TD-1 – Introduction au traitement d'images

Informatique Graphique et Vision

Les exercices suivants seront réalisés en TD avec le langage Python (3.X), et les bibliothèques NumPy, OpenCV (4.X) et MatPlotlib.

A. Préparation de l'environnement de travail

Installation de la bibliothèque OpenCV

- Ouvrez le terminal « Anaconda Prompt »
- Saisissez le code ci-dessous :

```
pip install opencv-python
```

Lancer un notebook Jupyter

- Ouvrez le terminal « Anaconda Prompt »
- Saisissez le code ci-dessous :

```
jupyter notebook
```

B. Traitement d'images avec OpenCV

Vous trouverez ci-dessous des exemples d'utilisation des principales fonctions OpenCV dont vous aurez besoin afin de réaliser ce TD.

Inclusion des bibliothèques nécessaires à ce TD dans votre script

```
01.  import cv2
02.  import numpy as np
03.  from matplotlib import pyplot as plt
```

Création d'une image

L'exemple ci-dessous montre la procédure pour créer une image noire (pixels à zéro) de taille 320 par 240, avec une précision de 8 bits (*unsigned*).

```
np.zeros([240,320], np.uint8)# 1 canal
np.zeros([240,320, 3], np.uint8)# 3 canaux
```

Lecture d'une image à partir d'un fichier

```
01.  fichier = "crochet.jpg"
02.  img = cv2.imread(fichier, cv2.IMREAD_COLOR)
```

Affichage d'une image

```
01.  cv2.imshow("Titre", img)//la variable img contient une image BGR
02.  cv2.waitKey(0)#a ne pas oublier
```

Convertir une image RGB en niveau de gris

```
imgNG = cv2.cvtColor(mat, cv2.COLOR_RGB2GRAY)
```

Autres conversions possibles : COLOR_RGB2BGR, COLOR_HSV2RGB, *etc.*

Retourner une image

```
flipImg = cv2.flip(img, 0)#0 : retourner verticalement, 1 : horizontalement
```

Redimensionner une image

```
01.  largeur = 320//nouvelle largeur
02.  hauter = 240//nouvelle hauteur
03.  imgR=cv2.resize(imgBase,(largeur,hauter),interpolation=cv2.INTER_AREA)
```

Afficher un histogramme

```
01.  histogramme = cv2.calcHist([img],[i],None,[256],[0,256])
02.  plt.plot(histogramme)
03.  plt.xlim([-10,260])#echelle de l'axe des abscisses
04.  plt.show()
```

Segmentation d'image - Seuillage par valeur d'intensité

Segmenter une image correspond à séparer les régions qui correspondent aux objets que l'on veut analyser. La méthode de seuillage ci-dessous sépare l'avant-plan et l'arrière-plan de l'image en se basant sur l'intensité des pixels. Les pixels avec une intensité supérieure au seuil donné seront considérés comme faisant partie de l'avant-plan, et par conséquent les autres pixels seront affectés à l'arrière-plan.

```
01.  seuil = 100
02.  valeurMaximale = 255
03.  typeSeg = 1 #binaire
04.  _, res = cv2.threshold(image, seuil, valeurMaximale, typeSeg )
```

Types de seuillage par valeur :

- 0: Binary
- 1: Binary Inverted
- 2: Threshold Truncated
- 3: Threshold to Zero
- 4: Threshold to Zero Inverted

Plus d'informations : https://docs.opencv.org/3.4/db/d8e/tutorial_threshold.html

Segmentation d'images - Seuillage par intervalle d'intensité

Le code source ci-dessous montre la procédure pour réaliser une segmentation par intervalle. Dans ce cas, les pixels avec une couleur entre « valeurMinimale » et « valeurMaximale » seront affectés à l'avant-plan, et les pixels restants seront affectés à l'arrière-plan.

```
01.  valeurMinimale = (low_H, low_S, low_V)
02.  valeurMaximale = (high_H, high_S, high_V)
03.  imgRes = cv2.inRange(imgHSV, valeurMinimale, valeurMaximale)
```

