

## TD1 – Images en couleur, binarisation et filtrage

**Le but** de ce TD est de manipuler et traiter une image à partir d’une bibliothèque OpenCV en Python. Nous allons appliquer les traitements suivants sur les images :

- Inversion des couleurs,
- Binarisation d’images,
- Filtrage.

Pour commencer, téléchargez le fichier [start.py](#) et analysez le code.

### Exercice 1 : Inversion de couleurs d’une image

Utilisez l’image [lena.jpg](#).

- 1) Écrivez le programme pour l’inversion des couleurs dans l’image en niveaux de gris. Sauvegarder l’image obtenue.



Utilisez l’image [circles\\_in\\_a\\_circle.jpg](#)

- 2) Modifiez le programme écrit précédemment pour faire l’inversion des couleurs dans les images en couleurs. Sauvegarder l’image obtenue.



## Exercice 2 : Image RVB

Utilisez l'image [lena.jpg](#).

- 1) Affichez (en niveaux de gris) des canaux vert, bleu, rouge séparément.
- 2) Affichez le canal vert (puis bleu et rouge) en couleur maintenant.



Utilisez l'image [circles\\_in\\_a\\_circle.jpg](#)

- 3) Trouvez l'assemblage des canaux vert, bleu et rouge pour obtenir l'image ci-dessous. Expliquer votre raisonnement.



**Indice pour l'exercice 2 :**

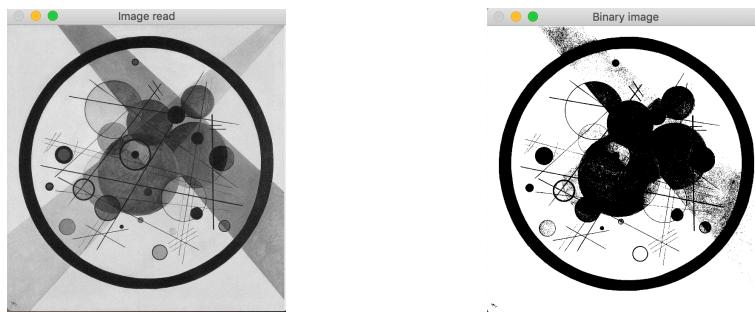
1. Utiliser les fonctions `split` et `merge` de OpenCV.
2. Pour créer la matrice remplis de zéros, utilisez le code suivant :

```
import numpy as np
size = img.shape[0], img.shape[1]
tmp = np.zeros(size, dtype=np.uint8)
```

### Exercice 3 : Binarisation d’une image

Utilisez l’image `circles_in_a_circle.jpg`

1. Écrivez une fonction qui prend en entrée une image en niveaux de gris et un seuil (un nombre entier) et qui binarise l’image en fonction de ce seuil.
  - Affichez l’image résultante
  - Exécutez la fonction avec les différents seuils



2. Utilisez maintenant la fonction de OpenCV qui binarise une image `cv.threshold()`

Pour plus d’information, regardez la documentation de OpenCV :

[https://docs.opencv.org/4.x/d7/d4d/tutorial\\_py\\_thresholding.html](https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html)

### Exercice 4 : Filtrage

- 1) Créer une fonction `flou1()` qui va remplacer, dans l’image de sortie, la valeur d’un pixel par la valeur moyenne de ce pixel avec ses 4 voisins :

$$p_{out}(i, j) = (p(i, j) + p(i - 1, j) + p(i + 1, j) + p(i, j - 1) + p(i, j + 1))/5.$$

Il est fortement recommandé de faire attention aux bords de l’image. En effet, la première et la dernière colonne, ainsi que la première et les dernières lignes ne peuvent pas être traitées, il faudra conserver les valeurs initiales.

- 2) Créer une nouvelle fonction `flou2()` qui va remplacer, dans l’image de sortie, la valeur d’un pixel par la valeur moyenne de ce pixel avec ses 8 voisins.
- 3) Appliquer 3 fois la fonction `filtre_flou2()` sur la même image. Pour cela, avec la même fonction, l’image de sortie devient l’image d’entrée pour l’itération suivante. Comparer les images obtenues.

Utilisez les images suivantes pour les questions ci-dessous : [lena\\_sp1.png](#), [lena\\_sp2.png](#), [lena\\_gauss1.jpg](#), [lena\\_gauss2.jpg](#).

- 4) Implémentez une fonction du filtrage `blur_filtering()` par filtre moyenneur qui utilise la fonction `blur()` de OpenCV et testez la avec les différentes images (avec le bruit poivre-et-sel et avec le bruit gaussien) et les différents noyaux.



- 5) Implémentez une fonction du filtrage `gauss_filtering()` par filtre gaussien (`GaussianBlur()` de OpenCV) et testez cette fonction avec les différentes images et les différents noyaux.



- 6) Implémentez une fonction `median_filtering()` qui applique le filtre médian (`medianBlur()` de OpenCV) sur une image. Comparez une image originale et sa version filtrée.



Pour plus d'information, regardez la documentation de OpenCV :

[https://docs.opencv.org/4.x/d4/d13/tutorial\\_py\\_filtering.html](https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html)

- 7) Implémentez une fonction du filtrage avec le filtre passe-haut suivant qui rendra l'image plus nette. Comparez une image originale et sa version filtrée.

**Indice :** Utiliser la fonction `filter2D()` de OpenCV.

0	-0.5	0
-0.5	3	-0.5
0	-0.5	0