Traitement d'image

TP 30

Structure de données - v1.0

Lycée polyvalent Franklin Roosevelt, 10 Rue du Président Franklin Roosevelt, 51100 Reims

Nous allons utiliser trois modules pour travailler avec les images :

- `imageio` pour lire les fichiers images ;
- `matplotlib.pyplot` pour afficher les images ;
- `copy` pour faire des vrais copies de structures.

Les trois lignes suivantes permettent de lire une image en couleur appelée `Lenna24.png`, de la transformée en une structure en liste de listes de listes et de l'afficher.

```
image = imageio.imread("Lenna24.png").tolist()
plt.imshow(image)
plt.show()
```

Pour afficher une image en noir et blanc, on pourra utiliser les lignes suivantes :

```
image = imageio.imread("Lenna8.png").tolist()
plt.imshow(image, cmap='gray', vmin=0, vmax=255)
plt.show()
```

Là, on crée alors une liste de listes.

Pour copier une structure imbriquée (liste de listes ou liste de listes de listes), il ne suffit pas de faire une copie superficielle du type : `M2 = M1[:]` ou, de façon équivalente, `M2 = M1.copy()`. En effet, la liste est bien une copie, mais les sous-listes référencent encore les mêmes objets. Il faut donc utiliser la fonction `deepcopy` du module `copy` :

```
from copy import deepcopy
M2 = deepcopy(M1)
```

1 Profondeur des couleurs

Pour vérifier le type d'une donnée, on peut utiliser la fonction `type`. Par exemple, `type([])==list` renvoie `True`. On rappelle que pour

Question 1 Écrire une fonction `taille(image)` qui renvoie une liste correspondant à la taille de l'image : nombre de lignes, puis nombre de colonnes et enfin nombre de composantes du pixel si l'image est en couleur. Vous pourrez vous aider d'une boucle `while`.

Question 2 Vérifiez que la taille des images `Lenna24.png`, `Lenna32.png`, `Lenna8.png` et `chausseeDes Geants.png` sont respectivement `[512, 512, 3]`, `[512, 512, 4]`, `[512, 512]` et `[519, 1366]`.

Question 3 Que permet le format `png` 32 bits par rapport au format `png` 24 bits ? Quelle est la taille d'un pixel (en nombre de bits) pour une image en niveau de gris ?

2 Incrustation par fond bleu

Question 1 Écrire une fonction qui prend en entrée les valeurs de 2 pixels et retourne une distance des 2 couleurs avec la formule suivante :

$$dist(p_1, p_2) = |p_{1R} - p_{2R}| + |p_{1G} - p_{2G}| + |p_{1B} - p_{2B}|$$

Question 2 En passant votre curseur sur l'affichage de l'image `personnage.png`, relever une valeur typique du fond bleu.

Question 3 Remplacer le fond bleu par la carte `carte.png` dans le fichier `personnage.png`.

3 Détection de contours

Afin de détecter des contours, on peut essayer d'approximer une dérivée spatiale par une approximation des différences finies. Cela correspond à des convolutions par les matrices suivantes :

$$\begin{pmatrix} -\frac{1}{4} & 0 & \frac{1}{4} \\ -\frac{1}{2} & 0 & \frac{1}{2} \\ -\frac{1}{4} & 0 & \frac{1}{4} \end{pmatrix} \begin{pmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ 0 & 0 & 0 \\ -\frac{1}{4} & -\frac{1}{2} & -\frac{1}{4} \end{pmatrix}$$

La première permet de détecter les variations horizontales, la deuxième, les variations verticales.

L'opération de convolution consiste à la somme des multiplications terme à terme entre les 9 pixels autour du pixels cibles. Ainsi, par exemple, la convolution avec la première matrice nous donne la formule :

$$Conv[i,j] = -\frac{1}{4}im[i-1,j-1] - \frac{1}{2}im[i,j-1] - \frac{1}{4}im[i+1,j-1] + \frac{1}{4}im[i-1,j+1] + \frac{1}{2}im[i,j+1] + \frac{1}{4}im[i+1,j+1]$$

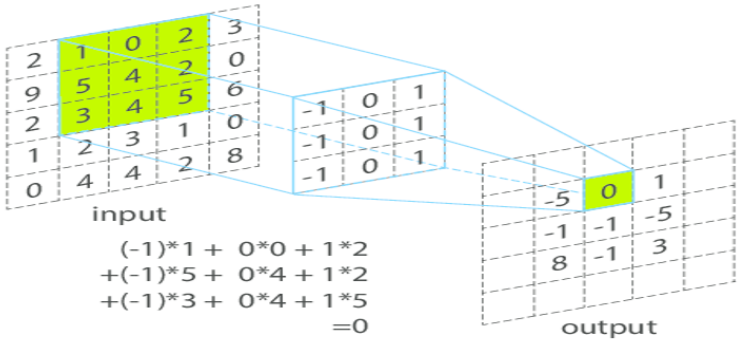

FIGURE 1 – Illustration de l'opération de convolution

Pour plus de simplicité, on ne traitera que les pixels hors des bords et des coins (ceux qui n'ont pas 9 voisins).

Question 1 Écrire une fonction `convolution(img, M)` qui renvoie le résultat de l'opération de convolution en utilisant l'image `img` et la matrice de convolution `M`. Pour faciliter la visualisation, ajoutera la valeur 128 à la somme pour recentrer la dynamique de l'image résultat.

Question 2 Appliquer les filtres présentés précédemment pour détecter les contours de l'image `Lenna8.png`. A quoi correspondent les parties claires et sombres de l'image résultat ?

4 Stéganographie

La stéganographie par réécriture du LSB (Least Significant Bit) consiste à cacher un texte dans une image (ici en niveau du gris). Chaque niveau de pixel étant codé sur 8 bits, la valeur de chaque pixel est comprise entre 0 et 255. Le codage se fait de la façon suivante :

- Pour chaque pixel de l'image, le bit le moins significatif (correspondant donc à la parité du nombre) correspondra à un bit de l'encodage du message caché ;
- Les bits extraits de l'image (les moins significatifs) seront regroupés 8 par 8 ;
- Chaque groupement de bits sera converti en un entier non signé (compris entre 0 et 255) ;
- Cet entier sera converti en un caractère Unicode à l'aide de la fonction `chr` de Python (par exemple chr(97) est évalué à `a`) ;
- On admettra par convention que le texte s'arrête lorsque l'entier lu est 128 celui-ci correspond à un caractère de contrôle Unicode.

Par exemple, supposons que la première ligne de l'image commence par 24, 13, 8, 24, 24, 24, 255, 2 : les bits correspondants sont 0100 0010 qui correspond à la valeur 66 qui code la caractère B. Avant l'opération de stéganographie, les valeurs des pixels pouvaient être par exemple dans l'état : 23, 12, 8, 23, 24, 23, 254, 1 ou 24, 13, 7, 24, 23, 23, 255, 2. On voit que la modification des valeurs est très faible, ce qui permet pour un observateur extérieur de ne pas se douter qu'un message est codé dans l'image.

Question 1 Décoder le texte caché dans la photo `chausseeDesGeants.png`.

Rappel : on pourra s'aider d'une fonction de conversion donnant un entier en fonction de sa représentation binaire sous forme de chaine de caractères : `int('01000010', 2) -> 66`.