**IMT Atlantique**
Filière par apprentissage
655 Avenue du Technopôle
29280 Plouzané

# Projet :

## Compression Vidéo

| | |
|---|---|
| **Groupe :** | NGUYEN Binh Minh<br>GONG Xiang<br>SANTOS SEISDEDOS Carlos |
| **Formation :** | FISE 2A – UE G : Compression de données : du codage de sources à la réalité virtuelle |
| **Date :** | 31 Mars 2020 |
| **Lieu :** | IMT Atlantique (Brest) |
| **Encadrant :** | DUPRAZ Elsa |

**IMT Atlantique**
Bretagne-Pays de la Loire
École Mines-Télécom

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

# Sommaire

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

# Contexte

Le but principal de ce projet c'est de **mettre en œuvre une solution de compression et décompression vidéo**. Afin d'organiser notre code MATLAB, nous avons créé un dépôt git en ligne sur GitHub.

Vous trouverez nos fichiers dans l'adresse suivante : github.com/chletes/IMTA-CD-Video_Compression_Project.

Dans le dépôt GitHub, vous trouverez trois dossiers :

- Le dossier /code/, qui contient les *scripts*, commençant par `'s_'`, et les fonctions, commençant par `'f_'`, que nous avons codé pour la réalisation de ce projet.
- Le dossier /data/images/, qui contient les images et les vidéos que nous utilisons pour faire les démonstrations (donc, utilisés par les *scripts*).
- Le dossier /ressources/, qui contient l'énoncé du projet (`'project_description.pdf'`) et autres dossiers.
    - Le dossier /ressources/biblio/ contient des informations que nous avons utilisé pour comprendre l'algorithme de compression JPEG ou l'algorithme d'estimation de mouvement, entre autres.
    - Les dossiers /ressources/BlockMatchingAlgoMPEG et /ressources/video_and_code contiennent des fonctions externes (pas à nous).
    - Les dossiers /ressources/TP1_Lossless_Coding et /ressources/TP2_Lossy_Source_Coding contiennent des fonctions dont nous avons codé pour les Travaux Pratiques 1 et 2 de l'UE.

Vous trouverez, dans la suite de ce document, des explications sur notre projet et des explications sur les fichiers les plus importants – *scripts* et fonctions – ainsi que des **performances**, en termes de temps écoulé, de taux de compression et de taux de distorsion, **de notre compression**.

Dans un premier temps, vous trouverez des explications sur le codage indépendant d'images suivant l'algorithme JPEG, ainsi que des mesures de performance.

Dans un deuxième temps, nous avons mis en œuvre une **solution complète pour encoder et décoder les images de la vidéo indépendamment les unes des autres**.

Dans un troisième temps, nous avons mis en œuvre un **codage prédictif pour encoder et décoder les images successives d'un vidéo les unes par rapport aux autres**.

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

# Codage indépendant d'images suivant l'algorithme JPEG et mesure des performances.

- Petite introduction sur l'algorithme JPEG.

L'algorithme que nous avons choisi pour le codage et le décodage d'images indépendantes c'est **l'algorithme de compression JPEG**. Il s'agit d'un algorithme de compression avec perte utilisé pour réduire la taille des fichiers d'images. Cela signifie qu'en décompressant ou en affichant l'image, vous n'obtenez pas exactement la même image qu'avant la compression. La compression avec perte ne convient pas aux images ou aux graphiques dont le texte, les lignes ou les bordures sont très nettes, mais elle est adaptée aux fichiers qui contiennent de grandes zones de couleur unie.

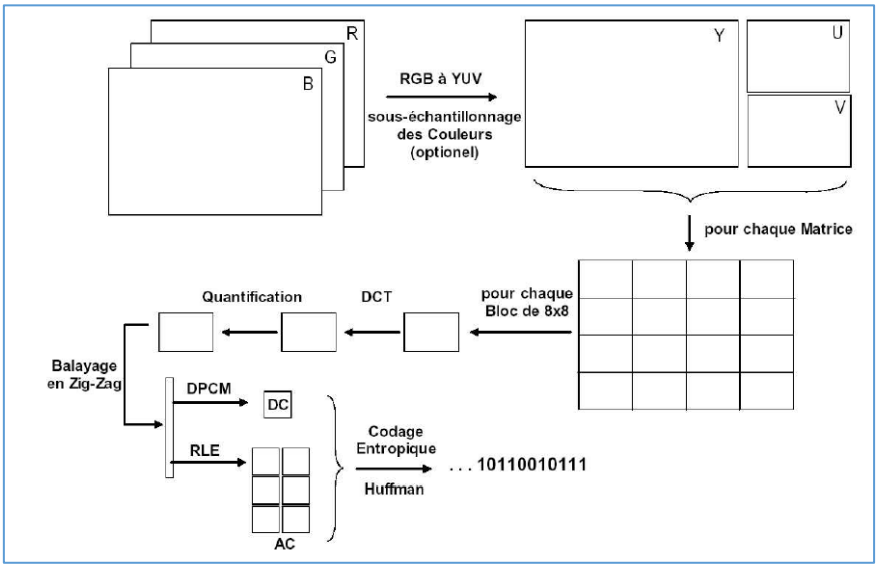
L'algorithme JPEG est basé sur deux phénomènes visuels de l'œil humain :

