Université de la Rochelle
Master 1 ICONE - Codage de l'information

TD/TeA - Algorithme MPEG

© Renaud Péteri

Année universitaire 2019-2020

Dans ce TeA, on se propose de simuler le principe du codage/décodage MPEG de manière simplifiée. Comme vous le savez (cf. cours), ce type de codage tire partie de la redondance temporelle des séquences d'images par prédiction/compensation de mouvement et est donc l'occasion de mettre en pratique ce que vous avez vu en TP.
Vous remettrez (évidement) un petit rapport sur vos manips commentées pour vendredi soir...

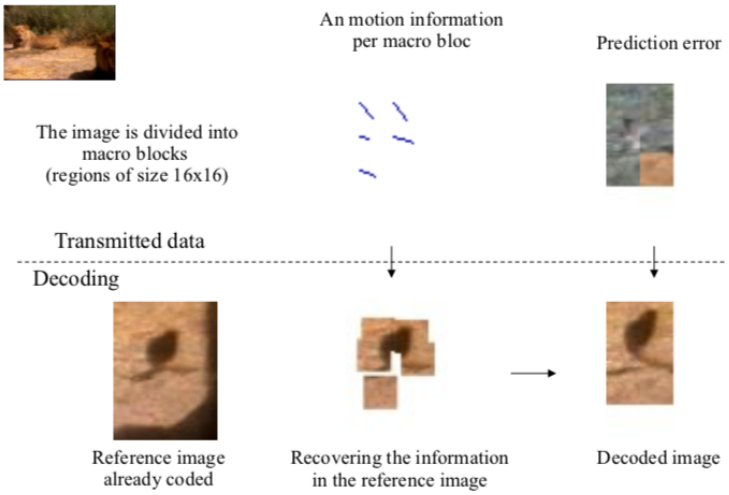

FIGURE 1 – Principe du codage MPEG

Préliminaires

Ouverture et traitement en série d'images

La fonction suivante permet l'ouverture d'une séquence d'images en Python :

```
1 import matplotlib.pyplot as plt
2 import os
3 import numpy as np
4 import glob
5 import time
6
7 def batch_processing(prefix,ext,indstart, indend):
8  """
```

```
prefixe:       image prefix of the sequence (i.e. "im_")
ext:           extention of the image (i.e. "png", "jpg")
indstart:     index of the first image of the sequence
indend:       indice of the last image of the sequence
function returns imList, imProcess: lists of images and processed images
"""

pathDB = "path_to_your_directory"
imList=[]
imProcessed=[]
Abs_path=os.path.abspath(pathDB)

if (os.path.exists(Abs_path)):
imfile_start= prefix +"%02d" %indstart + '.'+ ext
pathStart=os.path.join(Abs_path,imfile_start)
print("Starting image file:", pathStart)

imfile_end= prefix +"%02d" %indend + '.'+ ext
pathEnd=os.path.join(Abs_path,imfile_end)
print("Ending image file:", pathEnd)

#% Loading
tic = time.time()
print("Loading sequence...")
# PUT YOUR CODE HERE!
toc = time.time()
print(toc-tic, "sec elapsed for loading!")

#% Processing
tic = time.time()
print("Processing sequence...")
# PUT YOUR CODE HERE!
toc = time.time()
print(toc-tic, "sec elapsed for processing!")

else :
print("This path does not exist !")

return imList, imProcessed
```

1. Analysez les différentes étapes de cette fonction.
2. Ecrire le code permettant le chargement de la séquence d'images `cars.zip` dans une variable de type liste `imList`.
3. Donner un exemple d'appel de cette fonction.

Estimation du mouvement d'un macrobloc

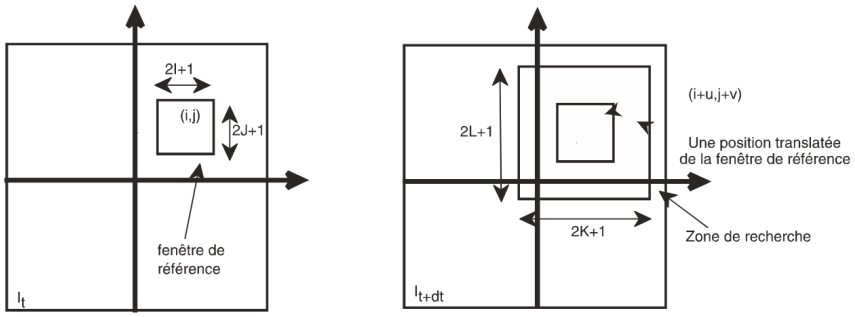

FIGURE 2 – Principe de recherche du déplacement d'un macro-bloc

Pour estimer le déplacement d'un macro-bloc (taille 16×16 pixels) entre 2 frames consécutives, on effectuer les étapes suivantes :

- On définit une zone de recherche autour du macro-bloc (généralement taille double du macrobloc, - voir Figure 2)
- On calcule le critère SSD (Somme des différences au carré - voir figure 3)) entre le macro-bloc et une fenêtre de même taille mais translatée du vecteur (u, v).
- La position $V_{min} = (u_{min}, v_{min})$ qui minimise cette différence est retenue et correspondra au déplacement du macro-bloc (utilser `np.argmin(...)`).

