Mini Projet : L'image mystère

1. Comment cacher une image dans une autre ?

Pour cacher une image dans une autre, on utilise le fait que l'œil ne distingue pas les faibles différences de couleur. Par exemple, le mot a b r a c a d a b r a tel qu'il est écrit ne contient pas deux lettres de même couleur, ce qui est difficile à voir. Les couleurs employées sont trop proches les unes des autres pour pouvoir être distinguées.

On utilise le fait que modifier légèrement la couleur des pixels d'une image est difficile à discerner pour l'œil humain. Prenons 2 images de même dimension, il suffit de mélanger les pixels des deux images originales, en privilégiant l'image 1 pour ne pas remarquer que l'image 2 est cachée dedans.

Pour chacune des coordonnées (x,y), on va mélanger une partie du pixel (x,y) de l'image 1 avec le pixel (x,y) de l'image 2. L'image obtenue sera très proche de l'image 1, mais l'image 2 sera « cachée » à l'intérieur.

Pour simplifier, voyons ce qui se passe sur une seule composante de couleur, disons le rouge (on appliquera en fait le même traitement aux trois couleurs). Imaginons que l'on souhaite mélanger un pixel dans l'image 1 dont l'octet rouge vaut $\mathbf{190}_{10}$ c'est-à-dire 10111110_2 avec l'octet de l'image 2 qui vaut $\mathbf{121}_{10}$, c'est-à-dire 01111001_2

L'idée est de recombiner les 5 bits les plus significatifs du premier pixel avec les 3 bits les plus significatifs de l'image 2.

De nos 2 octets d'origine **10111** 110 et **011** 11001 on ne garde que les bits les plus significatifs, en mettant ceux de la première image en tête. Ce mélange des deux octets d'origine produit le nouvel octet suivant: **10111 011** $_2$ soit $\mathbf{187}_{10}$

On remarque que l'on a peu modifié la valeur par rapport à celle de l'image 1, parce qu'on a gardé les 5 bits les plus significatifs. Sur notre exemple, la différence est seulement de 3, la nouvelle valeur est 187, ce qui donne un rouge **comme ce texte** au lieu de 190 dans l'image originale, ce qui donne un rouge **comme celui-ci** qui est très proche à l'œil nu. Au pire, on transforme les 3 derniers bits de **000** à **111**, ou vice-versa créant au plus une différence de 7. En résumé, la fusion des 2 images ressemble beaucoup à la première image.

2.Extraction d'une image cachée

But : Extraire d'une image, une image cachée par la méthode de la stéganographie décrite au dessus.

Prenons pour exemple un pixel couleur codé RVB de valeur [192,58,23] contenant un pixel d'une image cachée.

Q1. Justifier en appliquant la démarche ci-dessus que le pixel caché est de valeur [0,64,224]

Opération de décodage d'une composante de couleur

L'extraction de la composante de couleur cachée peut être obtenue en conservant par masquage les 3 bits de droite en partant du LSB de l'octet de la composante de couleur puis en les décalant sur la gauche (voir figure 1).

L'opération de masquage peut être réalisée à l'aide d'un opérateur ET logique effectué bit à bit. Sous python cet opérateur logique s'écrit `&`

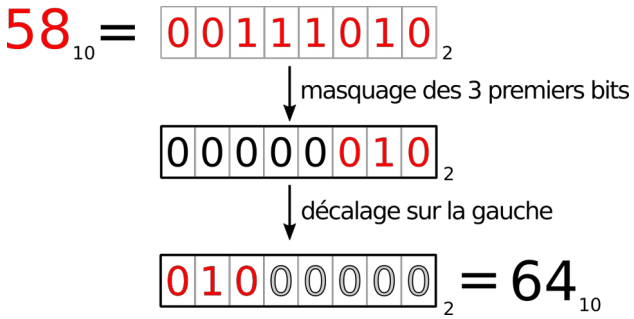

Figure 1: Méthode d'extraction de la composante de couleur cachée

Q2. Justifier que l'expression valeur & 7 renvoie le nombre égale aux 3 bits de droite de valeur

Le décalage de bits sur la gauche peut être réalisé sous Python par l'opérateur `<<`.

Q3. Tester sur la console python les expressions `1<<1`, `1<<4`, `3<<4` puis **justifier** sur votre feuille les résultats obtenus.

Q4. Déterminer le nombre de décalage à effectuer sur chaque composante de couleur d'un pixel.

Q5. En déduire l'expression en langage Python permettant d'extraire de la composante de couleur de l'image initiale, la composante de couleur de l'image cachée. **Tester** cette expression sur les trois composantes de couleur définies au dessus (question Q1)

Programme d'extraction de l'image cachée

L'algorithme d'extraction de l'image cachée est le suivant :

```
img ← image initiale # Charger en mémoire dans img l'image initiale
imgCachee ← copie de img # Créer imgCachee de même taille que img
pour chaque pixel de img :
    pour chaque composante de pixel :
        extraire composanteCachee de composante
        stocker composanteCachee dans imgCachee
```

Q6. Traduire cet algorithme en programme Python. La bibliothèque manipulationImage utilisée lors de l'activité 4 peut être réutilisée.

Q7. Tester ce programme avec l'image de test image\imageTest.png. Cette image contient quatre pixels de couleur [192,58,23] comme défini en question Q1

Q8. Extraire de l'image image_cachee.pgm l'image cachée.

Fusion de deux images

But : Fusionner une image dans une autre afin de la cacher en suivant la méthode de stéganographie.

Q9. En s'inspirant de la méthode d'extraction présentée précédemment, **décrire** sur feuille une démarche permettant de cacher une composante de couleur d'une image à cacher dans une composante de couleur d'une image support.

Q10. Etablir l'expression Python réalisant cette méthode de fusion de composantes de couleur.

Q11. Proposer un jeu de valeurs test permettant tester cette expression puis **valider** sur la console Python cette expression.

Q12. Etablir un programme Python permettant de cacher une image dans une autre de même format. **Valider** ce programme.