1. L'œil humain est beaucoup plus sensible aux changements de luminance que de chrominance, c'est-à-dire qu'il capte les changements de luminosité plus clairement que les changements de couleur.
2. L'œil humain détecte plus facilement les petites variations de luminosité dans les zones homogènes que dans les zones où la variation est importante, par exemple sur les bords des corps des objets.

Une des caractéristiques de JPEG est la possibilité d'ajuster le degré de compression. En théorie, **un taux de compression très élevé se traduira par une taille de fichier réduite, au prix d'une perte de qualité importante.** Au contraire, **un faible taux de compression produira une qualité d'image très similaire à celle de l'original, mais la taille du fichier sera plus importante.**

- Description de l'algorithme suivi.

Pour coder l'algorithme JPEG, nous avons suivi la suivante image, issue du document 'C1_C2_TraitementMonomedia.pdf' que vous pouvez trouver dans le dossier /ressources/biblio.



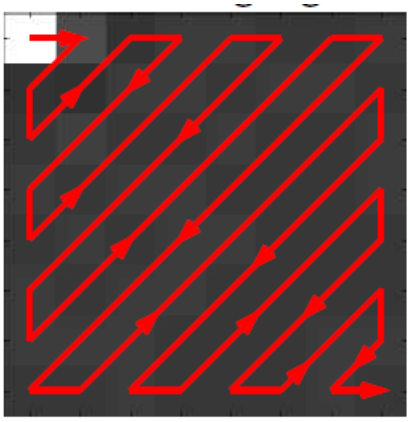

En effet, l'algorithme suit ces étapes :

1. Nous transformons l'image dans l'espace de couleur YUV. Y représente la luminance relative, et les deux autres, U et V, représentent la chrominance. En MATLAB, ces composantes sont représentées en sous forme de matrices.
2. Nous décomposons chaque composante en blocs de taille 8x8 et chaque bloc de taille 8x8 suit les suivantes étapes :
   i. Nous calculons la transformée en cosinus discrète (de l'anglais : DCT ou Discrete Cosine Transform) de chaque bloc pour obtenir les coefficients DCT de chaque bloc.
   ii. Nous pondérons les coefficients DCT de chaque bloc l'aide d'une table de quantification que prévoit la norme JPEG et qui dépend de la qualité désirée de l'image comprimée. Voici la table de quantification que prévoit la norme JPEG pour obtenir une qualité $Q$ égale à 50 :

$$Q_{50} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

   À partir de cette matrice de quantification, nous pouvons calculer d'autres matrices pour d'autres qualités $Q$.

iii. Grâce à un balayage en zig-zag, indiquée dans la suivante image, nous séparons le premier coefficient de chaque bloc quantifié – appelé composante continue ou DC – des restes des coefficients. Cela est fait parce qu'un traitement (un codage) spécifique est réservé pour eux : le codage prédictif DPCM (de l'anglais *Differential Pulse Code Modulation*). Le reste de coefficients – appelés composants AC – sont associés entre eux par un codage RLE (de l'anglais *Run-Length Encoding*)



iv. Pour finir avec la compression, le codage entropique de Huffman est utilisé pour les 64 coefficients du bloc – soit les coefficients DC et AC.

Pour décoder les images encodées, nous avons suivi les étapes de l'algorithme de compression JPEG dans l'ordre inverse.

- Description de notre algorithme.

Afin de pouvoir faire une démo de notre algorithme, vous trouverez le *script* `s_image_compressing.m` dans le dossier /code/ du dépôt GitHub. Nous avons aussi créé la suivante carte mentale pour vous expliquer comment nous avons codé la compression et la décompression d'une image, et notamment pour suivre le *script* d'une manière plus claire. La partie gauche de la carte mentale permet la compression d'une image ; et la partie droite permet la décompression de l'image.