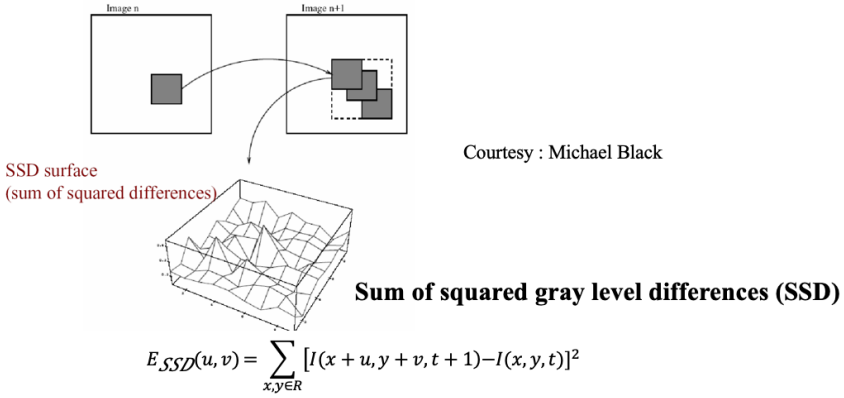

FIGURE 3 – Coût d'un déplacement par critère SSD

Cahier des charges

On souhaite simuler le codage d'une `P-frame` à parti d'une `I-frame`. L'idée est de pouvoir reproduire le schéma de la figure 1, **à vous de chercher aussi des informations utiles au besoin**.
Sur les 2 premières images de la séquence `cars`, vous devrez :

1. diviser votre `I-frame` en macro-blocks de taille 16 × 16.
2. estimer et afficher sur la première frame le mouvement des macro-block. Pour cela, on estimera dans la `P-frame` une fenêtre de recherche autour du macro-bloc via la fonction de coût SSD (*sum of square differences*)
3. Estimer l'image erreur de prédiction.

Enfin, pour toute la séquence, répéter les étapes précédentes pour toute la séquence (la 1ère frame sera la I-frame référence des 5 autres).
A votre avis, à quelle étape et comment la compression d'information aura t'elle lieu ?

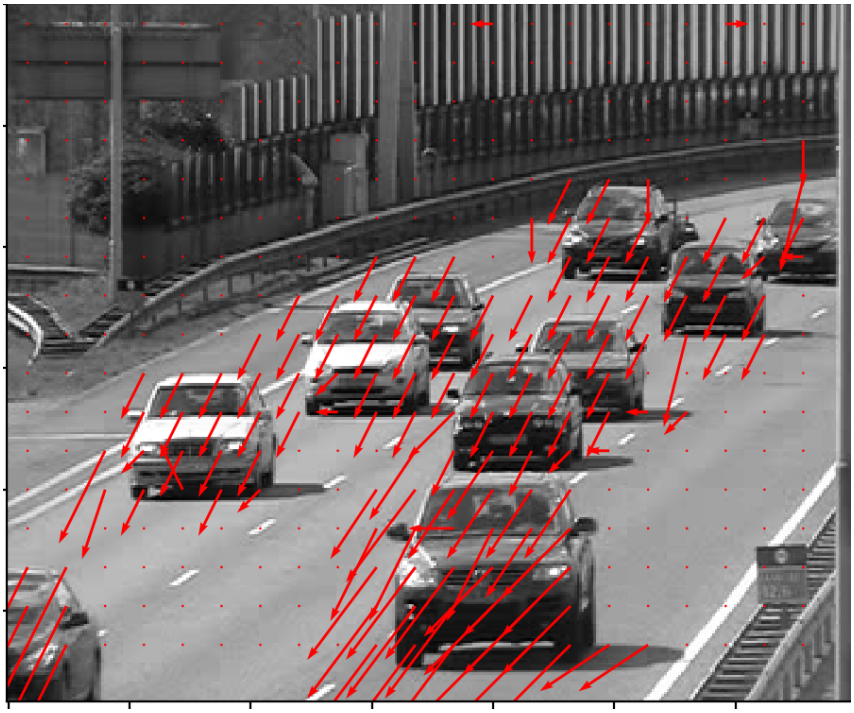

FIGURE 4 – Exemple de champs de vecteurs mouvements estimé pour chaque macro-bloc