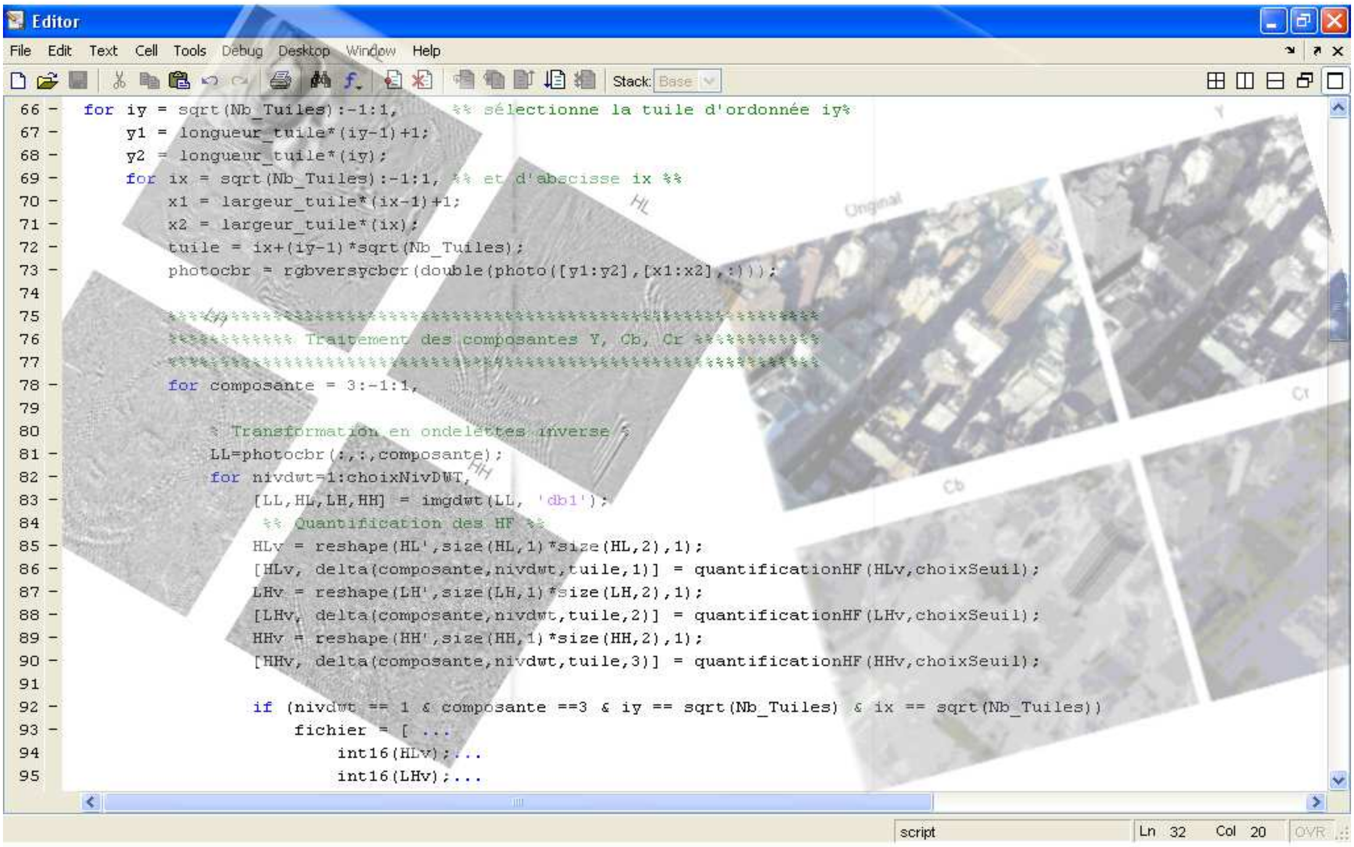
Projet de synthèse I2 – Traitement de l'image
Compression

Sébastien FRANÇOIS, Yvonnick BRUNET

avril 2006

(http://seb.france.free.fr/eseo/I2-TST/traitement-d'image/)

Résumé – Le but de ce projet est de découvrir le fonctionnement d'un système complet de compression d'images, par réalisation des différentes étapes par simulation sur Matlab. La compression mise en oeuvre est un hybride entre les normes JPEG et JPEG2000.



1 Introduction

Le projet de traitement numérique de l'image proposé en spécialité traitement du signal concerne la mise au point d'une technique de compression d'une image couleur.

Il met en œuvre des processus spécifiques au traitement d'image, tels que le changement d'espace de couleurs, le filtrage (transformation en ondelettes), mais aussi des notions beaucoup plus générales telles que la quantification, le seuillage, le codage DPCM, le sous-échantillonnage.
Le système permettra la compression et la décompression des images, de manière à pouvoir évaluer la qualité de la compression par des critères mathématiques tels que la puissance du rapport signal sur bruit, le taux de compression ou bien l'erreur quadratique moyenne.