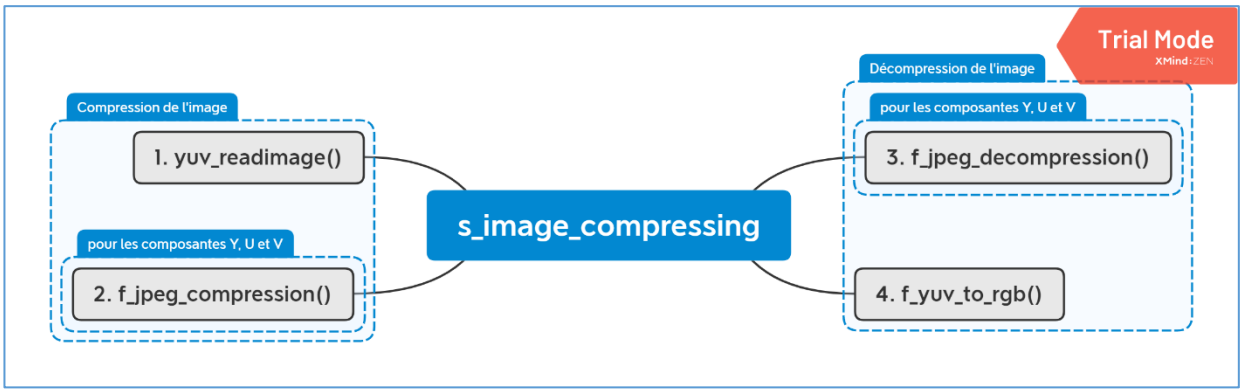



Figure 1. Carte mentale du *script* `s_image_compressing.m`

Quant à la compression d'une image, après la vérification de l'existence du fichier (ou de l'image), le *script* s'appuie sur la fonction `yuv_readimage()`, disponible dans le dossier /ressources/video_and_code, pour obtenir les 3 composantes de l'espace de couleurs YUV. Ensuite, la fonction `f_jpeg_compression()` est appelé pour obtenir chaque composante comprimée. En effet, c'est la fonction `f_jpeg_compression()` celle qui met en œuvre l'algorithme de compression JPEG, expliqué dans la section d'avant.

Et quant à la décompression de l'image, la fonction `f_jpeg_decompression()` est appelé pour obtenir chaque composante décomprimée. C'est cette fonction là celle qui met en œuvre l'algorithme de décompression JPEG. Ensuite, la fonction `f_yuv_to_rgb()` permet de transformer les trois composantes YUV à l'espace de couleurs RGB, grâce à ces formules :

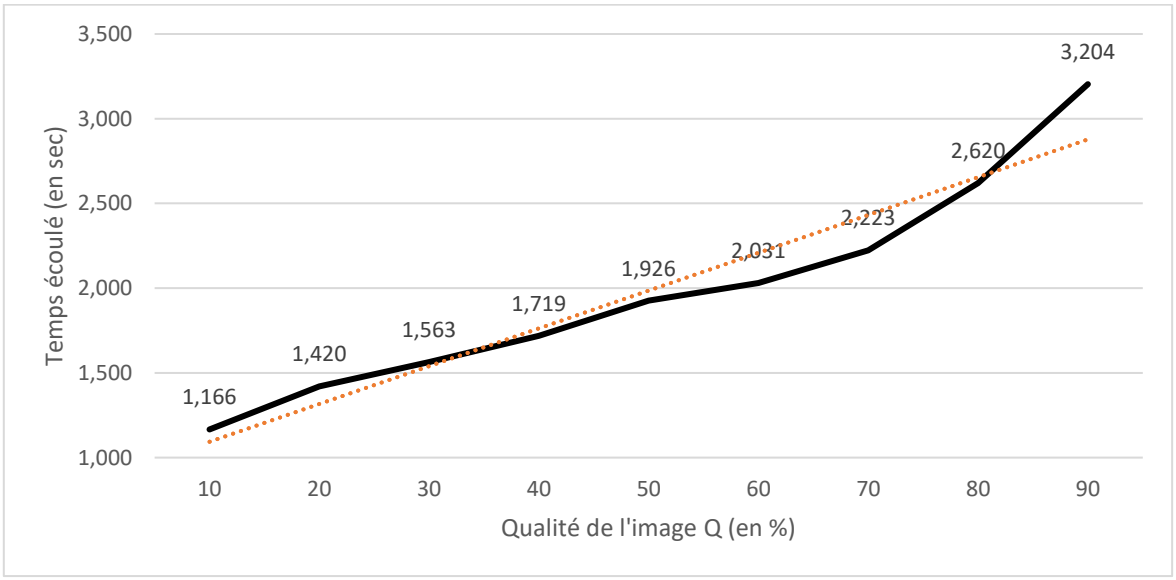
$$R \leftarrow Y + 1{,}403 \cdot (U - 128)$$

