IMT Atlantique
Technopôle de Brest-Iroise - CS 83818
29238 Brest Cedex 3
URL : **www.imt-atlantique.fr**

Rapport de projet

Rapport du projet de compression de données

Alexandre Lefebvre
Nicolas Malet
Marta Quiles Paramo

Date d'édition : 19 avril 2021
Version : 1.8

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

Sommaire

Résumé

Ce document présente les outils développés et les résultats obtenus à l'issue du projet de compression de données, dans le cadre du cours DATACOMP.

1. Format QCIF

Les vidéos utilisées pour les testes sont au format QCIF, avec les caractéristiques suivantes

channels : Y (luminance), U et V (chrominance)

dimensions : lxh = 176x144 pour Y, lxh = 88x72 pour U et V

fps : 25 images par secondes

taille.s^{-1} $25 \times \texttt{frame\_size} = 25 \times (176 \times 144 + 2 \times 88 \times 72) \approx 0.93 Mo.s^{-1}$

Des fonctions dans le fichier `video.py` permettent de lire et sauvegarder des vidéos au format qcif avec Python.

2. Codage entropique

2.1. Code de Huffman

Le code pour effectuer l'encodage et le décodage avec la méthode de Huffman est dans le fichier `./Huffman.py`. Connaisant les probabilités empiriques d'occurence $p(s)$ des symboles s de l'alphabet A, `[p(s) for s in A]`, l'algorithme pour trouver le code de Huffman est le suivant :

Algorithme de la méthode de Huffman

```
1. Calculer la probabilité empirique p(s) pour chaque symbole s.
2. Initialiser la liste l de groupes de symboles à [[s] for s in A].
3. Tant que len(l) != 1:
    S0 = l.pop(argmin [p(S) for S in l])
    S1 = l.pop(argmin [p(S) for S in l])
    Fusionner S0 et S1, l.append(S') avec S'=[S0,S1] et p(S')=S0+S1
4. Construire la table symbole/mot de code avec un parcours descendant de l.
```

2.1.1. Format du train binaire

Le train binaire comprend le minimum des symboles encodés, la table des probabilités empiriques et les symboles encodés :

- sign+uint8 : minimum des symbols $min(A)$ encodé au format uint8 avec un bit pour le signe (0 positif, 1 négatif).
- uint9 : longueur <n> de la table de probabilités, comprise entre 0 et 511 et égale à $\max(A) - \min(A) + 1$
- n x uint8 : probabilités empiriques pour chaque symbole, le premier symbole étant $\min(A)$ et le dernier $\max(A)$. Les probabilités sont multipliées par 255 et arrondis sur 8 bits.
- longueur variable : mots de code pour chaque symbole.

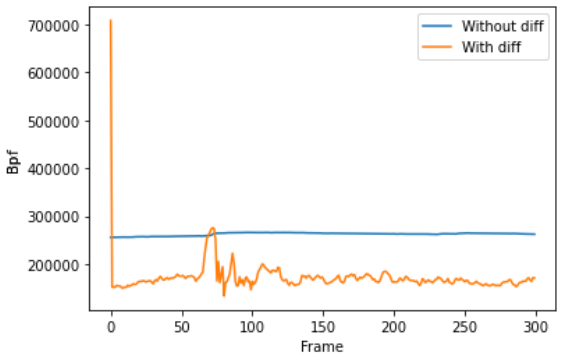
2.1.2. Résultats

Sur la figure 1 et dans le tableau 1, le codage entropique est comparé sur les données brutes des vidéos (sans différence d'image) et sur les images transformées avec $I'_0 = I_0$ et $I'_t = I_t - I_{t-1}, t > 0$ (voir 3.1).

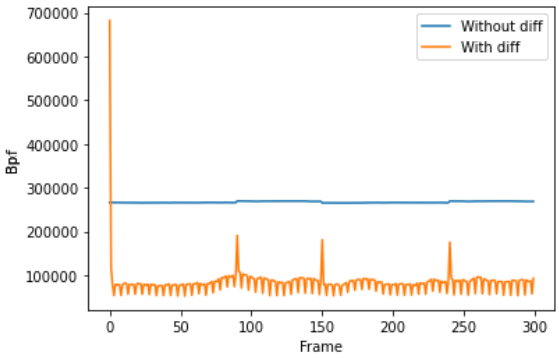
Les différences d'images permettent d'obtenir une compression plus grande, mais la lecture de la vidéo à n'importe qu'elle position dans le temps nécessite de tout décoder. Le débit binaire est aussi beaucoup plus bas en moyenne, mais la première image est une image de référence non transformée, avec des symboles peu utilisés et donc associés à un mot plus long. Il est également beaucoup plus stable lorsque les différences d'image ne sont pas utilisées puisque la taille de l'image compressée est

Vidéo	coastguard	hall	news	akiyo	carphone
Taille compressée (%)	86.3	86.9	87.9	89.2	88.2
Bit par symbole	6.9	6.9	7.0	7.1	7.0
Taille compressée (diff)	56.5	41.6	27.4	20.3	48.7
Bit par symbole (diff)	4.5	3.3	2.2	1.6	3.9

TABLE 1 – Résultats pour la méthode de Huffman



Bits par frame pour la vidéo 'coastguard'.



