École Polytechnique de Montréal
Département de génie informatique et génie logiciel

INF8770
Technologies multimédias

Rapport de laboratoire #2

Pipeline JPEG2000

Milan Lachance [1897637]
Guillaume Lévesque [1904308]

Soumis à M. Mehdi Miah
2 novembre 2020

Introduction

La compression des images sert à réduire le nombre de bit utilisé pour représenter, stocker et échanger les fichiers sur le réseau. Plusieurs solutions sont des combinaisons de techniques appliquées une après l'autre pour obtenir une compression efficace. Le sujet de ce rapport est le pipeline JPEG2000. Par défaut, nous implémentons le pipeline avec les paramètres suivants :

- étape 1 : RGB -> YUV, sous-échantillonnage 4:2:0
- étape 2 : DWT à 3 niveaux de récursion
- étape 3 : Quantification à zone morte de Seuil = 4 et Pas = 4
- étape 4 : LZW tel que vu en classe

Nous répondrons aux questions de l'énoncé tout en présentant notre implémentation pour les 6 étapes demandées. Des expérimentations seront effectuées, puis analysées. À noter que pour chacun des tests effectué, seulement l'élément testé sera différente de la paramétrisation par défaut.

Explication rapide de la structure du code

Nous avons réalisé le travail pratique avec du code en Python 3. Nous avons créé une classe pour chaque étape du pipeline. Ceci nous a permis de vérifier nos données et de comparer facilement les résultats de chaque étape. Chacune des classes enregistre les données transformées pour l'étape du pipeline correspondante. De plus, chacune des classes contient les méthodes nécessaires pour recevoir les données de l'étape précédente du pipeline et appliquer les transformations subséquentes, ainsi que le code pour faire le chemin inverse. Nous avons donc les classes suivantes pour effectuer l'ensemble du pipeline :

$$\text{yuv} \rightarrow \text{subsample} \rightarrow \text{dwt} \rightarrow \text{quantize} \rightarrow \text{lzw}$$

La classe compressedimage permet de prendre une image source et d'appliquer le pipeline au complet en faisant appel à toutes le classes mentionnée précédemment, ainsi qu'à la classe rgbimage qui permet de recevoir une image en format YUV et de la transformer en format RGB et de l'afficher. En ce qui concerne le calcul du taux de compression, nous avons utilisé la formule suivante pour trouver la taille initiale : $T_o = longueur_{image} * largeur_{image} * 3 * 8$ pour représenter les 3 canaux de 8 bits en RGB.
Notre implantation du la DWT demande que l'image traitée ait des dimensions de puissances de 2. Nous avons décidé de ne pas considérer les cas où il faut modifier l'image avant d'appliquer le pipeline, car ces situations n'ont rien à voir avec les opérations du pipeline en soit.
Lors du chemin inverse pour afficher le résultat de notre compression, nous n'avons pas implémenté la transformée inverse de LZW. Puisque cet algorithme de compression est sans perte, il n'a aucun impact sur l'aspect de l'image compressée. Nous avons donc utilisé les données de l'image quantifiées pour effectuer le chemin inverse.

Questions

Q1 : RGB/YUV

Le changement de l'espace de couleur RGB <-> YUV permet une séparation de la luminance (Y) et de la chrominance (UV). Nous utilisons les formules qui permettent un changement d'espace de couleur sans perte. Notre implémentation du changement de l'espace de couleur est simplement d'utiliser les formules suivantes :

$$Y = \frac{R + 2G + B}{4} \tag{1}$$

$$U = B - G \tag{2}$$

$$V = R - G \tag{3}$$

Cette opération prépare nos données pour effectuer un sous-échantillonage de la chrominance et ainsi effectuer une première compression des données. L'oeil humain étant plus sensible aux changements de luminance, nous conservons toutes les données du canal de luminance. La paramétrisation J :a :b permet de décider du nombre de données de chrominance conservées durant le sous-échantillonage. Voici quelques exemples avec leur taux de compression $TC = 1 - \frac{T_c}{T_o}$:

TABLE 1 – Taux de compression selon J:a:b

J:a:b	4:2:0	4:2:2	4:4:4
TC (%)	90.92	89.88	89.11

FIGURE 1 – 4:2:0

FIGURE 2 – 4:2:2

FIGURE 3 – 4:4:4

Les sous-échantillonnages 4:2:0 et 4:2:2 offrent un meilleur taux de compression, tout en conservant une bonne qualité d'image observée à l'oeil. Toutefois, la différence visuelle entre les 3 valeurs de J:a:b est faible et les taux de compression sont très semblables. Il est difficile de déterminer un choix qui serait meilleur que les autres dans tous les cas.