$$G \leftarrow Y - 0{,}714 \cdot (U - 128) - 0{,}344 \cdot (V - 128)$$

$$B \leftarrow Y + 1{,}773 \cdot (V - 128)$$

Quant aux fonctions `f_jpeg_compression()` et `f_jpeg_decompression()`, nous recommandons jeter un coup d'œil au code directement (nous avons mis beaucoup de commentaires pour l'expliquer).

Quant à la performance de notre code pour la compression et la décompression d'une image, nous avons mesuré le temps écoulé pour des qualités allant de $Q = 10\%$ à $Q = 90\%$. Le suivant graphique montre la moyenne du temps écoulé pour des qualités allant de $Q = 10\%$ à $Q = 90\%$. Nous pouvons donc distinguer une tendance croissante, montrée par la ligne en pointillés rouges, du temps de compression et décompression d'une image avec la qualité de l'image demandée. Nous pouvons conclure que plus on demande une qualité élevée, plus de temps l'algorithme met.




Moyenne du temps écoulé lors de la compression et décompression JPEG d'une image pour des qualités allant de $Q = 10\%$ à $Q = 90\%$.

Quant à la distorsion d'une image avant la compression et après la décompression, nous avons mesuré PSNR pour des qualités allant de $Q = 10\%$ à $Q = 90\%$ d'après les suivantes formules :

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right)$$

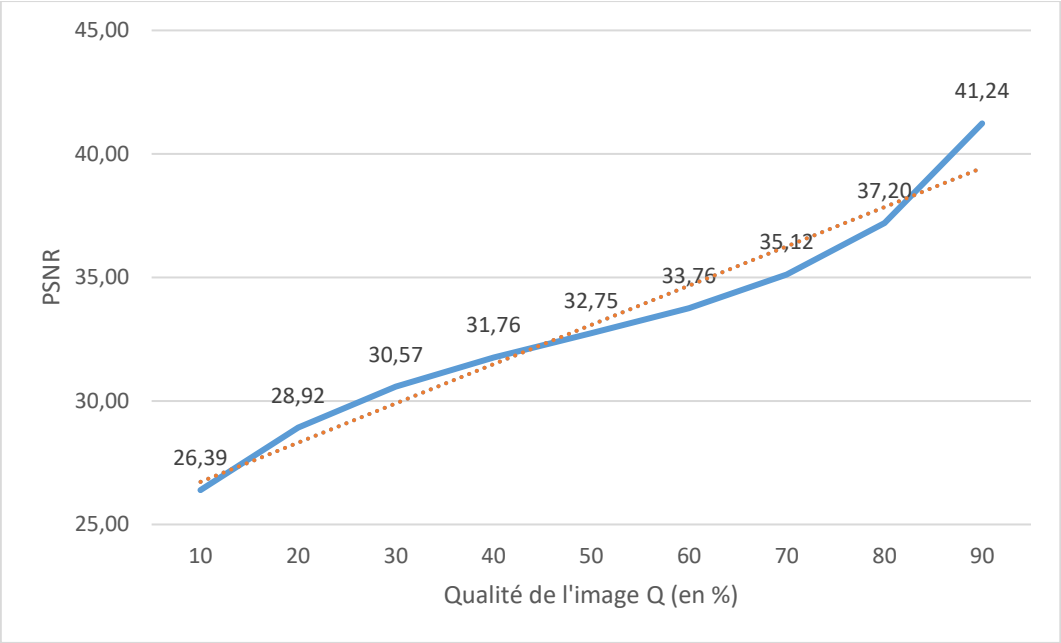
Où, $MAX_I^2$ représente la valeur maximale de I, soit l'image avant compression, et

$$MSE = \frac{1}{m \cdot n} \cdot \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2$$

Où,

- $I$ représente l'image avant compression, et donc $I(i,j)$ représente le pixel $(i,j)$ de l'image sans compression,
- $K$ représente l'image après décompression, et donc $K(i,j)$ représente le pixel $(i,j)$ de l'image décomprimée,
- $m$ et $n$ représentent la taille de l'image (nombre de pixels en colonne et nombre de pixels en ligne).