2 Descriptions des différentes étapes de la compression

2.1 Décomposition en tuiles

La compression peut tirer partie du fait qu'une image naturelle comporte en général des zones distinctes, par exemple dans le cas d'un paysage, il y a souvent une zone non négligeable et homogène formée par le ciel. Un simple découpage en tuiles de l'image permet d'améliorer le taux de compression, car l'algorithme est appliqué sur chaque tuile individuellement. Ces tuiles peuvent alors ensuite être plus ou moins compressées en fonction de leur contenu.

En effet, en découpant une image en plusieurs tuiles, on augmente la probabilité qu'une ou plusieurs de ces tuiles aient un contenu uniforme. Si la tuile à traiter est uniforme, la majeure partie de l'information visuelle est transmise dans les composantes basses fréquences. Celles-ci pourront être compressées très efficacement par un codage tenant compte des ressemblances entre des pixels adjacents.

L'ensemble des étapes qui suivent seront alors appliquées à chaque tuile.

Dans le cadre de cette étude, nous avons limité le nombre de tuiles aux valeurs suivantes : [1, 4, 16, 64] (puissances paires de 2).

2.2 Transformation intercomposante RGB vers YCbCr

La représentation la plus courante d'une image en informatique est le RGB24, c'est un espace dans lequel chaque composante est une couleur primaire Rouge, Vert et Bleu. Chaque composante est quantifiée sur 8 bits, un pixel est donc défini sur 24 bits (ou 24bpp). La couleur d'un pixel est produite par synthèse additive des différentes composantes pondérées.

Cependant, dans cet espace l'information est répartie de manière relativement égale sur l'ensemble des plans. On peut en général, de manière expérimentale, tout à fait deviner le contenu d'une image en ne visualisant qu'un seul des plans RGB.
Il existe d'autres espaces dans lesquels la répartition se fait en terme de luminosité et de couleurs par exemple. C'est le cas de l'espace YCbCr, notamment utilisé dans le codage des couleurs des systèmes de télévision.

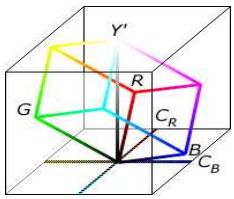


FIG. 1: Espaces de couleur RGB et YCbCr

Y est une composante de luminance tandis que Cb et Cr sont dites de chrominance et forment l'information de couleur de l'image. La composante Y supporte l'essentiel de l'information de l'image. On assimile son contenu à une image noir et blanc équivalente (ce qui permettait à l'époque une rétrocompabilité avec les postes de télévision noir et blanc).

Cette transformation permet une compression plus efficace car l'information est principalement condensée dans le plan Y, on peut donc compresser plus efficacement Cb et Cr.

Nous avons créé les fonctions suivantes :

rgbversycbcr	Variable d'entrée : `RGB`: *Image RGB de classe «double».* Variable de sortie : `YCBCR`: *Image YCbCr de classe «double».*
ycbcrversrgb	Variable d'entrée : `YCBCR`: *Image YCbCr «double».* Variable de sortie : `RGB`: *Image RGB «double».*

Par la suite chaque plan est traité individuellement.

2.3 Transformation en ondelettes

La transformée en ondelettes permet, par l'utilisation d'un couple de filtres appliqués successivement, de séparer les bandes de fréquences d'un plan.
Le calcul d'une transformée en ondelettes introduit un sous-échantillonnage.

Nous avons utilisé l'ondelette de Haar qui correspond au couple de filtres [1,1] et [1,-1].
La transformation en ondelettes s'effectue de la manière suivante :

1. filtrage horizontal passe-bas [1,1],
   avec sous-échantillonnage par 2, forme le plan L
2. filtrage horizontal passe-haut [1,-1],
   avec sous-échantillonnage par 2, forme le plan H
3. filtrage vertical passe-bas et passe-haut sur ces nouveaux plans avec sous-échantillonnage par 2,
   forme les plans LL, HL, LH et HH

Les 4 plans ainsi obtenus représentent : les basses fréquences pour LL, les détails horizontaux pour LH, les détails verticaux pour HL ainsi que les détails diagonaux pour HH. Dans le cas d'un niveau de transformée en ondelettes supérieur à 1, on applique de nouveau cet algorithme sur le plan LL.

Nous avons créé les fonctions suivantes :

imgdwt	Variables d'entrée : `X`: *Plan à traiter «double».* `nom_ondelette`: *Ondelette «chaîne».* Variables de sortie : `LL`: *Plan "passe-bas" «double».* `HL`: *Plan détails "verticaux" «double».* `LH`: *Plan détails "horizontaux" «double».* `HH`: *Plan détails "diagonaux" «double».*
iimgdwt	Variables d'entrée : `LL`: *Plan "passe-bas" «double».* `HL`: *Plan détails "verticaux" «double».* `LH`: *Plan détails "horizontaux" «double».* `HH`: *Plan détails "diagonaux" «double».* `nom_ondelette`: *Ondelette «chaîne».* Variable de sortie : `YCBCR`: *Image YCbCr de classe «double».*

Les plans sont ensuite traités de manières différentes :
le dernier plan LL subit un codage DPCM, tandis que les plans HLn, LHn et HHn sont quantifiés puis seuillés.