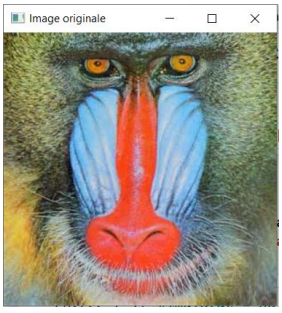
C. Exercices

Exercice 1 – Création d'images

1. Créez une image noire de taille 160 pixels par 120 pixels.
2. Affichez cette image dans une fenêtre OpenCV nommée « Image noir de taille … ».
3. Créez une image blanche de taille 80 pixels par 60 pixels.
4. Affichez cette image dans une fenêtre OpenCV nommée « Image blanche de taille … ».
5. Créez une image verte de taille 30 pixels par 40 pixels.
6. Affichez cette image dans une fenêtre OpenCV nommée « Image verte de taille … ».
7. Déposez votre travail sur la plateforme de cours.

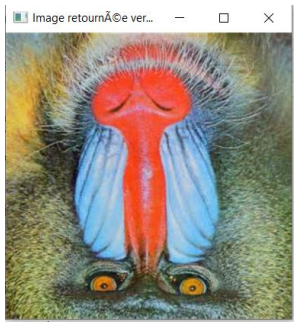
Exercice 2 – Manipulation d'une image

8. Ouvrez l'image « *Singe.jpg* »
9. Affichez cette image dans une fenêtre OpenCV nommée « Image originale ».

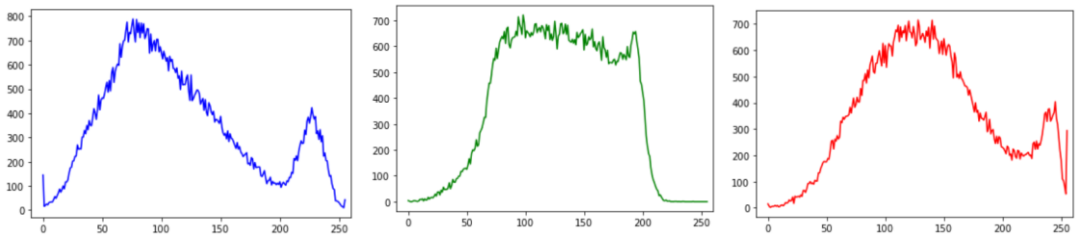


10. Ecrivez une fonction qui redimensionne une image et renvoie le résultat par retour de fonction. L'image à redimensionner ainsi que la largeur et la hauteur à utiliser seront des paramètres de la fonction. Utilisez votre fonction avec l'image chargée précédemment et affichez-la dans une fenêtre « OpenCV » nommée « Image redimensionnée ».

11. Ecrivez une fonction qui retourne une image verticalement. Affichez le résultat de votre fonction dans une fenêtre « OpenCV » nommée « Image retournée ».