Le suivant graphique montre la variation du PSNR pour des qualités allant de $Q = 10\%$ à $Q = 90\%$. Nous pouvons donc distinguer une tendance croissante du PSNR, montrée par la ligne en pointillés rouges, avec la qualité de l'image demandée. Nous pouvons conclure que plus on demande une qualité meilleure, plus le PSNR est élevé, comme c'est normal.



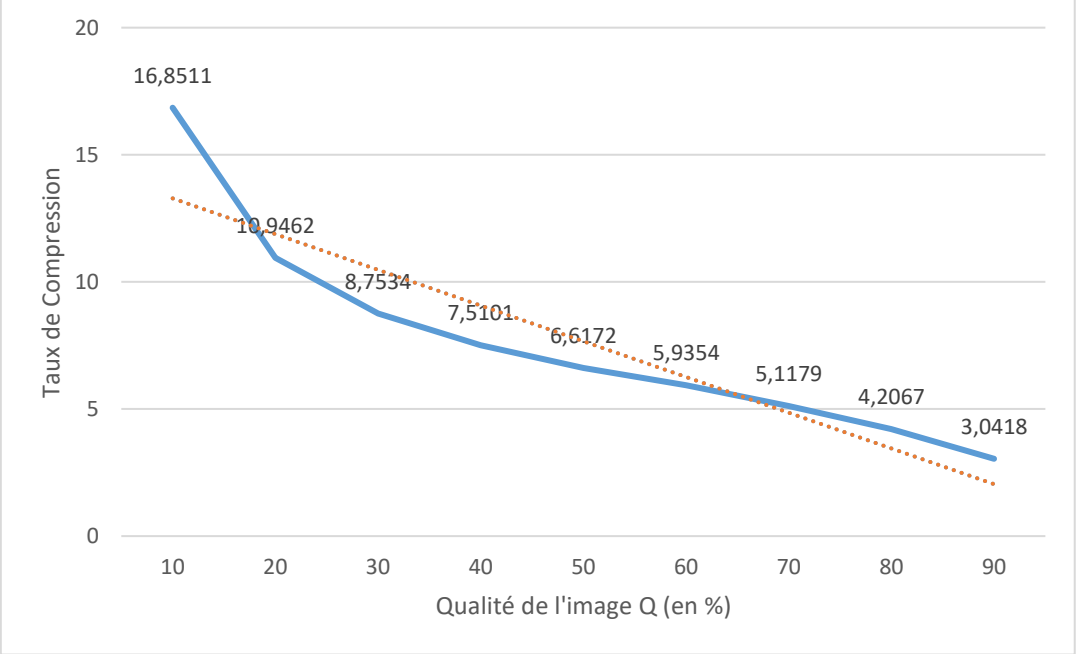

PSNR d'une image pour des qualités allant de $Q = 10\%$ à $Q = 90\%$.

Quant au taux de compression de notre algorithme, nous avons mesuré cela d'après la suivante formule :

$$\eta = \frac{\text{taille de l'image avant compression}}{\text{taille de l'image après compression}}$$

Pour la taille de l'image avant compression, nous avons calculé la somme des tailles de chaque composante YUV avant compression; et pour la taille de l'image après compression, nous avons calculé la somme des tailles de chaque composante YUV après compression.

Le suivant graphique montre la variation du taux de compression pour des qualités allant de $Q = 10\%$ à $Q = 90\%$. Nous pouvons donc distinguer une tendance décroissante du taux de compression, montrée par la ligne en pointillés rouges, avec la qualité de l'image demandée. Nous pouvons conclure que plus on demande une qualité meilleure, moins l'algorithme est capable de comprimer l'image.




Taux de compression d'une image pour des qualités allant de $Q = 10\%$ à $Q = 90\%$.

Après ces tests de performance, nous avons pu vérifier qu'**un taux de compression très élevé se traduit par une taille de fichier réduite, au prix d'une perte de qualité importante**, comme avancé en théorie.