2.4 Codage DPCM

Le codage Differential Pulse Code Modulation permet de compresser de manière intelligente un plan comportant des valeurs homogènes.

On considère ici que dans le plan LL (l'information homogène de l'image), la valeur d'un pixel est fortement corrélée à celles de ses voisins passé au travers d'un prédicteur. Le prédicteur est une simple matrice de coefficients permettant de pondérer les pixels voisins du pixel à coder.
L'erreur obtenue entre le plan d'origine et le plan prédit est alors transmise, et le gain de cette solution repose sur le fait que l'on peut de manière générale coder cette erreur sur un nombre bien inférieur de bits par pixel.

Il existe des versions du codeur plus complexes, dans lesquelles le pixel n'est pas codé en fonction de la valeur de ses voisins mais des valeurs décodées de ses voisins. Ceci implique de simuler le fonctionnement du décodeur dans la boucle. Ici nous avons choisi de n'utiliser qu'un codeur simple. Il suffit alors simplement de transmettre le plan d'erreur et le prédicteur qui a servi au codage.

Nous avons utilisé le prédicteur et les fonctions suivantes :

1/4	1/4	1/4
1/4	X	

dpcm	Variable d'entrée : `X`: *Plan à traiter «double».* `predicteur`: *Matrice du prédicteur «entier»[4].* Variable de sortie : `X_DPCM`: *Plan erreur «double».*
idpcm	Variable d'entrée : `X`: *Plan erreur «double».* `predicteur`: *Matrice du prédicteur «entier»[4].* Variable de sortie : `XPRIM`: *Plan décodé «double».*

2.5 Quantification scalaire et seuillage

Les plans contenant des fréquences plus élevées HLn, LHn et HHn sont quantifiés par extension des valeurs sur 16 bits (à l'aide d'un coefficient multiplicatif de mise à l'échelle), les valeurs sont ensuite seuillées. Celles inférieures en valeur absolue au seuil sont mises à zéro.

Nous avons créé les fonctions suivantes :

quantificationHF	Variables d'entrée : `X`: *Plan à traiter «double».* `seuil`: *Pourcentage du seuil «entier».* Variables de sortie : `X_quantifie`: *Plan traité «double».* `delta`: *Coefficients «double».*
iquantificationHF	Variables d'entrée : `YCBCR`: *Image YCbCr «double».* Variables de sortie : `RGB`: *Image RGB «double».*

Les données sont maintenant prêtes à être transformées en un train binaire.

2.6 Ecriture du fichier

Les valeurs sont écrites dans un fichier par la fonction *save* de Matlab, celles-ci seront au format entier signé sur 16 bits.

2.6.1 En-tête

Pour recréer l'image, les fonctions de décompression ont besoin de connaître ces différents paramètres : les dimensions d'une tuile, le nombre de tuiles utilisées, la profondeur de transformation en ondelettes, les coefficients de la quantification ainsi que le prédicteur du codage DPCM.
On transmet l'inverse de chaque coefficient du prédicteur, car le format des données du fichier est sur 16 bits, or les coefficients de pondération ne sont pas des entiers mais des fractions (ici 1/4).

Nous avons choisi de répartir les informations de la manière suivante :

En-tête :

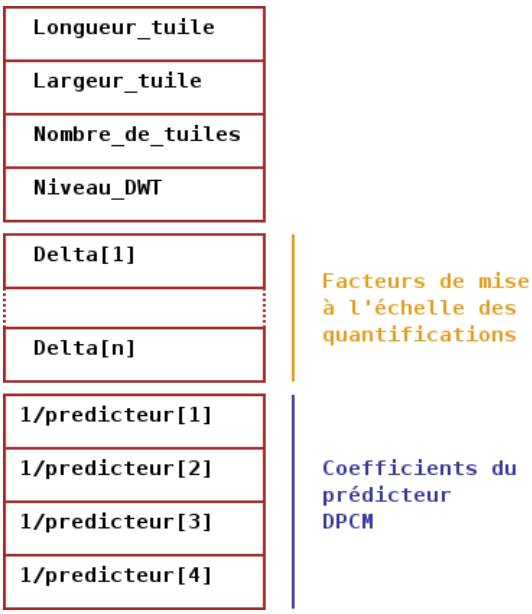


FIG. 2: En-tête du fichier

2.6.2 Données de l'image

Le reste de la structure était imposé dans le guide du mini-projet : pour chacune des tuiles, on transmet les plans dans l'ordre Y, Cb, Cr, avec en premier la composante LL, puis HLn+1, LHn+1, HHn+1, puis HLn, LHn, HHn. Corps du fichier :

