'autre sens.

Discussion + évaluation perte, conversion couleur seulement

Avec le sub-sampling, on remarque que les pertes sont assez importantes, notamment lorsqu'il y a du bruit sur les images, ou beaucoup de changements rapides de couleur. Cela est logique car le sub-sampling est un opération consistant à retirer des données (compression) sur les canaux Cr Cb, donc ceux comportant les couleurs.
Même sans sub-sampling, même si on ne les voit pas à l'œil nu, il y a un risque de pertes non négligeables dues à la précision des calculs. Les constantes utilisées pourraient être plus précises, et on procède à un arrondi pour stocker la valeur sur un byte qui entraine évidemment des pertes.

Avec Sub-Sampling

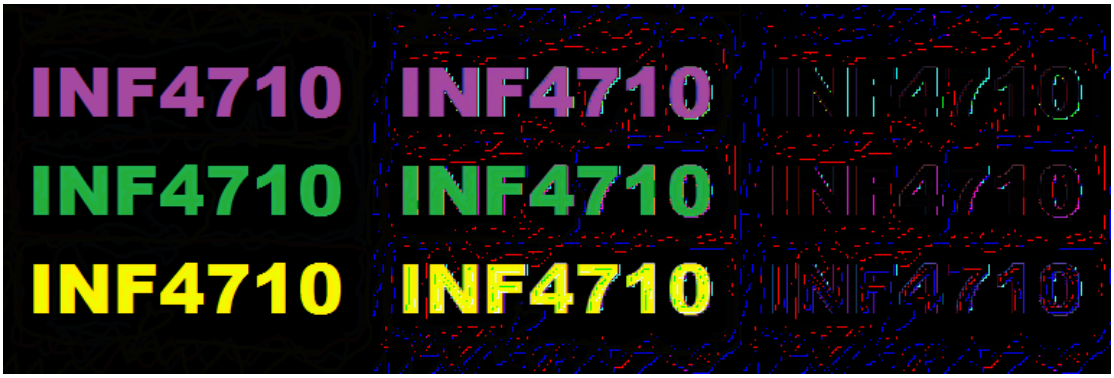

Figure 1 : logo noise

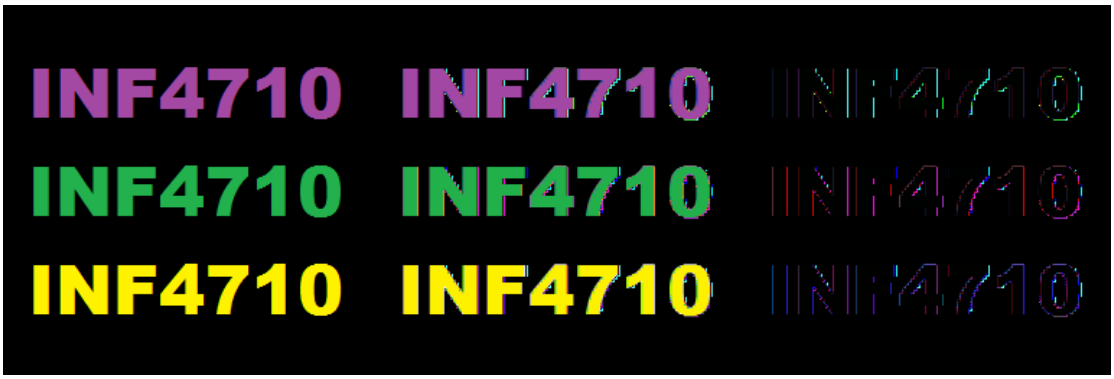

Figure 2 : logo

Figure 3 : Peppers

Sans Sub-Sampling

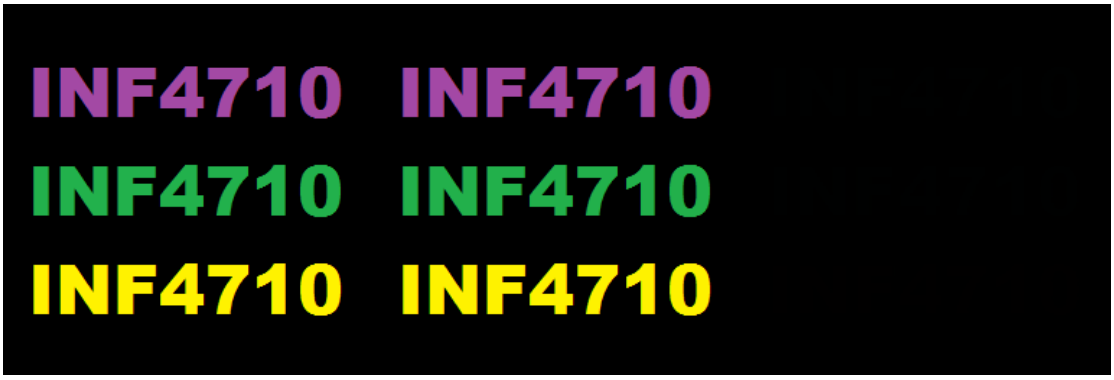

Figure 4 : logo noise

Figure 5 : Peppers

Discussion + évaluation perte, DCT seulement

La DCT, bien que censé être sans perte, révèle des pertes non négligeables sur l'image. Il est à noter que sur les images ci-dessous, des pertes proviennent aussi des étapes précédentes du pipeline de compression. En pratique des pertes sur la DCT sont obligatoires à cause de la taille des valeurs stockées. Les opérations effectuées lors de la DCT donnent des nombres avec beaucoup de chiffres après la virgules qui sont arrondis pour entrer dans des variables.