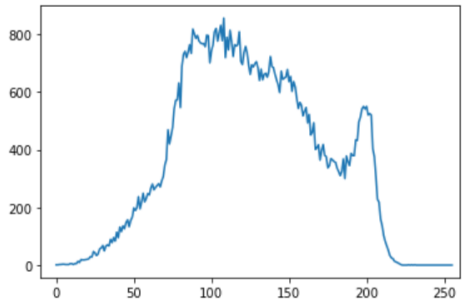


12. Calculez et affichez les histogrammes des différents canaux de votre image.

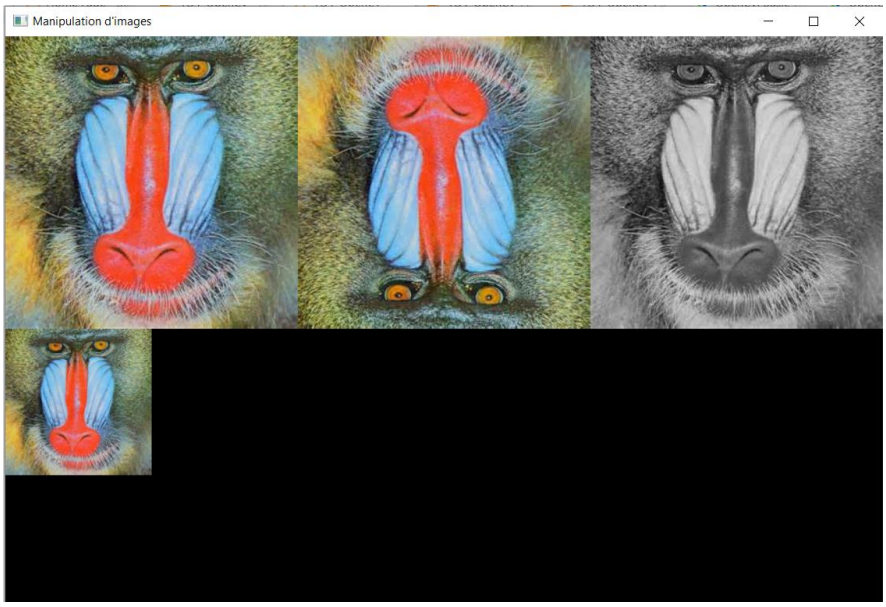


13. Convertissez votre image (en couleurs) en niveau de gris, et affichez le résultat dans une fenêtre nommée « Niveau de gris ».

14. Calculez et affichez l'histogramme de l'image en niveau de gris.



15. Affichez les résultats de vos opérations précédentes dans une seule image/fenêtre. Vous pouvez utiliser le pseudo-algorithme ci-dessous afin de vous guider sur les principales étapes à réaliser.



Pseudo-algorithme :

- Désactivez les affichages intermédiaires réalisés pendant les étapes précédentes (i.e., *imshow*).
- Créez une image en BGR avec une définition de 3x la largeur et 3x la longueur de l'image utilisée pendant ce TD.
- Créez une fonction qui copie les pixels d'une image A vers une image B. L'image B doit être de définition supérieure ou égale à l'image A. Dans ce cas-ci, l'image B sera 3x fois plus grande que l'image A afin de nous permettre d'afficher l'ensemble de notre travail.

> Pensez à rajouter des paramètres de décalage horizontal et vertical comme paramètres de la fonction afin de placer l'image A dans l'image B selon vos besoins.

- Créez une fonction qui copie les pixels d'une image contenant un seul canal dans tous les canaux d'une autre image à trois canaux. Cette procédure nous permettra d'afficher une image en niveau de gris dans une image RGB, par exemple.
- Utilisez les fonctions créées ci-dessus afin de copier l'image originale et les résultats des opérations précédentes vers l'image de grande taille créée.

> **Attention** : Afin d'afficher une image en niveau de gris dans une image BGR, vous pouvez dupliquer l'information de l'image d'origine dans les canaux de l'image de destination.

- Affichez l'image « Résultat » dans une fenêtre nommée « Traitement d'images »

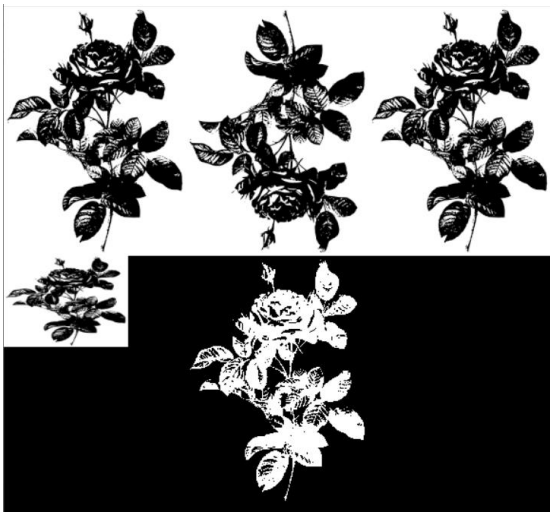
16. Déposez votre travail sur la plateforme de cours

Exercice 3 – Seuillage par valeur

Nous allons maintenant développer une méthode de segmentation basique afin de séparer le fond de l'image de son avant-plan contenant l'objet d'intérêt.