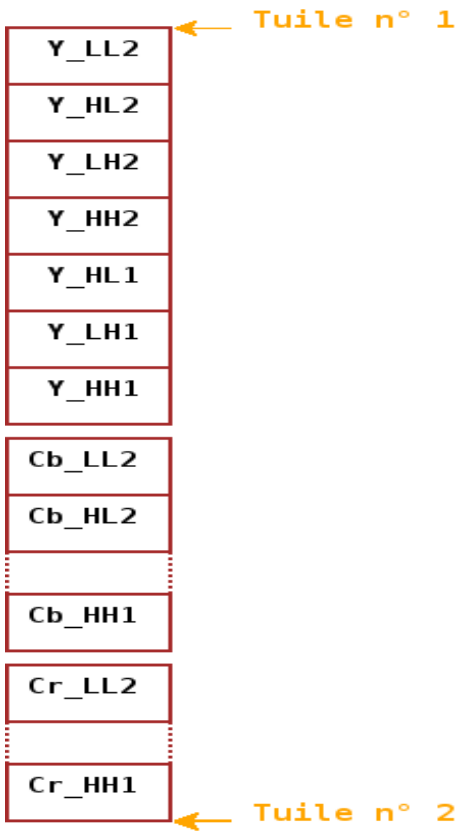


FIG. 3: Enregistrement des plans

2.6.3 Codage RLE et Huffman

Le train de données ainsi constitué de l'en-tête et des données passe ensuite par une compression RLE suivie d'une compression d'algorithme dit de Huffman.

Le codage Run Length Encoding permet de compresser efficacement les répétitions de valeurs dans un train de données. Il remplace les séquences de répétition par la valeur et le nombre d'occurences de celles-ci. Ce système est très efficace par exemple sur les zones uniformes de notre image, car le seuillage et le codage prédictif permettent de créer d'importantes séquences de zéros, et celles-ci peuvent être remplacées par une séquence RLE équivalente qui ne nécessite que quelques octets.

Le codage Huffman est une approche différente : ici le train est analysé dans son ensemble et l'on réalise des statistiques sur les fréquences d'apparitions des différentes valeurs. On cherche à exprimer sur un nombre de bits minimum les valeurs revenant le plus régulièrement, pour ce faire l'algorithme crée un arbre qui permet une lecture unique des données codées et ce malgré que le codage des données se fasse à longueur variable.

Ici nous avons fait appel à de simples exécutables externes pour assurer le codage et le décodage.

2.7 Décompression

La décompression s'effectue de manière symétrique, selon les étapes suivantes :

1. Décodage Huffman
2. Décodage RLE
3. Recréation de l'image LL codée par DPCM
4. Déquantification des plans HL, LH et HH
5. Transformation en ondelettes inverse
   Lors de la décompression, la transformation inverse est exactement symétrique, le sous-échantillonnage est remplacé par des insertions de valeurs nulles (zero-padding).

3 Evaluation des résultats obtenus

3.1 Critères d'analyses des résultats

Après avoir réaliser la compression et la décompression d'une image RVB, nous devons nous assurer de la fiabilité des méthodes employées. Pour cela, nous avons fait plusieurs tests qui prédisent la qualité de la compression.
Plusieurs paramètres sont pris en compte parmi lesquels :

- le paramètre de qualité (seuil de quantification)
- le choix du nombre de transformer en ondelettes(DWT)
- le nombre de tuiles
- la nature de l'image.

On utilise des critères objectifs qui conditionnent la qualité et la pertinence de la compression de l'image. Quatre critères sont choisis parmi lesquels :

- la proportion de coefficients nuls dans l'image quantifiée

  $TZ = \frac{Nbre\ de\ zeros\ contenus\ dans\ l'image}{Nbre\ de\ bits\ total}$

- le taux de compression :

  $TDC = \frac{Nbre\ de\ bits\ avant\ compression}{Nbre\ de\ bits\ aprs\ compression}$

- l'erreur quadratique moyenne(MSE) :

  $MSE = \frac{1}{MN} \sum_{k=1}^{M} [x(k,l) - \hat{x}(k,l)]^2$

- le PSNR :

  $PSNR = 10 \log_{10} \frac{2^{16}}{MSE}$

4 Interprétation des résultats

4.0.1 Influence du seuil

Nous avons choisi différentes images prises dans des scènes courantes (paysage, plage, montagne). Nous avons fixé tous les paramètres excepté le seuil de quantification, le faisant varier de 10 (bonne qualité) à 100 (compression maximum) avec un pas de 10.

- Taux de zéros et taux de compression

  Si on augmente le seuil, on peut prédire qu'il y aura un nombre de zéros croissant dans l'image. Plus nous avons de séquences identiques dans une image et plus les algorithmes de Huffman et RLE seront performants.