# Codage d'un vidéo suivant l'algorithme JPEG et mesures de per performances.

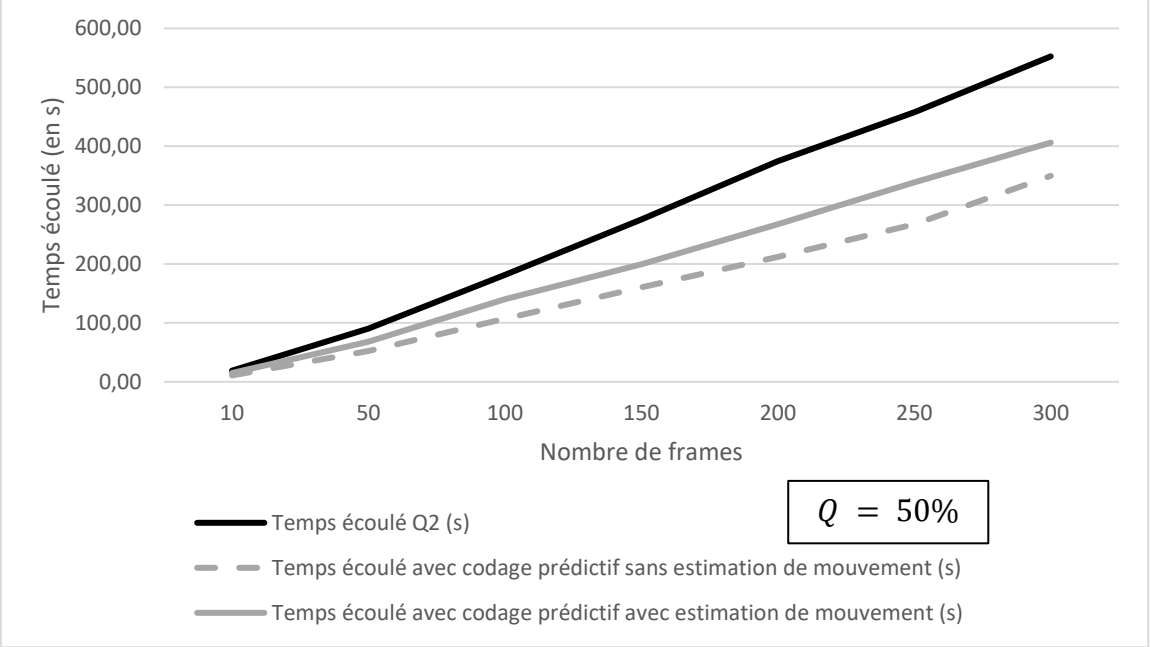
Nous avons mis en œuvre trois codages vidéo.

Le premier s'agit d'un codage indépendant des images suivant l'algorithme JPEG. Dans celui-ci, nous « lisons » chaque image de la vidéo et nous appliquons l'algorithme JPEG sur chaque image. Nous estimons que ce codage sera le codage le moins performant en tant que temps écoulé et taux de compression.

Le deuxième s'agit d'un codage prédictif. Dans cet exemple, les images sont prédites à partir de l'image précédente. Ensuite, la matrice résultante de la prédiction est ensuite codée avec l'algorithme JPEG. Même si nous appliquons une étape de plus, théoriquement, la matrice résultante a moins d'énergie que les valeurs originales des pixels et peut donc être codée avec moins de bits. De ce fait, nous estimons que ce codage sera meilleur par rapport au codage précèdent en termes de temps écoulé lors de la compression et la décompression et en termes de taux de compression, mais nous avons des doutes quant à la qualité visuelle.

Le troisième s'agit d'un codage prédictif dans lequel nous estimons les mouvements temporels grâce à l'exploitation de la corrélation spatiale. Le fait de tenir en compte les mouvements permettra d'obtenir un vidéo avec meilleure une qualité visuelle qu'avec le codage prédictif sans estimation de mouvement. Nous estimons aussi des performances pires en termes de temps écoulé lors de la compression et la décompression.

