Compression Vidéo

Frédéric BRESSO[1] et Charles CERISIER[2]

15 mars 2019

1 Introduction

L'objectif de ce TP est de comprendre et de réaliser la partie d'estimation de mouvement d'un encodeur vidéo. En effet, fichier vidéo possède une grande redondance statistique au niveau des données spatiales, mais aussi des données temporelles. Afin de faire une estimation de mouvement, nous avons réalisé, en partie, un algorithme de Block Matching. Ce programme a pour but d'encoder le mouvement dans une vidéo, en recherchant des blocs similaires entre différentes images. Cet algorithme de compensation de mouvement est utilisé dans certaines normes de compression vidéo comme le H.264 et MPEG-2.

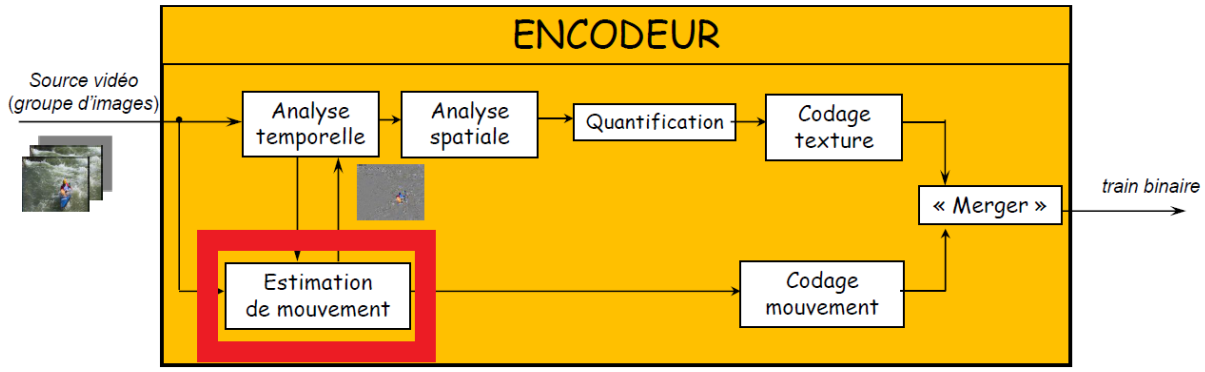

Figure 1: Schéma simplifié d'un encodeur vidéo

2 Fichier YUV

Les vidéos utilisées lors de ce TP étaient des vidéos YUV non compressées. Le premier objectif était donc de lire les images avec Python 3. Sachant que la résolution des images était de 288×352 et qu'elles étaient échantillonnées en 4:2:0, il était facile de déterminer la taille d'une image.

$$Taille_{image} = 352 \times 288 + 2 \times (176 * 144)$$
$$Taille_{image} = 152064 \text{ octets}$$

Pour obtenir la première image, il est donc nécessaire de récupérer les 152064 premiers octets du fichier yuv.

Dans notre algorithme de Block Matching, seule la composante Y de l'image récupérée est intéressante car c'est elle qui possède le plus d'information.

Afin de manipuler plus facilement et d'observer l'image Y que nous venons de récupérer, on a reconstruit une image en 2 dimensions (2D).

