

Filtres d'image

Le principe du filtrage est de modifier la valeur des pixels d'une image, généralement dans le but d'améliorer son aspect. Chaque pixel transformé est calculé en prenant compte le pixel d'origine ou de tout ou partie des pixels de l'image (filtrage global). Les opérations sur les histogrammes ou les opérations qui nécessitent de passer dans l'espace de Fourier sont des filtres globaux.

1. But de l'activité

→ Développer une classe de filtres d'images

2. Classe de manipulation d'image

Lors de cette activité nous manipulerons des images matricielles, c'est-à-dire représentées par des tableaux de pixels.

Nous utiliserons la bibliothèque Pillow pour s'affranchir de la question des formats de fichiers. Les principales commandes pour manipuler un fichier images sont les suivantes :

Ouverture et enregistrement de fichiers d'image avec Pillow

Le bout de code suivant convertit le fichier `tigre.jpg` (au format JPEG) en `tigre.png` (au format PNG) :

```
import PIL.Image as Image
img = Image.open(r'tigre.jpg')
img.save(r'tigre.png')
```

Informations sur une image

Dans un interpréteur Python, essayez :

```
import PIL.Image as Image
img = Image.open(r'tigre.jpg')
```

puis essayez d'utiliser la documentation interne de python (par exemple via la fonction `dir` de Python ou la commande `help` de l'interpréteur) pour explorer les informations fournies par l'objet image. Essayez par exemple d'obtenir sa taille.

Représentation d'une image en mémoire

Si `img` est une image chargée avec `PIL.Image.open`, on accède à ses pixels via la méthode `img.load()` qui renvoie un tableau indexé par des couples d'entiers (et non pas une matrice au sens python du terme).

Par exemple :

```
pixels = img.load()
print(pixels[0,0])
```

Modifier une image

Pour modifier un pixel, on change sa valeur dans le tableau des pixels :

```
pixels[0,0] = 0
```

La modification de la couleur d'un pixel dans l'objet image `img` s'effectue par la commande :

```
img.putpixel((c,l),(r,v,b))
```



L'affichage de l'image `img` est commandée par :

```
img.show()
```




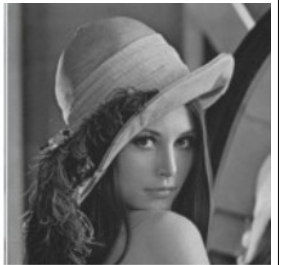
3. Classe filtre

La classe Filtre à développer est décrite par le diagramme de classe suivant :


Cette classe met en œuvre différents traitements dont les caractéristiques sont les suivantes :

Nom de la méthode	Type de filtrage	Description	Filtrage effectué
<code>reverse()</code>	Négatif	Remplace tous les pixels de l'image par leur valeur en négatif. Obtenir le négatif d'une image est très simple : toutes les composantes x de tous les pixels de	 



		l'image sont remplacées par $255 - x$.	
red()	Rouge	<p>Chaque pixel de l'image est une combinaison de rouge, de vert et de bleu.</p> <p>En assignant la valeur 0 aux composantes verte et bleue, on obtient une image à dominante rouge.</p> <p>Écrire une méthode red() qui réalise cette opération.</p>	 
color2grey()	Niveau de gris	<p>L'œil est plus sensible à certaines couleurs qu'à d'autres. Le vert (pur), par exemple, paraît plus clair que le bleu (pur). Pour tenir compte de cette sensibilité dans la transformation d'une image couleur en une image en niveaux de gris, on ne prend généralement pas la moyenne arithmétique des intensités de couleurs fondamentales, mais une moyenne pondérée. Pour simplifier les choses, nous prendrons ici la moyenne « classique ».</p>	 



threshold()	Seuillage	<p>Le seuillage d'image est la méthode la plus simple de segmentation d'image.</p> <p>À partir d'une image en niveau de gris, le seuillage d'image peut être utilisé pour créer une image comportant uniquement deux valeurs, noir ou blanc (monochrome). On remplace un à un les pixels d'une image par rapport à une valeur seuil fixée (par exemple 50). Ainsi, si un pixel à une valeur supérieure au seuil, il prendra la valeur 255 (blanc), et si sa valeur est inférieure, il prendra la valeur 0 (noir).</p>	

4. Travail à effectuer

1. **Créer** un fichier python `filtre.py`.

2. **Écrire** une classe `Filtre` à partir du diagramme de classe présenté page 2

NB : le corps des méthodes ne sera pas développé immédiatement ; on utilisera l'instruction Python `pass` en attendant.

3. **Compléter** le constructeur de classe.

4. **Compléter** les méthodes suivantes et tester au fur et à mesure leur validité :

- `size()` : retourne la taille en pixels d'une image sous forme de tuple largeur, hauteur
- `width()` : retourne la largeur d'une image en pixels
- `height()` : retourne la hauteur d'une image en pixels
- `weight()` : retourne le poids d'une image en pixels



→ `get_pix(x, y)` : retourne la valeur du pixel de coordonnées (x,y), ou None si erreur

5. Compléter les méthodes de traitement d'image. Tester au fur et à mesure chaque méthode