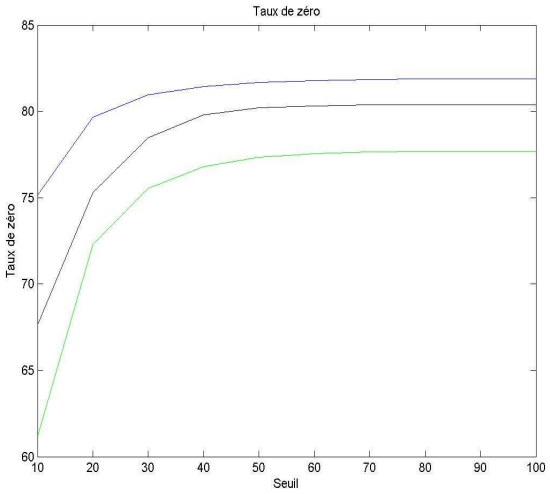


FIG. 4: Taux de zéros en fonction du seuil de quantification

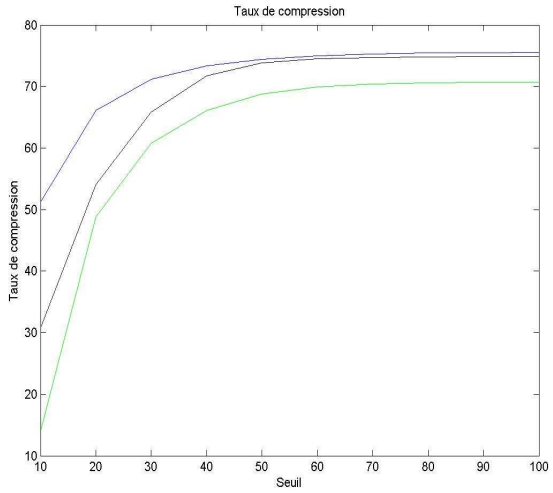


FIG. 5: Taux de compression en fonction du seuil de quantification

Les allures des 2 courbes ci-dessus sont quasiment identiques. En effet, lorsque le seuil de quantification augmente, des séquences de zéros sont créées, donc le taux de zéro augmente aussi, ce qui a pour conséquence l'augmentation du taux de compression. La prédiction faite et le résultat obtenu sont en corrélation avec ce que nous attendions.

- Erreur quadratique moyenne

L'erreur quadratique moyenne (MSE) indique la différence entre l'image de départ et l'image après la compression. Plus le seuil est élevé, plus l'erreur est grande.

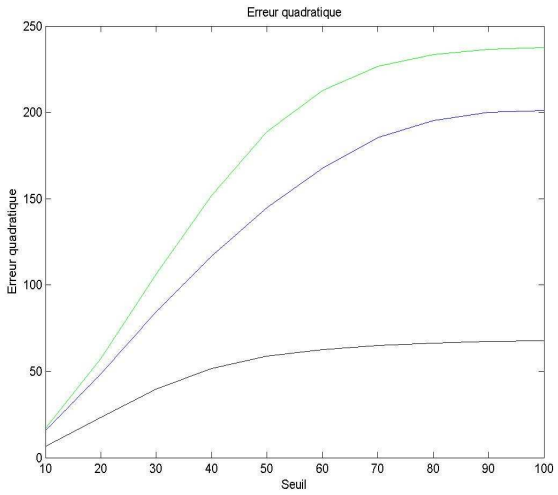


FIG. 6: Erreur Quadratique en fonction du seuil de quantification

Lorsque le seuil est faible, le MSE lui est proprotionnel et l'erreur est négligeable. Puis le MSE se stabilise à un certain seuil.

- PSNR

Le PSNR est un critère objectif, donnant une idée de la qualité de l'image compressée par rapport à celle d'origine. Plus il est élevé, plus l'image compressée est proche de l'image source.

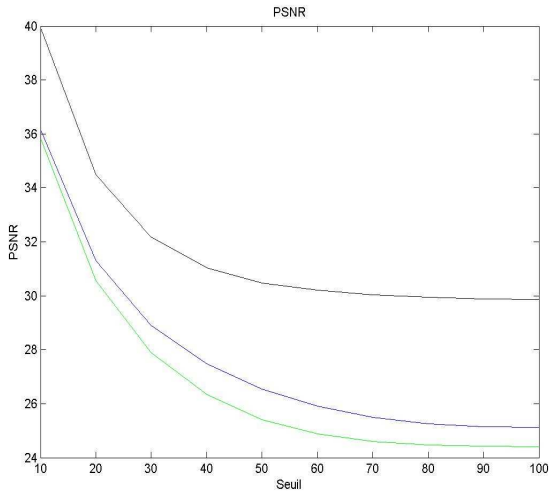


FIG. 7: PSNR en fonction du seuil de quantification

Le PSNR décroît donc lorsque le seuil augmente.

- Conclusion sur le seuil

Le seuil est le facteur primordial dans la qualité de l'image compressée. Pour une valeur de seuil comprise entre 20 et 30, on obtient un bon compromis entre la qualité de l'image et le taux de compression. Si l'on souhaite que le fichier compressé soit petit, il faut augmenter le seuil.

