

Projet Vision

Partie 1 : Texte caché dans une image

Aoucher Kenza

`anoukenza@gmail.com`

Benmessaoud Ahmed Sif

`ahmed.sif.benmessaoud.13@gmail.com`

Table des matières

1	Encodage	4
2	Première méthode d'encodage : RBGA	4
3	Seconde méthode d'encodage : YCbCr	5
4	Application	6

1 Encodage

Un message est une suite de caractères, un *char generator* génère alors la représentation binaire de chaque caractère, et les renvoie un à un à l'encodeur. Ce dernier parcourt les $(n + 1)$ premiers pixels qui composent l'image, avec n étant la taille du message à encoder. Le dernier pixel est mis à 0 pour signaler au décodeur que le message est terminé. D'autres méthodes existent dans la littérature afin de communiquer la taille du message au récepteur, comme encoder celle-ci dans le premier pixel ou dans les pixels d'un canal réservé à cet effet (la luminance Y dans YCbCr par exemple). Enfin l'image contenant le message est envoyée en RGB au programme principal. La figure 1 représente ce processus d'encodage du message secret dans l'image avec les deux techniques implémentées.

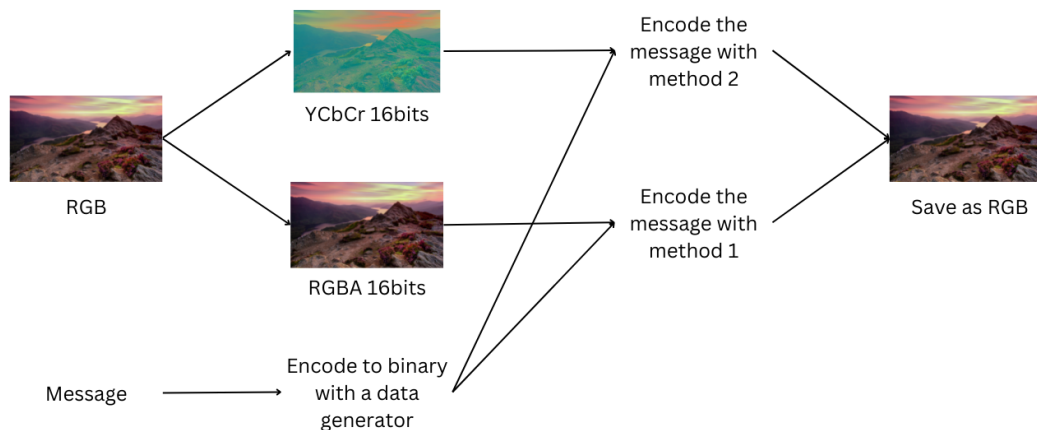


FIGURE 1 – Processus d'encodage

2 Première méthode d'encodage : RBGA

Cette méthode exploite un quatrième canal *Alpha* qui représente la luminosité de l'image, ainsi un pixel est décrit par 4 valeurs (R, G, B et Alpha). Cela nous permet de réserver les deux bits de poids faibles de chaque valeur afin d'encoder le caractère qui nécessite 8

bits ($4 \times 2 = 8$), comme le montre la figure 2. La taille maximale du message est alors :

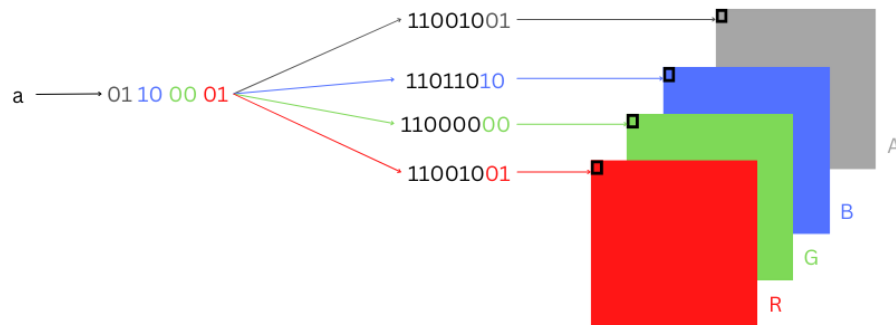
$$\text{max} = \text{width} \times \text{height}$$


FIGURE 2 – Méthode RGBA

3 Seconde méthode d'encodage : YCbCr

Dans le *color space* YCbCr l'image est représentée sur 3 canaux (luminance Y, chroma blue Cb et chroma red Cr) (Figures 3 et 4).



FIGURE 3 – Image en YCbCr



FIGURE 4 – Canaux YCbCr

Les canaux Cb et Cr contiennent le message secret, tel que les 4 bits de poids faibles sont mis à 0111, pour éviter les erreurs, et les bits 5 et 6 renferment 2 bits d'un caractère. Avec cette technique un caractère est encodé sur 4 pixels, comme le montre la figure 2.