Bits par frame pour la vidéo 'news'.

FIGURE 1 – Comparaison des bits par frame avec et sans les différences d'images pour la méthode de Huffman

fortement liée à l'amplitude des mouvements dans la vidéo à l'instant correspondant. Le motif qui apparaît uniquement dans le cas de la vidéo 'news' est peut-être causé par les mouvements des danseurs en arrière-plan.

2.2. Code de Lempel-Ziv-Welch

Le code pour effectuer l'encodage et le décodage avec la méthode de Lempel-Ziv-Welch est dans le fichier `./LZW.py`. L'algorithme pour trouver le dictionnaire est le suivant :

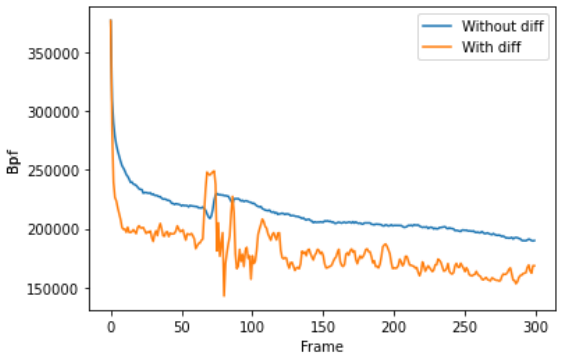
Algorithme de la méthode de Lempel-Ziv-Welch

1. Translater l'alphabet pour que les symboles soient dans $\mathbb{N}$.
2. Initialiser le dictionnaire avec les symboles de A (de 0 à max(A)).
3. Remplir le dictionnaire avec l'algorithme classique en utilisant les symboles translatés.
4. Connaissant la longueur maximum des mots dans le dictionnaire, compléter tous les mots avec des 0 pour qu'ils soient tous de la même longueur.

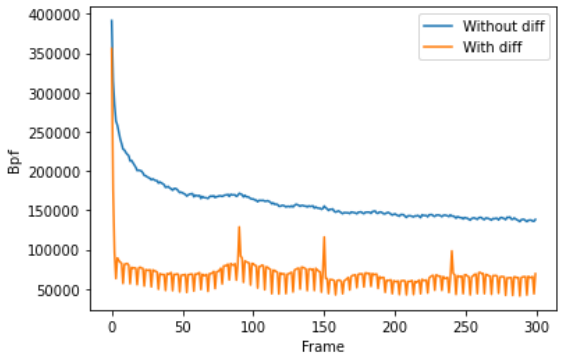
2.2.1. Format du train binaire

Le train binaire comprend le minimum des symboles encodés, la table des probabilités empiriques et les symboles encodés :

- sign+uint8 : minimum des symbols $min(A)$ encodé au format uint8 avec un bit pour le signe (0 positif, 1 négatif).
- uint9 : étalement des symboles ,compris entre 0 et 511 et égale à $\max(A) - \min(A) + 1$
- uint8 : longueur <n> des mots de code.
- k×uint<n> :série de mots de code de taille <n>.



Bits par frame pour la vidéo 'coastguard'.



Bits par frame pour la vidéo 'news'.

FIGURE 2 – Comparaison des bits par frame avec et sans les différences d'images pour la méthode de Lempel-Ziv-Welch

2.2.2. Résultats

Sur la figure 2 et dans le tableau 2, le codage entropique est comparé sur les données brutes des vidéos (sans différence d'image) et sur les images transformées avec $I'_0 = I_0$ et $I'_t = I_t - I_{t-1}, t > 0$ (voir 3.1).

Vidéo	coastguard	hall	news	akiyo	carphone
Taille compressée (%)	69.9	67.0	52.7	40.9	65.8
Bit par symbole	5.6	5.4	4.2	3.3	5.2
Taille compressée (diff)	59.6	42.8	22.1	14.4	49.3
Bit par symbole (diff)	4.8	3.4	1.7	1.1	3.9

TABLE 2 – Résultats pour la méthode de Lempel-Ziv-Welch

Concernant la compression et le nombre de bit par symbole, les remarques sont les mêmes que pour la méthode de Huffman mais les résultats sont globalement meilleurs. Le débit binaire est là aussi meilleur mais plus sensible aux mouvements lorsque les différences d'images sont utilisés.