4.0.2 Influence du nombre de transformées en ondelettes(DWT)

Nous avons fixé tous les paramètres excepté le nombre de transformées en ondelettes (DWT), le faisant varier de 1 à 4 avec un pas de 1. De cette manière, on peut apercevoir l'influence de la profondeur de DWT sur la compression de l'image.

- Taux de zéro et de compression

  De même que pour le seuil, plus on augmente le nombre de transformées en ondelettes, plus les taux de compression et de zéros augmentent.

- Erreur quadratique moyenne

  Les valeurs des pixels de l'image compressée sont de plus en plus altérées lorsque le nombre de DWT augmente.

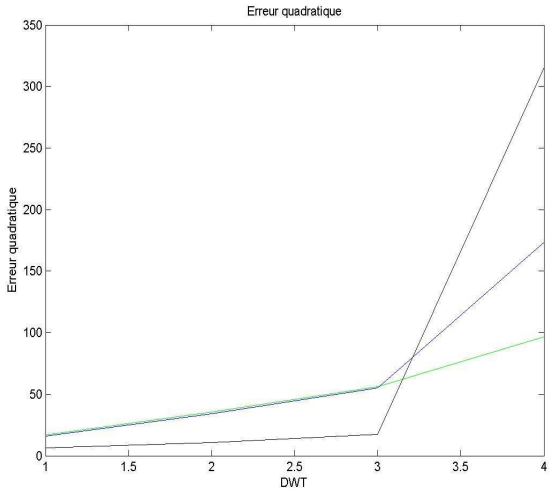


FIG. 8: Erreur Quadratique en fonction du nombre de DWT

  Jusqu'au niveau de 3 transformées en ondelettes, l'erreur entre l'image d'origine et celle après compression augmente faiblement. Cependant lorsque l'on passe à 4, l'erreur est au minimum triplée.

- PSNR

  Pour l'ensemble de ces images, plus on augmente le nombre de DWT et plus l'image se dégrade pour un seuil donné.

- Conclusion sur le nombre de DWT

  Pour un nombre de DWT de 2 ou 3, la qualité et le taux de compression de l'image sont corrects.

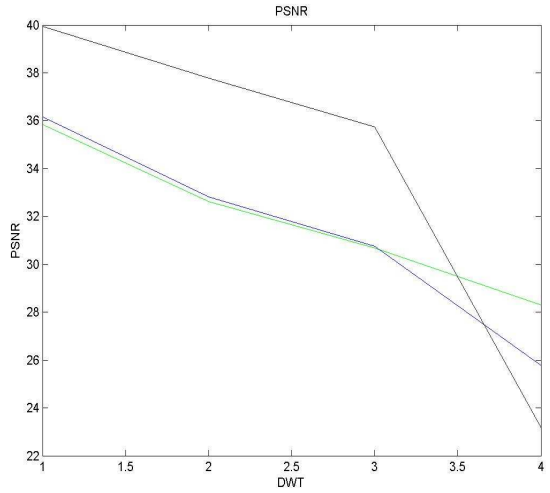


FIG. 9: PSNR en fonction du nombre de DWT

4.0.3 Influence du nombre de tuiles

Nous avons fixé l'ensemble des paramètres excepté le nombre de tuiles, le faisant varier de 1 à 64. On découpe l'image en plusieurs tuiles, qui sont traitées individuellement, et ce dans le but d'optimiser la quantification en l'appliquant sur des zones plus uniformes.