La taille max du message est alors :

$$\max = \frac{2 \times \text{width} \times \text{height}}{4} = \frac{1}{2} \times \text{width} \times \text{height}$$

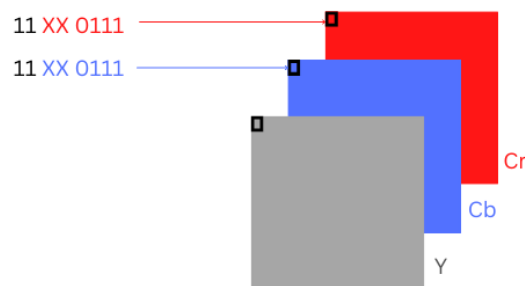


FIGURE 5 – Méthode YCbCr

4 Application

Dépendances

1. numpy : manipulation des matrices
2. cv2 : ouverture des images, conversions, traitements...
3. streamlit : l'interface graphique (*web app*)

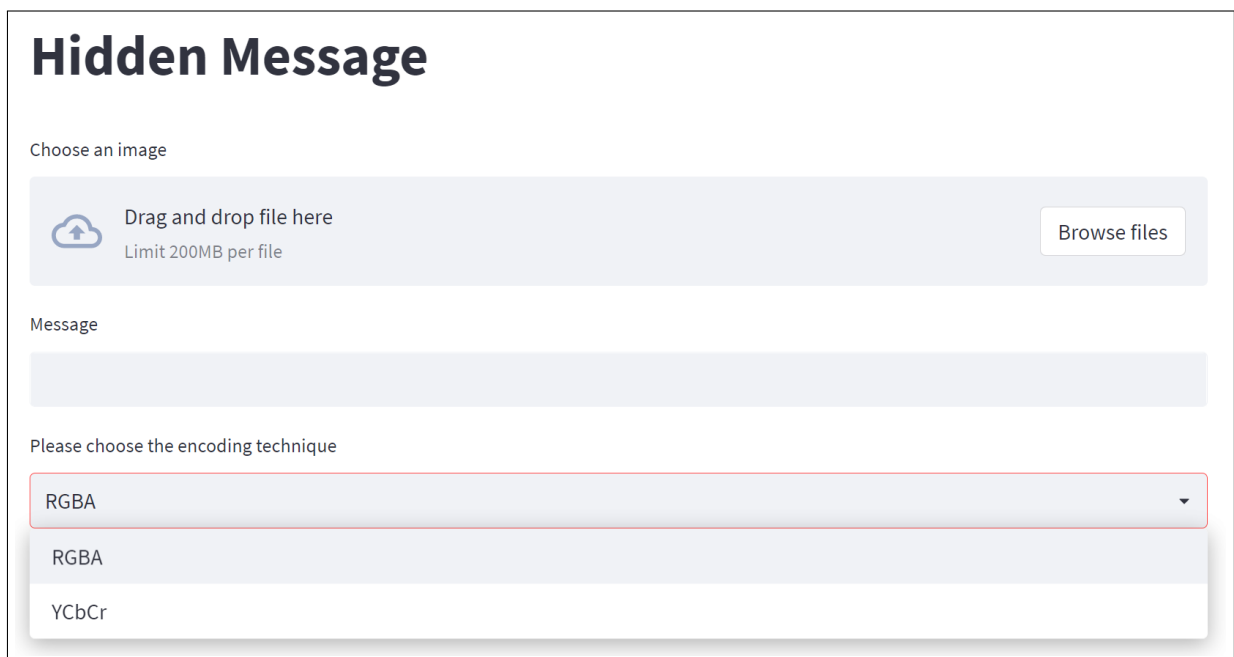
Exécution

L'exécution se fait à l'aide des lignes suivantes (la première sert à installer les dépendances du programme et la seconde à lancer l'exécution) :

```
pip install -r requirements.txt  
streamlit run main.py
```

Processus

L'interface permet d'importer l'image souhaitée, d'ajouter le message à encoder ainsi que la méthode désirée comme le montre la figure 6. En utilisant le *slider* il suffit de faire glisser le curseur afin d'envoyer l'image contenant le message encodé au récepteur (Figure 7).



Hidden Message

Choose an image

Drag and drop file here
Limit 200MB per file

Browse files

Message

Please choose the encoding technique

RGBA

RGBA

YCbCr

FIGURE 6 – Choix de l'image, du message et de la méthode

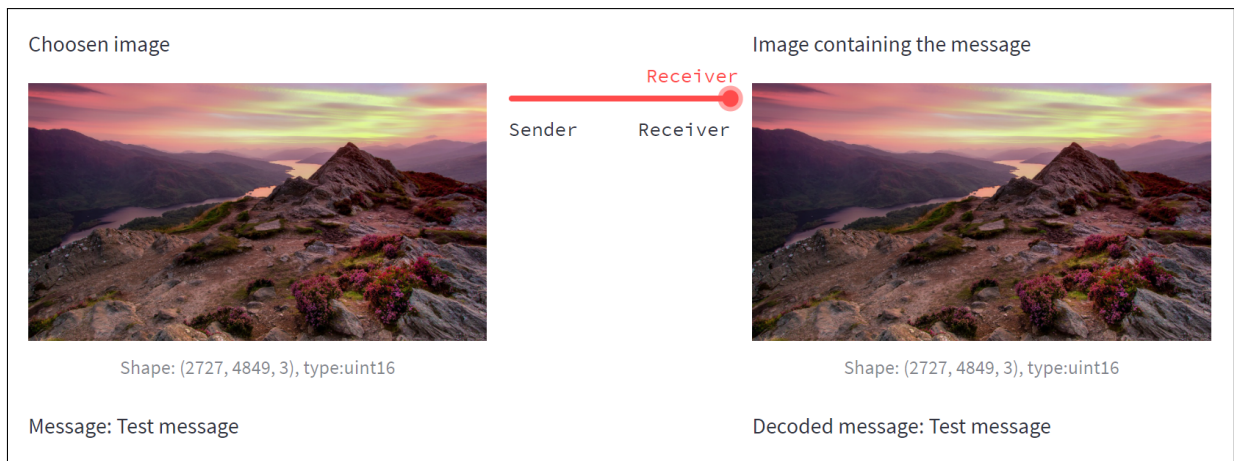


FIGURE 7 – Envoie au récepteur et décodage