INF8770 - Techniques Multimédia

Automne 2018

Travail Pratique 2
Pipeline JPEG2000

Groupe 1

1795724 - Alexandre Hua
1788028 - Francois Toulouse

2018 / 10 / 26

Code utilisé

En ce qui concerne la conversion en couleurs, nous avons implémenté les fonctions suggérées de façon à rapidement pour pouvoir les appliquer sur les matrices numpy de l'image.

La sélection de couleurs 4:2:0 a été réalisée en recopiant quatre fois la moyenne des groupes de pixels sur les canaux U et V, diminuant effectivement du quart la résolution (et la quantité d'information) des variations de couleurs (Chromas) de l'image.

L'algorithme de Transformée par Ondelettes Discrète (Discrete Wavelet Transform, ou DWT) et son algorithme opposé a été implémenté par une fonction récursive qui découpe l'image successivement en x et en y, donnant un dictionnaire contenant les résultats f11, f1h, fh1, et fhh. Selon le niveau de récursion, f11 est aussi découpé et remplacé par un nouveau dictionnaire, ayant subi la même transformation. La transformation inverse consiste à déterminer le niveau de récursion du dictionnaire en entrée et en additionnant les valeurs f1h, fh1 et fhh pour chaque niveau de récursion.

La quantification du vecteur résultant est réalisée selon un "dead-zone" de 0 à d donnant une valeur constante de zéro, et une fonction linéaire de d à 255 réajustée de 0 à 255, découpée par pas discrets à l'aide des propriétés de la division entière.

Une fonction de quantification différente, qui préserve les valeurs négatives présentes dans les matrices de différences f1h, fh1, et fhh, a été utilisée pour générer les images. Malheureusement, cette fonction était incompatible avec l'encodeur LZW, décrit ci-dessous.

Pour la partie d'encodage avec la méthode LZW, nous nous sommes basés sur le code de notre professeur sur le répertoire Github https://github.com/gabilodeau/INF8770/blob/master/Codage%20LZW.ipynb, mais nous avons effectué de nombreuses modifications. En effet, l'exemple donné dans le code permet d'encoder des messages de type String. Cependant, dans notre cas, les données en entrées que nous désirons passer sont des valeurs de nombre entier contenues dans un vecteur. De plus, l'exemple ne fait qu'encoder le message lui-même, mais afin de pouvoir décoder correctement le message, nous avons besoin d'autres données supplémentaires, telles que le dictionnaire original contenant les symboles uniques, ainsi qu'un nombre indiquant le nombre de symbole unique.

Pour la partie de décodage de LZW, nous n'avons pas réussi à trouver de bons exemples sur lesquels nous pouvions basés notre code, donc nous avons dû écrire le décodeur entièrement par nous-même.

Voici la trame de notre message encodé :

|__nombre de symbole(8bits)__|__dictionnaire__|__message encodé__|

Remarque que nous n'avons pas utilisé de séparateur, car le nombre de symbole permet justement de savoir où s'arrêter avant d'atteindre le message encodé.

Expériences réalisées

L'expérience que nous avons effectué est essentiellement de répéter le pipeline de compression d'image JPEG pour différentes images avec des propriétés différentes. En effet, nous avons voulu vérifier quel effet est ce que certains facteurs ont sur le taux de compression finale de l'image. Ces facteurs sont :

- La variété de couleur utilisée dans l'image
- La tonalité de gris
- Les dimensions de l'image

Les étapes de la pipeline sont les suivantes :

- Conversion des valeurs RGB de chaque pixel de l'image en YUV
- Échantillonnage des pixels selon une proportion de 4:2:0
- Application de la transformée en ondelettes discrète
- Quantification les valeurs produits par la DWT
- Compression avec la méthode LZW

Finalement, lorsqu'on obtient notre compression finale, il s'agit maintenant d'effectuer tous les étapes inverses pour ré-obtenir notre image originale. Parmi les nombreuses étapes, certaines méthodes sont des compressions avec perte, donc nous nous intéressons également sur la qualité de l'image décompressée par rapport au taux de compression.