Q2 : Usage de la DWT

Les opérations de transformées changent les référentiels (bases) de l'image pour séparer les composantes de façon à les rendre indépendantes, créer des patrons de valeurs semblables et faciliter la compression par la suite. La DWT seule ne compresse donc pas l'image. La DWT consiste à obtenir des ensembles d'approximations et de détails reliés aux intensité des signaux de l'image. Des filtres passe-haut et passe-bas sont appliqués sur notre image pour mettre en évidence les changements d'intensité de signal de l'image. En combinant les résultats des filtres avec l'image redimensionnée, nous avons une image à 4 cadrans : nous retenons l'image approximative, les détails verticaux, les détails horizontaux, puis les détails diagonaux. Nous avons implémenté la DWT de façon à traiter chacun des canaux de façon indépendante, puis nous ajoutons les résultats un à la suite de l'autre pour former l'image transformée. L'image de base en est réduite en taille, puis on peut appliquer d'autres récursions à partir de cette-dernière. Nous avons appliqué 3 récursions dans notre implémentation de base.

Q3 : Impact du niveau de récursion de la DWT

Il est intéressant de tester la DWT avec plusieurs niveaux de récursion pour voir à quel point on conserver une petite image d'approximation. Avec plus de récursions, il y aura une plus grande partie des signaux sera mise à 0 lors de la quantification, ce qui devrait augmenter le taux de compression. Voici quelques exemples :

TABLE 2 – Taux de compression selon le nombre de niveaux (DWT)

nb Niveaux	1	2	3	4
TC (%)	87.97	90.33	90.92	91.07

FIGURE 4 – DWT 1 niveau

FIGURE 5 – DWT 2 niveaux

FIGURE 6 – DWT 3 niveaux

FIGURE 7 – DWT 4 niveaux

Il est intéressant de noter que les taux de compressions ne sont pas très loin les uns des autres, spécialement entre les niveaux 3 et 4. Ceci est dû à la taille de moins grande de l'image à partir de laquelle on effectue la transformée. Considérant les rendements décroissants que nous offrirons les récursions subséquentes, on peut affirmer que 3 ou 4 niveaux sont suffisants.

Q4 : Quantificateurs

4.1) Puisque nous utilisons un quantificateur uniforme a zone morte, la paramétrisation de celui-ci se fait en changeant le seuil et le pas. Le seuil détermine à partir de quelle valeur la donnée est jetée, puis le pas détermine l'intervalle des valeurs initiales qui seront attribuées à la nouvelle valeur quantifiée. Dans notre pipeline de base, nous avons utilisé un seuil de 4 et un pas de 4. On leur attribut la même valeur pour respecter la version uniforme. Nous avons aussi expérimenté avec différents seuils pour visualiser l'impact de celui-ci sur la qualité de l'image finale et son taux de compression :

TABLE 3 – Taux de compression selon différents seuils et pas de quantification

Seuil et pas	2	4	8	16
TC (%)	85.04	90.92	95.09	97.55

FIGURE 8 – seuil et pas = 2

FIGURE 9 – seuil et pas = 4

FIGURE 10 – seuil et pas = 8

FIGURE 11 – seuil et pas = 16

4.2) On remarque une grande différence visuelle entre les différents seuils. Particulièrement, le paramètre de seuil = 4 est intéressant à observer. Le taux de compression est déjà bon, alors que la qualité de l'image à l'oeil reste très semblable à l'image avec un seuil de 2. La valeur de seuil = 4 est donc un bon compromis pour une bonne qualité d'image avec bonne compression et si la taille de l'image à compresser n'est pas minuscule (un aperçu d'image, par exemple), il sera assez rare de choisir un seuil de 8 au prix de la qualité de l'image.
Il y a un effet de bloc qui se produit quand on augmente la zone morte de beaucoup (voir seuil =

16). Ceci est attribuable au grand nombre de données qui sont mises à zéros lors de la quantification. Tous les changements de signal faibles se font attribuer la valeur 0. À noter qu'à bien des endroit, la luminosité change à "l'intérieur" d'un bloc, ce qui nous permet de conserver un certain niveau de détails car les grandes différences de signal seront conservé.