3. Transformation et quantification

Plusieurs fichiers contiennent les fonctions présentées dans cette partie :

- **Difference_image.py :** fonctions permettant de calculer les differences d'image $I'_t = I_t - I_{t-1}$.
- **DCT.py :** fonctions pour calculer/inverser une série de transformations en une seule fois (dans l'ordre : DCT, quantification avec des matrices psychovisuelles, RLE).
- **Motion_estimation.py :** fonctions (pas complètement débuguées) utilisant la similitude par bloc et la DCT.

Le code pour calculer les différences d'images est dans le fichier Le code pour effectuer la série de transformation DCT/Quantification/RLE est dans le fichier `./LZW.py`.

3.1. Différence d'images

Les différences d'image sont calculées pour chaque image de la façon suivante :

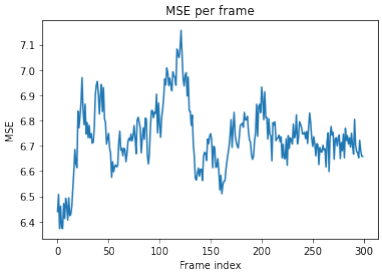
- $I'_0 = I_0$
- $I'_t = I_t - I_{t-1}$

Cette transformation permet de réduire la varriance des symboles dans l'alphabet en utilisant le fait que les valeurs successives des pixels sont corrélées. L'erreur de quantification peut se propager lors du décodage

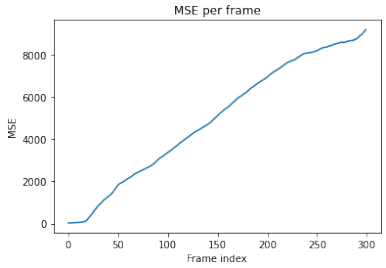
comme le montre l'image du milieu sur la figure 3. La solution implémentée consiste à garder une image non modifiée à interval régulié T :

- $I'_t = I_t$ si $t \bmod T = 0$
- $I'_t = I_t - I_{t-1}$ sinon.

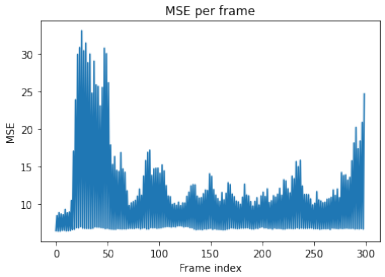
La qualité de la vidéo obtenue est assez mauvaise bien qu'elle soit suffisante pour comprendre la scène et les résultats obtenues (image de droite, Figure 3) montrent que même avec une valeur de T faible, la MSE reste élevée. L'origine de ce problème est peut-être la matrice psychovisuelle utilisée qui ne serait pas adaptée pour quantifier des différences d'images. En effet la quantification accorde d'avantage d'importance aux composantes de basse fréquence dans la DCT alors que les différences d'images sont constituées de détails (plutôt des hautes fréquences).



Quantification sur les images brutes



Quantification sur les différences d'images



Quantification sur les différences d'images, avec T=2

FIGURE 3 – Propagation de l'erreur de quantification lorsque les différences d'images sont utilisées (vidéo 'hall')

3.2. DCT, quantification et RLE

La DCT, la quantification et le RLE sont calculé à la suite dans une seule fonction définie dans `DCT.py`.

3.2.1. DCT

La DCT est calculée de manière classique, sur des blocs 8x8, en première partie de la fonction.

3.2.2. Quantification

La quantification est effectuée en deuxième partie de la fonction, après avoir déterminé la valeur optimale pour Q_{opti}. Chaque bloc B_{uv} produit par la DCT est quantifié de la façon suivante :

$$B_{quantif} = B_{uv}/(QM_{psycho})$$

La matrice de quantification M_{psycho} est une matrice psychovisuelle (différente selon que le bloc B_{uv} proviennent de Y,U ou V). Le coefficient Q peut-être automatiquement choisit égal à Q_{opti} ou passer en paramètre de la fonction, auquel cas il doit être supérieur ou égal à Q_opti. Cette valeur optimale de Q est déterminée pour que les symboles obtenus après quantification soit le plus étalé possible dans la limite de $[-255, +255]$, c'est à dire pour maximiser la qualité. Prendre une valeur plus grande pour Q réduit la qualité et l'étalement des valeurs quantifiées. La même valeur de Q est utisée pour quantifier tous les blocs obtenus avec les données passées en paramète de la fonction. Cette valeur est quantifiée sur $[0, 255]$ avec $\lceil Q \times 128 \rceil$ et mise en tête de la série de symboles obtenue après le RLE.