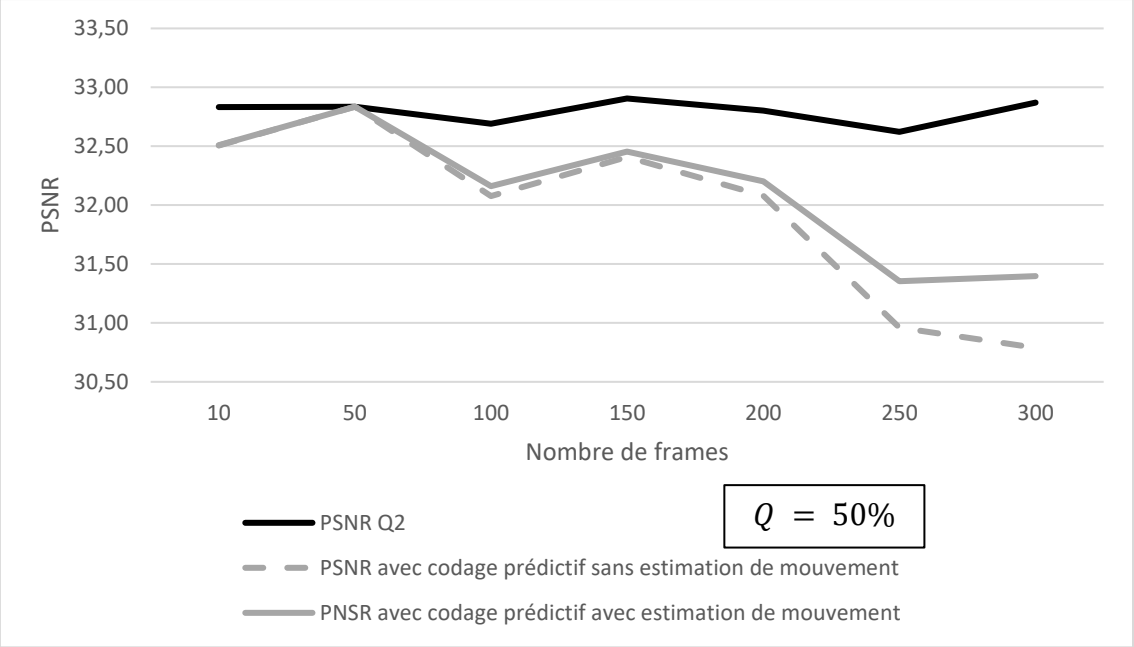
Pour mesurer la performance des trois codages lors de la compression et la décompression d'un vidéo, nous avons mesuré le temps écoulé pour une qualité égale à 50% et un nombre de frames allant de 10 à 300. Le suivant graphique montre la moyenne du temps écoulé pour une qualité égale à 50% et un nombre de frames allant de 10 à 300 pour les trois codes.




Moyenne du temps écoulé lors de la compression et décompression d'un vidéo pour une qualité $Q = 50\%$ et un nombre de frames allant de 10 à 300.

Nous pouvons donc voir, que quant au temps écoulé pour la compression et la décompression d'un vidéo, le codage le plus performant est le codage prédictif sans estimation de mouvement. En effet, même si pour un nombre de frames petit, le temps écoulé lors de la compression et la décompression d'un vidéo est similaire pour les trois codages ; lorsque le nombre de frames augmente, nous trouvons que le codage le plus rapide est le codage prédictif sans estimation de mouvement et, au contraire, le codage le plus lent est le codage indépendant des images suivant uniquement l'algorithme JPEG.

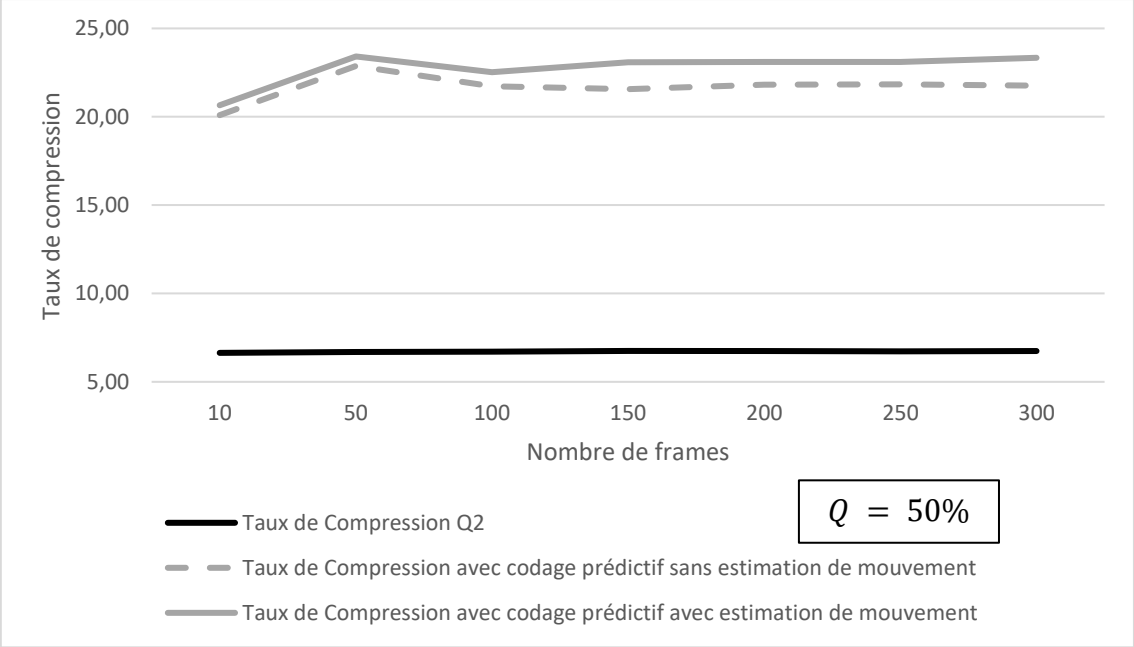
Nous avons mesuré aussi le PSNR des images d'un vidéo avant et après compression, pour une qualité égale à 50% et un nombre de frames allant de 10 à 300. Le suivant graphique montre la moyenne du PSNR pour une qualité égale à 50% et un nombre de frames allant de 10 à 300 pour les trois codes.