- Taux de zéros et taux de compression

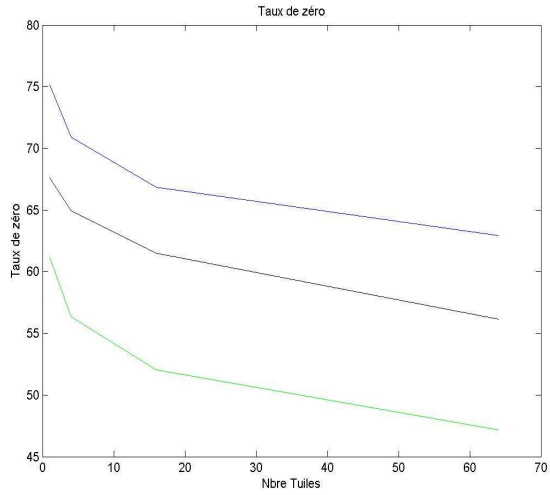


FIG. 10: Taux de zéros en fonction du nombre de tuiles

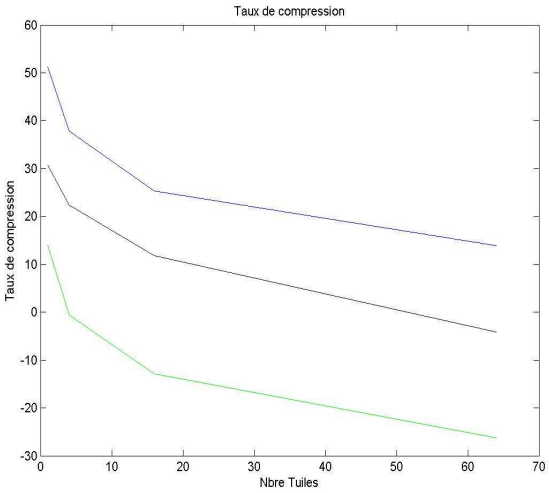


FIG. 11: Taux de compression en fonction du nombre de tuiles

Sur les images que nous avons utilisées, le gain apporté par un découpage en tuiles n'a pas été réellement manifeste, le découpage a même eu plus tendance à alourdir le fichier compressé en baissant le taux de zéros dans l'image. Ceci est dû au fait qu'aucune des tuiles de nos images n'était réellement à tendance uniforme.

- Erreur quadratique moyenne

Grâce à la mesure de MSE, on observe que plus l'image est découpée en un nombre important de tuiles, et plus l'erreur entre l'image d'origine et celle compressée est faible.

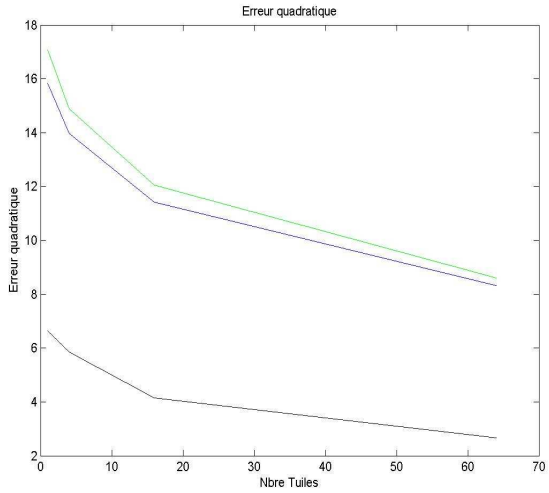


FIG. 12: Erreur Quadratique en fonction du nombre de tuiles

- PSNR

L'observation du PSNR nous amène à la même conclusion.

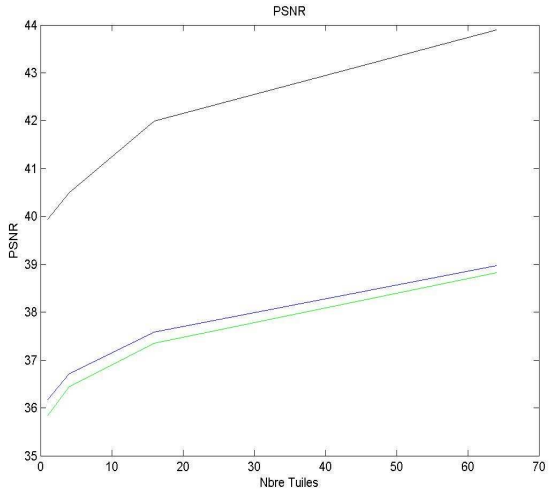


FIG. 13: PSNR en fonction du nombre de tuiles

- Conclusion sur le nombre de tuiles

Pour des paramètres fixes, un découpage en tuiles permet de conserver une meilleure qualité d'image, mais dans notre cas, ceci est principalement dû au fait que la compression n'a pas été optimisée par le découpage, car les fichiers résultants sont plus volumineux. Il semble donc normal que la qualité de leur image soit plus élevée.

4.0.4 Influence de la nature de l'image

La manipulation consiste à déterminer l'influence de la nature de l'image. Nous avons sélectionné 3 images différentes : la première est plutôt une photo avec des valeurs relativement proches les unes des autres, la seconde est une photo normale de paysage et enfin la dernière est très contrastée (c'est-à-dire avec des écarts de valeurs élevés).

- Taux de zéro et taux de compression

Les algorithmes de compression sont plus performants lorsque les zones sont homogènes.

TAB. 1: Taux de zéros

nature	sombre	normale	clair
Taux de Zéros	90.4	66.2	91.1

TAB. 2: Taux de compression

nature	sombre	normale	clair
Taux de compression	75.3	2.9	74.7

Le nombre de zéros ainsi que le taux de compression sont meilleurs pour une photo aux valeurs proches.

- Erreur quadratique moyenne

  Les résultats sont très différents : l'erreur est minimale pour l'image homogène, tandis qu'elle est très élevée pour l'image fortement contrastée.

TAB. 3: Erreur quadratique

nature	sombre	normale	clair
Erreur quadratique	2.1	30.1	46.3

- PSNR

  Plus l'image a des valeurs proches les unes des autres et plus sa restitution est fidèle.

TAB. 4: PSNR

nature	sombre	normale	clair
PSNR	44.9	33.4	31.5

5 Corrélation entre les différents critères objectifs

5.1 Relations entre le taux de compression et le Taux de zéros

On sait que le nombre de zéros et le taux de compression sont intimement liés, car notre technique repose en grande partie sur la création de zones importantes de zéros grâce au codage DPCM et à la quantification.
Pour pouvoir mieux apprécier le lien entre ces deux mesures, nous avons tracé la courbe liant ces deux paramètres.