Question 1

D'après nos observations, il est avantageux de changer d'espace de couleurs afin de se concentrer sur un canal en particulier plutôt que trois. En RGB, la luminosité et la combinaison des 3 valeurs est importante à la fois pour déterminer le contraste sombre-lumineux et pour la variance en teinte des couleurs. Avec YUV, le canal Y sert exclusivement à la luminosité, et U, V servent à conserver la variance de couleur.

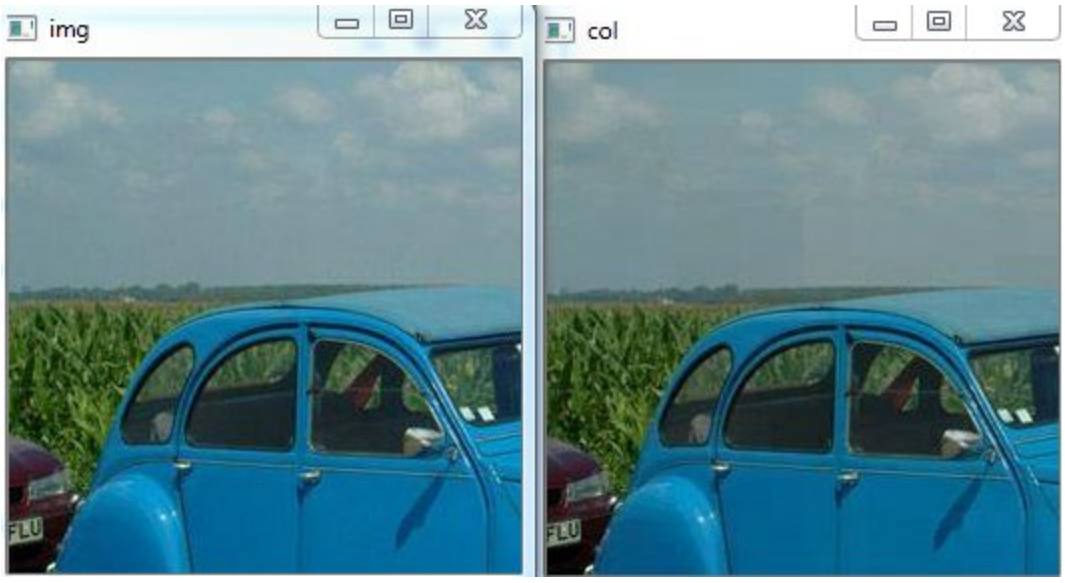
Puisque le contraste lumineux est plus perceptible et plus souvent important dans l'image, on peut bénéficier de l'échantillonnage 4:2:0 en limitant les pertes d'informations: dans un même carré de 4 pixels, Y est beaucoup plus susceptible de changer que U et V.

Question 2

L'algorithme DWT est, à la base, un algorithme sans perte. Lorsque utilisé seul, il est incapable de réduire la taille de l'image: chaque paire de pixels d'un canal de couleur (2*8 bits) se retrouvent encodés selon deux nouveaux "pixels" contenant leur moyenne et leur différence (2*8 bits). La DWT existe, dans ce contexte, pour tenter de créer des séquences répétitives pour faciliter la compression des zones n'ayant pas assez de différences.

Question 3

Un niveau de récursion plus élevé ajoute certains effets secondaires et artefacts à l'image: le découpage par moyenne et différence mais l'enregistrement en valeurs discrètes forcent certains pixels d'être décalés dans leur valeurs. Cela est malheureusement presque imperceptible dans la photo RGB fournie, mais en voici un exemple à partir d'un échantillon permettant une récursion extrême, où l'on peut voir un faible motif de damier apparaissant sur l'image de droite:



R = 8, Des artefacts carrés apparaissent en damier sur l'image reconstruite.

Question 4

Pour la quantification, augmenter le "step" (S) et le "dead-zone" (D) ont pour effet de diminuer l'information contenue dans l'image en traitement: ici, plus "S" devient élevé, moins de tons de couleurs distincts deviennent présents dans l'image. Du même fait, puisque la quantification s'applique sur les canaux U et V, l'espace de couleur possible de l'image se restreint.

R = 1, D = 0, S = 1: Résultat sans perte.

R = 1, D = 0, S = 4: On commence à distinguer des limites claires dans les gradients de couleurs.