Q5 : Taux de compression

Pour évaluer la performance du pipeline que nous avons programmé, nous l'appliquons sur différents types d'images. Les images de gauche sont compressées, alors que les images de droite sont les originales. Les paramètres utilisés sont ceux par défaut :

TABLE 4 – Taux de compression selon des types d'images

type	noir/blanc	très coloré	photo	lent dégradé
TC (%)	83.76	71.90	87.20	98.23

FIGURE 12 – compression d'une image en noir et blanc

La compression est bonne pour une image en noir et blanc, mais pas autant que pour la photo utilisée auparavant. Toutefois, cette donnée pourrait s'avérer fausse, car on peut représenter la photo de noir et blanc avec un seul canal (luminance). Ce faisant, sa taille originale serait 1/3 de celle utilisée dans notre algorithme. D'un autre côté, nous aurions pu implémenter une version de l'algorithme pour recevoir un seul canal et ainsi bénéficier d'une taille finale également plus faible.

FIGURE 13 – compression d'une image très colorée

Dans le cas d'une image très colorée, le taux de compression est moins bon que pour l'image en noir et blanc. Toutefois, la qualité de l'image est très bonne, car la présence de beaucoup de changements brusques de signaux nous permet de conserver un grand niveau de détails lors la DWT.

FIGURE 14 – compression d'une photo standard

La compression d'une photo standard est bonne, et on ne perd pas beaucoup de précision au niveau de l'image vue à l'oeil nu. De plus, une photo de personne contient beaucoup de section à faible différence entre les signaux (sur la peau par exemple). Ceci permet une meilleure compression suite à la transformée DWT.

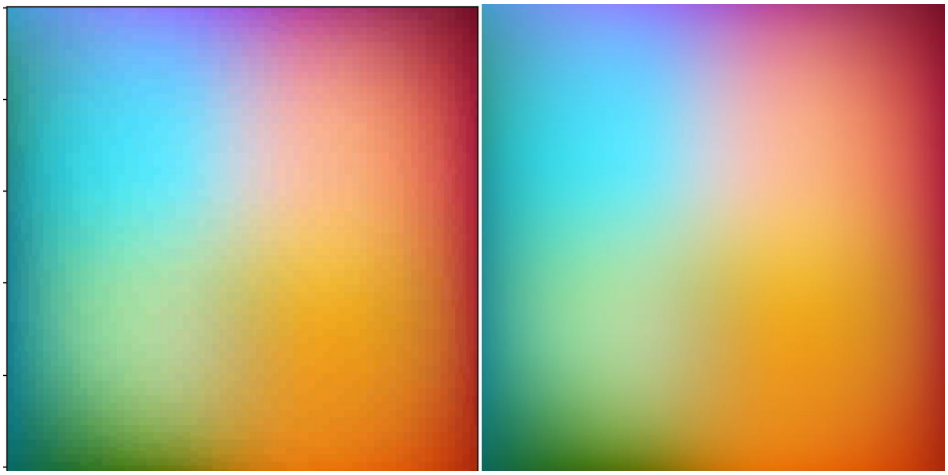

FIGURE 15 – compression d'une image avec lent dégradé

La compression est excellente pour ce type d'image, principalement à cause de l'absence de changement brusque du signal. Toutefois, ceci fait en sorte que la reconstruction de l'image dépend beaucoup plus du cadrant en haut à gauche de la DWT que des autres cadrans, qui conservent les détails de changement de signaux. Les couleurs sont sous-échantillonnées et on observe une diminution non-négligeable de la qualité de l'image et la formation de blocs.

Q6 : DWT vs DCT

Un avantage visuel de DWT par rapport à DCT est l'absence d'effet de bloc. Cet artefact est visible aux frontières des blocs de pixels. DCT effectuerait un découpage en blocs avant de traiter les données, alors que DWT traite l'image du début à la fin sans séparation. Attention, un effet de bloc peut quand même se produire, mais ce ne sera pas à cause de la transformée elle même. C'est plutôt lors de la quantification que DWT devient "avec perte" et un certain effet de bloc se produit.

Conclusion

Ce travail a été intéressant pour comprendre le comportement des pipelines de compression. Les taux de compressions selon plusieurs paramètres a été particulièrement intéressant à modéliser et nous comprenons mieux le fonctionnement des formats que nous utilisons tous les jours.
La difficulté la plus importante que nous avons rencontré a été de tester nos résultats à mesure que nous implémentions le pipeline. Les transformations des données pour pouvoir les visualiser nous a fait perdre bien du temps. Une autre difficulté rencontrée lors de la programmation a été de bien implémenter les formules selon les canaux de données. Encore une fois, une question de type dans python nous a fait tourner en rond. En tout, nous sommes satisfaits de ce travail pratique.