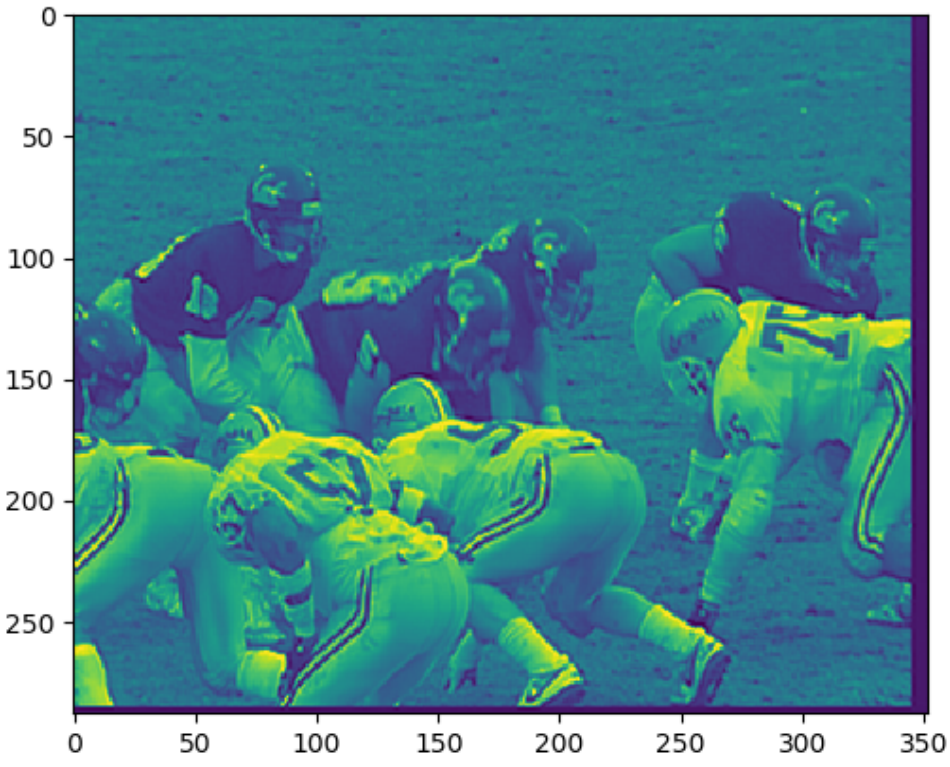

Figure 2: Composante Y de l'image 1 de la vidéo Football

3 Block Matching

Pour trouver le meilleur bloc correspondant au bloc d'une image courante dans une image de référence, l'algorithme de Block Matching parcourt l'image bloc par bloc dans un certain ordre. Nous avons réalisé 2 types d'algorithme de Block Matching avec le même critère de ressemblance des blocs : la SAD.

3.1 SAD

La Somme des différences absolues (SAD) est un algorithme simple utilisé afin de trouver une corrélation entre les blocs ou macro-blocs d'une image. Elle est déterminée en calculant la différence absolue entre chaque pixel dans un bloc de l'image référence et le pixel d'un bloc de l'image courante.

$$SAD = \sum | \, block1_{i,j} - block2_{i,j} \, |$$

3.2 Full Search Algorithm

Dans un premier temps, la recherche du meilleur bloc était réaliséé dans l'ensemble de l'image de référence. Cependant, cet algorithme était trop gourmand en calcul et nos ordinateurs prenaient en moyenne 25 secondes pour comparer le bloc courant à tous les blocs de l'image de référence. Sachant qu'il y avait 396 blocs (de 16×16 pixels) dans l'image courante, il fallait environ 3 heures de calcul. C'est pourquoi, nous avons réduit notre fenêtre de recherche.

Dans ce second algorithme, nous avons donc utilisé une fenêtre de recherche de 48×48 pixels autour du bloc courant en partant du principe que le mouvement n'a pas été trop important. Le temps de calcul pour tous les blocs est d'ici largement inférieur à l'algorithme précédent, environ 1 minute 30.

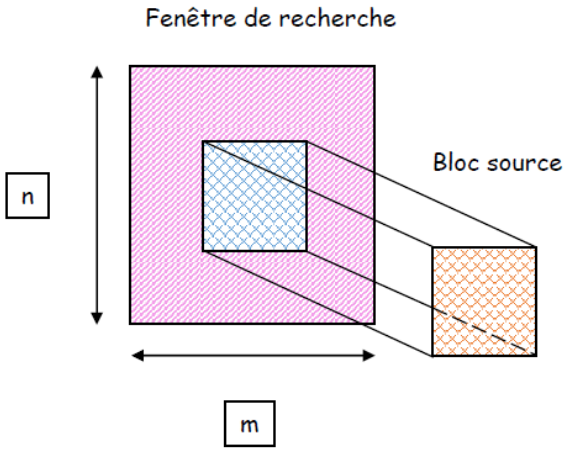

Figure 3: Block Matching avec fenêtre de recherche

Pour chaque meilleur bloc trouvé, on enregistre un vecteur mouvement qui correspond au déplacement du bloc de l'image de référence à l'image courante : c'est l'estimation de mouvement.

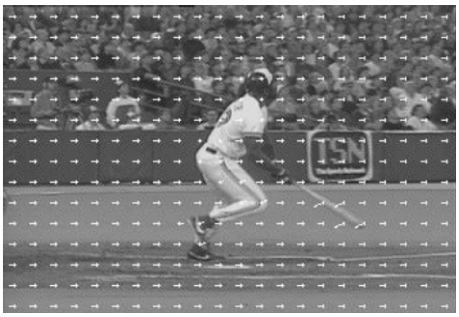

Figure 4: Image d'un champ de mouvement horizontal

3.3 Compensation de mouvement

La compensation du mouvement est un algorithme utilisé dans un décodeur. Le but de la compensation de mouvement est récréer l'image courante à partir de l'image de référence et du champ de vecteurs.