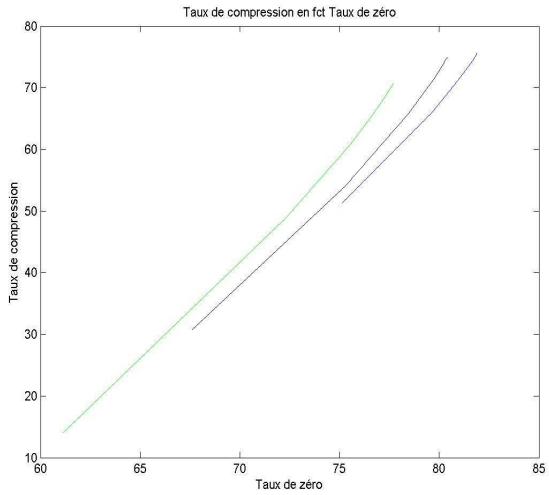


FIG. 14: Taux de compression en fonction du nombre de zéros dans l'image

On peut constater que ces deux mesures sont proportionnelles. Plus l'image a de zéros et meilleur est le taux de compression, et ce indépendamment des autres réglages.

5.2 Relations PSNR / Taux de zéros et PSNR / Taux de compression

Le PSNR permet d'évaluer la ressemblance entre l'image originale et celle compressée, plus il est fort et plus les images sont proches. A l'inverse, un faible PSNR indique qu'une partie de l'information de l'image est perdue.

5.3 Influence du seuil et du nombre de tuiles

- PSNR en fonction du Taux de zéros

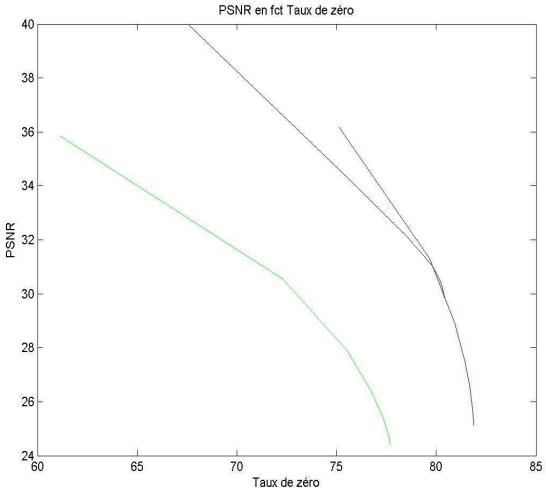


FIG. 15: PSNR en fonction du nombre de zéros dans l'image

- PSNR en fonction du Taux de compression

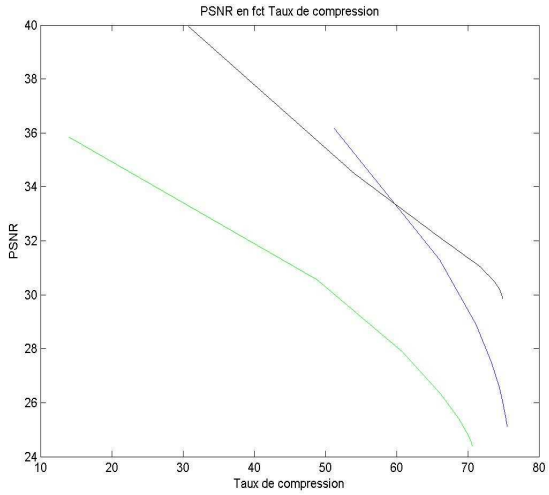


FIG. 16: PSNR en fonction du Taux de compression

L'examen du PSNR nous confirme que plus l'image est compressée, plus elle s'éloigne de l'image d'origine.

5.4 Conclusion

Les résultats de la compression dépendent en grande partie des réglages des différents niveaux de la chaîne, ceux-ci peuvent être fixés selon l'utilisation de l'image.

Pour compresser une image en utilisant notre programme, nous avons retenu les paramètres suivants :

Qualité	*Seuil*	*Nombre de DWT*	*Nombre de tuiles*
Basse	> 30	> 3	1 ou 4
Moyenne	25	2 ou 3	1 ou 4
Haute	< 20	< 2	16

6 Bilan

Nous avons beaucoup apprécié de pouvoir travailler à notre rythme, en reconstruisant progressivement la chaîne de compression / décompression, tout en testant au fur-et-à-mesure de l'ajout de nouveaux blocs.
Nous avons aussi pu découvrir en détails le fonctionnement de la transformation en ondelettes, du codage DPCM et de la quantification. Ce mini-projet nous a permis de pouvoir prendre en main correctement Matlab et de nous familiariser avec les problèmes que pose la représentation d'images sur informatique.
Enfin, nous avons pu revenir de manière concrète sur le cours de traitement d'image, en approfondissant les notions qui y avaient été abordées.