R = 1, D = 0, S = 16: En plus de la luminosité, on peut remarquer une perte de coloration pour les couleurs moins vives du ciel et du pavé. On remarque aussi une perte de détails, évident par l'apparition de divers artefacts alignés sur la grille de pixels 2x2.

La quantification avec des valeurs de "D" progressivement plus élevées, dans le but d'éliminer des valeurs trop proches de zéro, pousse l'image à devenir de plus en plus foncée, vu la plus grande proportion de pixels ajustés avec des valeurs plus basses:

R = 1, D = 0, S = 1: Résultat sans perte.

R = 1, D = 16, S = 1: la résolution est diminuée, puisque les valeurs de différences minimes sont aussi sujettes au filtre de “dead-zone”. Notez aussi la perte de détails dans les ombres et la perte de coloration dans le ciel.

R = 1, D = 40, S = 1: l’image devient presque monochrome, à l'exception des teintes saturées de l’automobile. Limage s’est significativement assombrie.

Avec un certain niveau de récursion DWT, la quantification qui inclut un “dead-zone” finit par complètement détruire les valeurs de différences et fait ainsi perdre la majorité des détails:

R = 3, D = 0, S = 1: résultat sans perte.

R = 3, D = 16, S = 1: la résolution paraît diminuée. Notez que certains détails à haut contrastes (tels les roues ou les plaques) survivent tant bien que mal à cette perte.

Ce phénomène vient du fait que les valeurs de “différence” f1h, fh1, et fhh résultant de l’application de l’algorithme DWT subissent aussi cette même transformation: malgré certains détails à très hauts contrastes (tel le lettrage des plaques), peu des zones de l’image ont une variance chromatique ou différence de luminosité supérieure à la valeur de “D”, ainsi ces segments 4x4 deviennent colorés de façon uniforme.

Question 5

Voici les résultats que nous obtenus à partir de la compression avec la méthode LZW sur différents types d’image, soit une image en couleur typique, une image de ton gris, une image de fond complètement noir, une image de fond avec des couleurs en dégradé et une image très colorée.

	taille en binaire (bits)	taille en LZW (bits)	taux de compression
Photo Standard	5760000	1893068	67.134%
Noire et Blanc	9830400	2583212	73.722%
Fond noir	30000	1706	94.313%
Fond dégradé	11620800	4173560	64.085%
Image très colorée	7872000	3381200	57.048%

Tab.1 Taux de compression selon le type d’image

Comme on peut le voir dans le taux de compression, nous avons pu compresser nos images. On peut remarquer le taux de compression de l’image de fond noir est beaucoup plus élevé, ce qui signifie une plus faible compression de l’image. Il semble être le cas également pour l’image de ton gris. Nous pouvons conclure que la compression d’une image semble être meilleure dans le cas où l’image comporte une plus grande variété de couleur. Similairement pour l’image très colorée, comme cette image comporte plusieurs couleurs de plusieurs intensités différentes, celle-ci connaît la meilleure compression.

Notez que les images que nous avons utilisées pour notre expérience sont de dimensions variées. Certaines images sont très petites comme l’image de fond noir qui est de 50x50 pixel, alors que d’autres sont beaucoup plus grandes, ce qui explique la grande différence sur le nombre de bits d’image en image.

Question 6

L’application de plusieurs filtres de quantification ayant un “Dead-Zone” non-zéro consécutifs a pour effet de multiplier la perte de données du filtre.

Test de filtre double, haute qualité:

R = 1, D = 2, S = 2, Appliqué deux fois: on remarque que les découpages 2x2 de la compression par DWT deviennent plus prononcées.

Test de filtre double à basse qualité:

R = 1, D = 0, S = 16, Appliqué deux fois: le deuxième passage du filtre semble amplifier et propager les artefacts lumineux et chromatiques du premier passage.

Tests de filtres alternés:

R = 1, D = 0, S = 16, Puis R = 1, D = 2, S = 2: le passage du deuxième filtre ne semble pas avoir affecté la qualité de l'image au delà des effets du premier filtre.

R = 1, D = 2, S = 2, Puis R = 1, D = 0, S = 16: le passage du deuxième filtre donne des résultats similaires à son application seule. Certains effets et artefacts sont différents de son application seule.