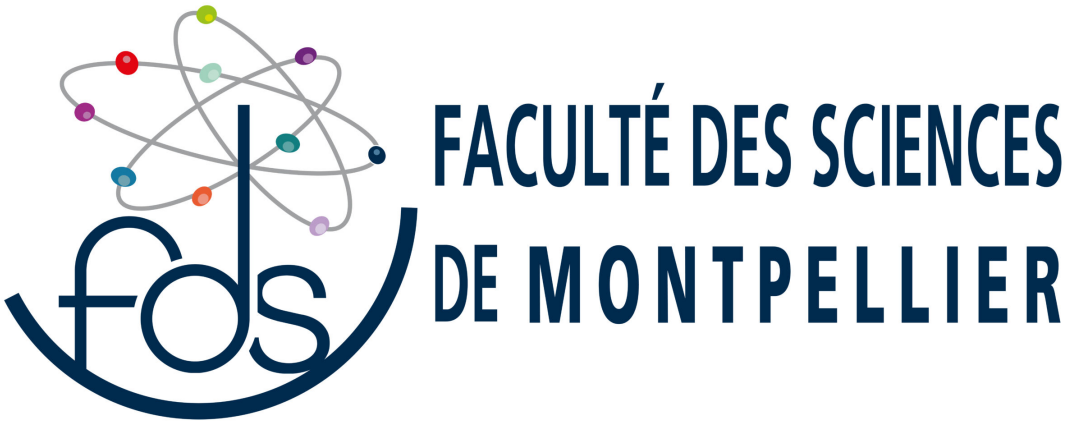
Université de Montpellier

M1 IMAGINE - Codage et Compression
Compte Rendu TP2 - Changement d'espace couleur

Etudiant :
Guillaume Bataille

Encadrant :
William Puech

Année 2022-2023



0.1 Contexte et objectif

Dans ce cours, nous avons vu différents espaces couleur servant à représenter une image

Objectif :

-Observer l'impact sur la qualité d'une image couleur après réduction spatiale.
-Comparer les performances en fonction de l'espace couleur utilisé.
-Et enfin programmer le tout pour avoir en entrée une image couleur au format ppm et en sortie une image couleur au format ppm ainsi que sa qualité en termes de PSNR.

Voici l'image qu'on a choisi pour les traitement de ce tp :

FIGURE 1 – cameleon.ppm (512x322)

Sommaire

1. Compression par 2 dans l'espace RGB

Afin de compresser notre image RGB (.ppm) par 2 on procède comme cela :

1.1 Découpe en composante RGB

Découper l'image R G B en 3 composantes.

FIGURE 1.1 – Les composantes R G et B de l'image cameleon.ppm

Comme convenu dans le cours, l'image G (centrale) semble contenir la majorité des informations.

1.2 Compression de 2 composante

-Choisir deux de ces composantes et les reduire par 4 (par 2 en longueur et en largeur)

FIGURE 1.2 – Les composantes R G et B reduite de l'image cameleon.ppm (même scale qu'au dessus)

La compression est faire en écrasant des blocs de 4 pixels en un seul pixel représentant la moyenne du bloc.

1.3 Rééchantillonage

-On reéchantillonne ces deux composantes pour revenir au format initial

FIGURE 1.3 – Les composantes R G et B reéchantillonnées de l'image cameleon.ppm

On constate qu'il y a une perte de qualité et donc d'information, ce qui est normal car l'image initiale est en perte d'information. On reconstruit celle ci en remplissant nos blocs de 4 pixels par pixel (moyenne) de l'image reduite précédente.

1.4 Assemblement final RGB

-On assemble les trois composantes et on repasse dans l'espace couleur RGB pour avoir une image compressée par 2 (sans prise en compte de l'entête).

FIGURE 1.4 – image cameleon.ppm avec R conservée - G conservée - B conservée

Le PSNR est de :
-**25,7** dB pour l'image R sauvegardé
-**26,6** dB pour l'image G sauvegardé
-**25,7** dB pour l'image B sauvegardé

2. Compression par 2 dans l'espace YCrCb

2.1 Conversion de l'espace couleur RGB -> YCrCb

Pour faire la conversion, on utilise une fonction pour faire la conversion RGB -> YCrCB

```
1  //Fonction qui convertit une image RGB en une image YCrCb
2  void RGBtoYCbCr(OCTET *ImgIn, OCTET *ImgOut, int taille)
3  {
4      for (int i = 0; i < taille; i++)
5      {
6
7          ImgOut[3 * i] =
8  std::min(255., std::max(0., 0.299 * ImgIn[3 * i] + 0.587 * ImgIn[3 * i + 1] + 0.114 * ImgIn[3 * i + 2]));
9          ImgOut[3 * i + 1] =
10 std::min(255., std::max(0., -0.1687 * ImgIn[3 * i] - 0.3313 * ImgIn[3 * i + 1] + 0.5 *ImgIn[3 * i + 2]+128));
11         ImgOut[3 * i + 2] =
12 std::min(255., std::max(0., 0.5 * ImgIn[3 * i] - 0.4187 * ImgIn[3 * i + 1] - 0.0813 *ImgIn[3 * i + 2]+128));
13     }
14 }
```