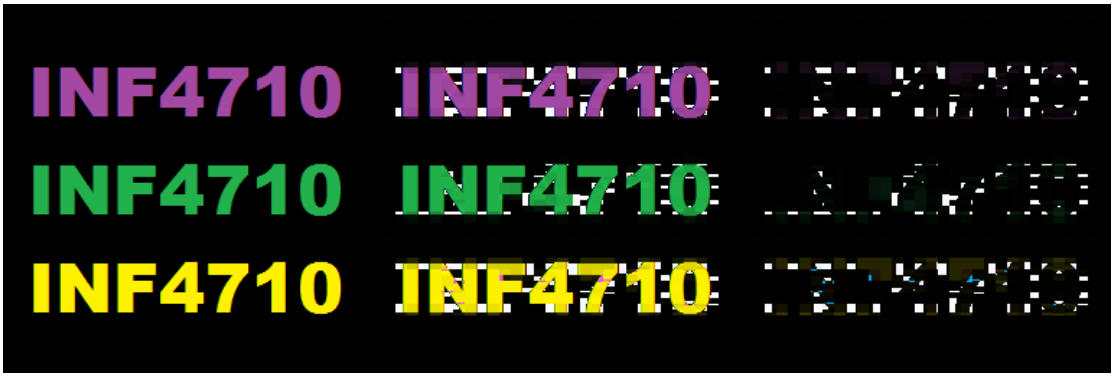

Figure 6 : logo

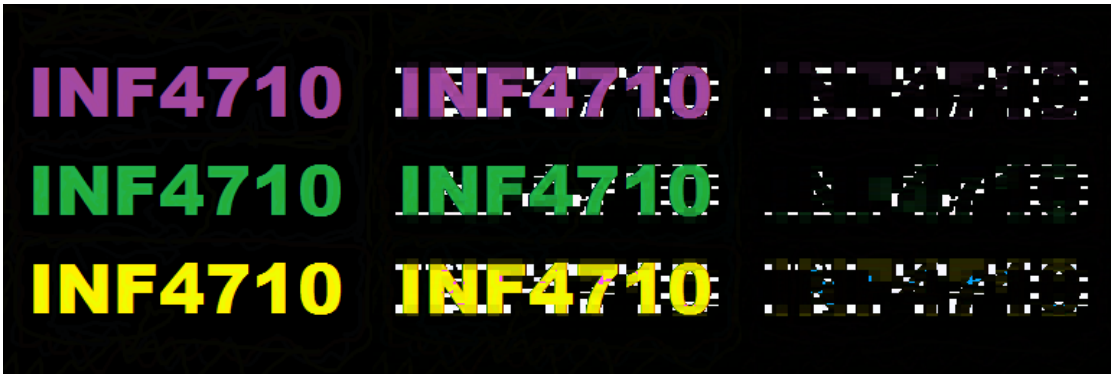

Figure 7 : logo noise

Figure 8 : cameraman

Discussion + évaluation perte, pipeline complet

Le pipeline comprend 6 étapes. Deux d'entre elles (découpage et zigzag) ne consistent qu'à changer la façon dont sont stockées les données et n'entraines aucune perte.
Dans les 4 étapes restantes, deux sont théoriquement sans pertes (Huffman et DCT), mais vu que les calculs se font sur ordinateurs, des pertes proviennent des représentations des numériques.
Enfin des erreurs viennent aussi de la précision des calculs, et des valeurs utilisées pour ces derniers.

Taux de compression

Voici le tableau des taux de compressions des images, selon la quantification et le sub-sampling.

Image	F10		F50		F90		F100	
	no sub	sub	no sub	sub	no sub	sub	no sub	sub
Airplane					0.784		0.574	0.735
Baboon							0.285	0.603
Cameraman							0.663	0.731
Lena							0.469	0.696
Logo Noise							0.682	0.802
Logo							0.626	0.766
Peppers							0.447	0.692

En ce qui concerne le sub-sampling, on remarque qu'il permet de gagner jusqu'à 50% en taille sur l'image final compressé, donc bien plus que s'il est utilisé seul. Cela montre que c'est la combinaison des méthodes de compression qui fait la force du pipeline.
La quantification permet aussi de largement améliorer la compression, mais au dépend d'une perte de qualité des images. Il faut donc trouver un juste milieu entre la qualité voulu et la taille des images.