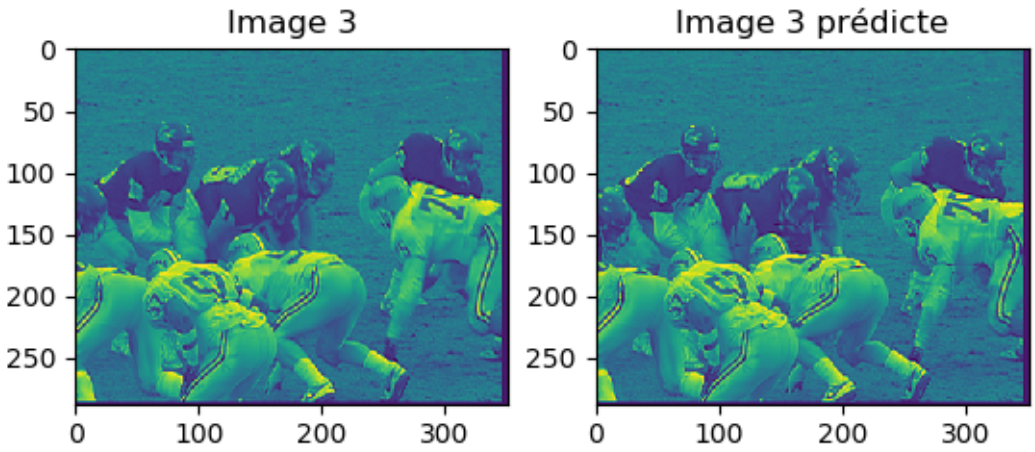

Figure 5: Comparaison de l'image 3 de la vidéo et celle compensée en mouvement

On observe quelques différences aux endroits où le déplacement a été le plus important.

3.4 Quart de Pixel

Le mouvement n'étant pas pixelique, il est intéressant d'essayer d'augmenter la résolution de l'image de façon artificielle. Afin d'obtenir une précision au quart de pixel, il faut dans un premier temps créer une image vide 4 fois plus grande que celle de base. Dans un second temps, on remplace 1 pixel sur 4 de cette nouvelle image par les pixels de l'image de référence. Finalement, on réalise une moyenne pondérée par leur distance entre les 4 pixels existants.

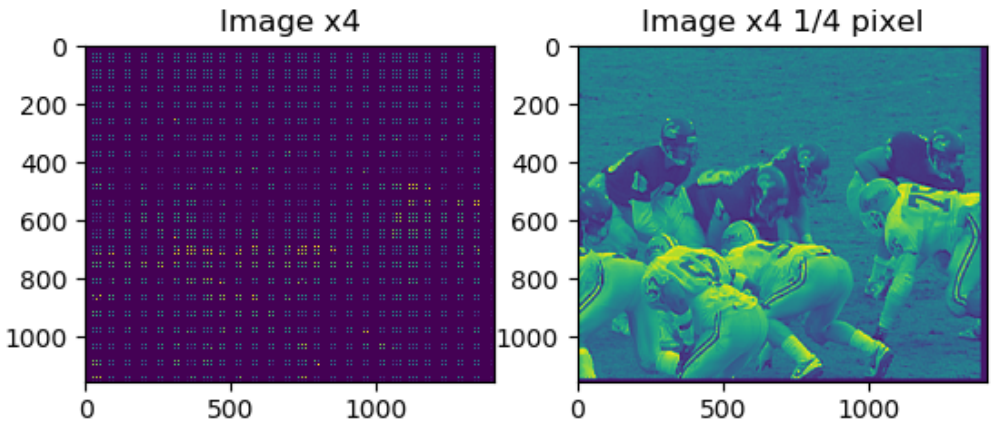

Figure 6: Comparaison de l'image x4 et l'image x4 reconstruite au quart de pixel

Cette nouvelle image, précise au quart de pixel peut être réinjectée dans un algorithme de block matching pour augmenter la précision.

4 Conclusion

Ce TP a été très intéressant car il nous a permis de comprendre en partie un encodeur vidéo et l'algorithme de Block Matching utilisé dans la norme H264. Ce projet a très bien complété notre premier TP de compression d'image où nous nous étions intéressé à la redondance des données spatiales, ici on s'intéresse en plus à la redondance temporelle.

Afin d'améliorer notre programme, nous aurions pu implémenter l'estimation hiérarchique.

References

[1] J.Viéron, ESEO 2019, Compression Vidéo - Rappels

[2] J.Viéron, ESEO 2019, H264