Et on anticipe aussi la transformation inverse YCrCb -> RGB

```
1  //Fonction qui convertit une image YCrCb en RGB
2  void YCbCrtoRGB(OCTET *ImgIn, OCTET *ImgOut, int taille)
3  {
4      for (int i = 0; i < taille; i++)
5      {
6          ImgOut[3 * i] =
7  std::min(255., std::max(0., ImgIn[3 * i] + 1.402 * (ImgIn[3 * i + 2] - 128)));
8          ImgOut[3 * i + 1] =
9  std::min(255., std::max(0., ImgIn[3 * i] - 0.34414*(ImgIn[3 * i + 1] - 128) - 0.71414*(ImgIn[3 * i + 2]-128)));
10         ImgOut[3 * i + 2] =
11 std::min(255., std::max(0., ImgIn[3 * i] + 1.772 * (ImgIn[3 * i + 1] - 128)));
12     }
13 }
```

Attention, on verifie que les valeurs soient bien entre 0 et 255 après conversions

Voici ce qu'on obtient et l'image RGB en comparaison :

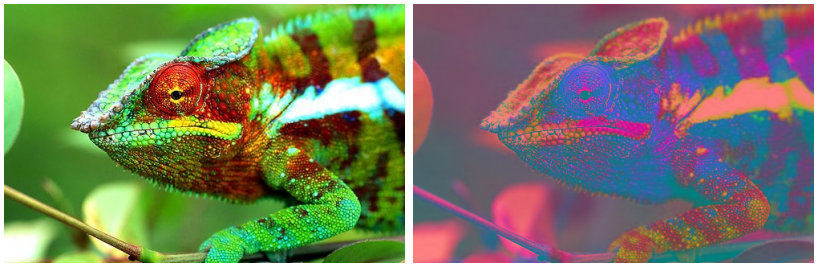


FIGURE 2.1 – Image RGB et Image après conversion YCrCb du cameleon

2.2 Découpe en composante YCrCb

Découper l'image Y Cr Cb en 3 composantes.

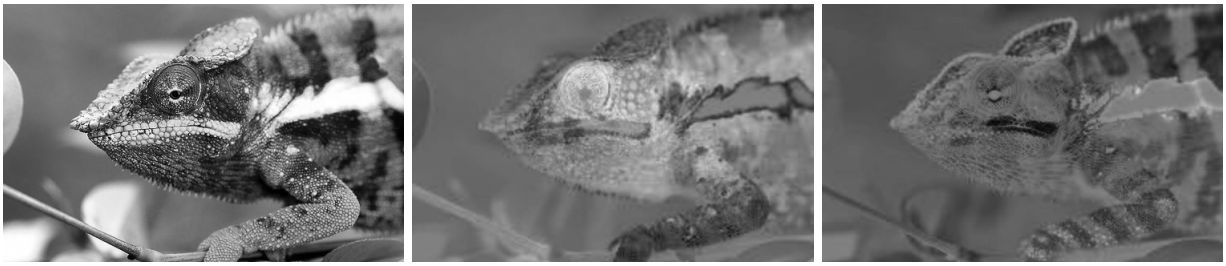


FIGURE 2.2 – Les composantes Y Cr Cb de l'image cameleon.ppm

On voit que la composante Y, la luminance, porte la plupart des détails de l'image.

2.3 Compression des 2 composante Cr et Cb

-Prendre les composantes Cr et Cb et les reduire par 4 (par 2 en longueur et en largeur)

FIGURE 2.3 – Les composantes Cr Cb reduite de l'image cameleon.ppm

La compression est faire en écrasant des blocs de 4 pixels en un seul pixel représentant la moyenne du bloc.

2.4 Rééchantillonage

-On reéchantillonne ces deux composantes pour revenir au format initial

FIGURE 2.4 – Les composantes Cr et Cb reéchantillonnées de l'image cameleon.ppm

On retrouve encore une fois le flou dû a la perte d'information lié a la compression du dessus.

2.5 Assemblement final YCrCb

-On assemble les trois composantes pour avoir une image compressée par 2 (sans prise en compte de l'entête).
Le PSNR de cette image est de **36** dB.

FIGURE 2.5 – image cameleon.ppm et l'image reconstruite après fusion de Y, Cr flou et Cb flou