3.2.3. RLE

Les blocs quantifiés sont lus selon les digonales (Figure 4) puis encodés avec un codage sur longueur de séquence. Le codage est effectué de la façon suivante :

- Le premier symbole codé est le premier symbole lu sans modification.
- Tant qu'il reste des symboles non nul, ajouter (nombre de 0, coefficient non nul)
- Ajouter (0,0) à la fin de la séquence (balise de fin de séquence)

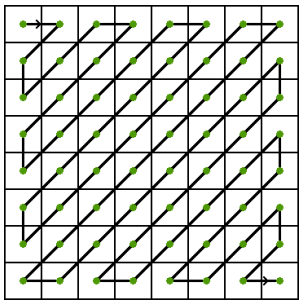


FIGURE 4 – Lecture diagonale

3.2.4. Résultats

Les différences d'images ont été calculées avec $T = 2$ mais comme mentionné précédemment les différences d'images ne permettent pas d'obtenir des vidéos de qualité très bonne.

Vidéo	coastguard	hall	news	akiyo	carphone
MSE (DCT seule)	7.98	6.74	18.0	13.8	10.5
MSE (DCT+Diff)	30.6	10.4	25.6	14.5	30.1
DCT seule	63.1 %	47.5 %	34.2 %	23.1 %	36.1 %
DCT + LZW	29.1 %	19.0 %	10.9 %	6.42 %	15.2 %
DCT + Huffman	30.6 %	23.3 %	15.9 %	10.2 %	17.2 %
Diff+ DCT	58.2 %	36.7 %	21.2 %	14.2 %	29.5 %
Diff+ DCT + LZW	25.5 %	15.1 %	7.05 %	3.78 %	12.3 %
Diff+ DCT + Huffman	26.0 %	16.6 %	9.10 %	5.59 %	12.9 %

TABLE 3 – Résultats pour la DCT

3.3. Estimation du mouvement

Les fonctions utilisées pour effectuer l'estimation de mouvement sont définies dans `Motion_estimation` mais ne sont pas complètement débuguées. Une image sur deux est codée en utilisant une similitude par bloc avec pour références l'image précédente et l'image suivante. Les autres images restent inchangées.

Annexes

Lien du dépot git contenant le code du projet : https://github.com/alexandre-lefebv/data_compression.git

Le dépot est organisé de la façon suivante :

- **video.py :** fonctions pour lire et sauvegarder des vidéos au format qcif.
- **Difference_image.py :** fonctions permettant de calculer les differences d'image $I'_t = I_t - I_{t-1}$.
- **Huffman.py :** fonctions pour coder/decoder une série de symboles en utilisant un codage de Huffman.
- **LZW.py :** fonctions pour coder/decoder une série de symboles en utilisant un codage de Lempel–Ziv–Welch.
- **DCT.py :** fonctions pour calculer/inverser une série de transformations en une seule fois (dans l'ordre : DCT, quantification avec des matrices psychovisuelles, RLE).
- **Motion_estimation.py :** fonctions (pas complètement débuguées) utilisant la similitude par bloc et la DCT.
- **tools.py :** fonctions pour mesurer la qualité et l'efficacité des commpressions testées.
- **test_res*.qcif :** vidéos qcif après compression, créées dans le main.ipynb.
- **main.ipynb :** Notebook permettant de tester différentes configuration de la chaîne de compression.
- **./tutos :** Notebooks permettant de tester individuellement certains blocs (essentiellement pour le débug).
- **./videos :** vidéos qcif pour effectuer les testes de compression.
- **./image rapport :** images sauvegardées pour le rapport.

OUR WORLDWIDE PARTNERS UNIVERSITIES - DOUBLE DEGREE AGREEMENTS

3 CAMPUS, 1 SITE

IMT Atlantique Bretagne–Pays de la Loire – **http://www.imt-atlantique.fr/**

Campus de Brest
Technopôle Brest-Iroise
CS 83818
29238 Brest Cedex 3
France
T +33 (0)2 29 00 11 11
F +33 (0)2 29 00 10 00

Campus de Nantes
4, rue Alfred Kastler
CS 20722
44307 Nantes Cedex 3
France
T +33 (0)2 51 85 81 00
F +33 (0)2 99 12 70 08

Campus de Rennes
2, rue de la Châtaigneraie
CS 17607
35576 Cesson Sévigné Cedex
France
T +33 (0)2 99 12 70 00
F +33 (0)2 51 85 81 99

Site de Toulouse
10, avenue Édouard Belin
BP 44004
31028 Toulouse Cedex 04
France
T +33 (0)5 61 33 83 65

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

Imprimé à IMT Atlantique
Dépôt légal : Avril 2021
ISSN : n° ISSN