Moyenne du PSNR lors de la compression et décompression d'un vidéo pour une qualité $Q = 50\%$ et un nombre de frames allant de 10 à 300.

Nous pouvons donc voir, que quant au PSNR, le codage le plus performant est le codage indépendant des images suivant uniquement l'algorithme JPEG. Même si les autres codages restent assez près du PSNR moyen de ce codage, le graphique met en évidence que le PSNR des codages prédictifs a une tendance décroissante avec l'augmentation du nombre de frames.

Et finalement, nous avons mesuré aussi le taux de compression de la vidéo lors de sa compression, pour une qualité égale à 50% et un nombre de frames allant de 10 à 300 aussi. Le suivant graphique montre le taux de compression d'un vidéo pour une qualité égale à 50% et un nombre de frames allant de 10 à 300 pour les trois codes.




Taux de compression $\mu$ lors de la compression et décompression d'un vidéo pour une qualité $Q = 50\%$ et un nombre de frames allant de 10 à 300.

Nous pouvons donc voir, que quant au taux de compression, le codage le plus performant est le codage prédictif avec estimation de mouvement, suivi du codage prédictif sans estimation de mouvement. Le graphique met en évidence la mauvaise performance du codage indépendant des images suivant uniquement l'algorithme JPEG en tant que taux de compression.

Vous pouvez trouver dans le dépôt GitHub un vidéo qui à subit la compression et la décompression des trois codages et comparer ainsi vous-mêmes la qualité visuelle.

## Conclusions

Les résultats obtenus pour la compression d'images avec l'algorithme JPEG vérifient les hypothèses théoriques qu'on faisait au début de ce rapport. En effet, **un taux de compression très élevé se traduira par une taille de fichier réduite, au prix d'une perte de qualité importante.**

Quant à la compression vidéo, les résultats comparant les trois codages nous servent de guide pour choisir un codage ou autre en fonction de nos besoins.

Personnellement, en regardant les vidéos après compression et décompression des trois codages, nous trouvons que le vidéo avec la meilleure qualité visuelle s'agit du vidéo qui subit le codage indépendant des images suivant uniquement l'algorithme JPEG, comme nous avançait le graphique du PSNR. Cependant, comme le reste de graphiques met en évidence, il s'agit du codage qui met le plus le temps à la compression et décompression du vidéo et le codage le moins performant en termes de taux de compression.

Le deuxième vidéo avec la meilleure qualité visuelle s'agit du vidéo qui subit le codage prédictif avec estimation de mouvement, comme nous avançait à nouveau le graphique du PSNR. En plus, il s'agit du codage le plus performant en termes de taux de compression.

Le codage prédictif sans estimation de mouvement est le plus performant en termes de temps écoulé pour la compression et décompression d'un vidéo. Cependant, personnellement et visuellement, nous trouvons des erreurs de codage assez souvent dans la vidéo, spécialement près de la danseuse.

En conséquence, si nous devions choisir un de ces trois codages pour compresser un vidéo, nous essayerions de définir une cible (la meilleure qualité visuelle possible, ou le taux de compression le plus élevée possible, par exemple) pour choisir le codage qui permet de l'atteindre. Voici quelques scénarios d'exemple :

- Si nous cherchions la meilleure qualité visuelle indépendamment de la taille du fichier compressé, nous utiliserions le codage indépendant des images suivant uniquement l'algorithme JPEG.
- Cependant, si nous cherchions une bonne qualité visuelle et une taille du fichier compressé (assez) réduite, c'est-à-dire si nous aurions un petit problème de stockage, nous utiliserions le codage prédictif avec estimation de mouvement.
- Et, si par exemple, nous cherchions à compresser un vidéo le plus rapide possible indépendamment de la qualité visuelle, nous utiliserions le codage prédictif sans estimation de mouvement.