1. Dupliquez votre notebook et renommez-le. Par exemple, « Segmentation »
2. Modifiez votre script afin de charger l'image « fleur.png ».
3. Ecrivez une fonction qui segmente une image en niveau de gris par valeur. L'image en niveau de gris ainsi que le seuil à utiliser seront de paramètres de la fonction, qui doit renvoyer le résultat de cette opération comme retour de la fonction.
4. Appliquez votre fonction à l'image chargée et copiez le résultat dans l'image « Résultat ».

> Vous pouvez regarder l'histogramme de votre image afin d'identifier une valeur de seuil adaptée à la segmentation de l'objet de l'arrière-plan.



5. Déposez votre travail sur la plateforme de cours

Exercice 4 – Seuillage par intervalle

Nous allons maintenant développer une méthode de segmentation par intervalle afin de séparer le fond de l'image de son avant-plan en utilisant l'espace couleur HSV.

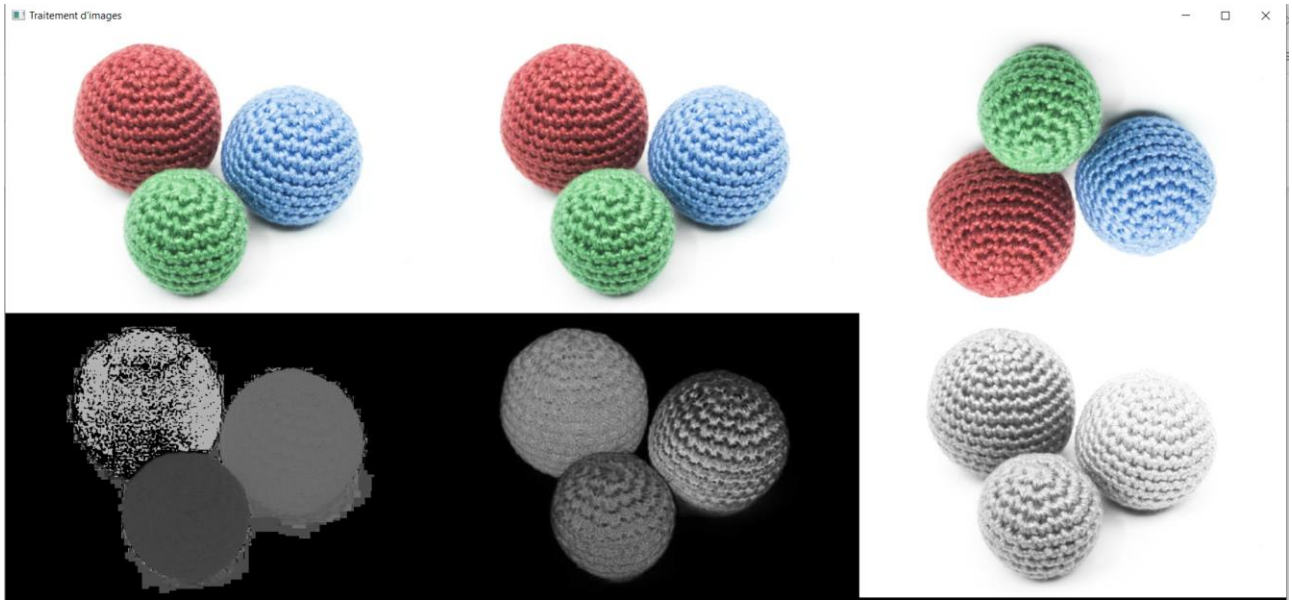
6. Dupliquez votre notebook et renommez le nouveau fichier avec le nom « SegmentationHSV ».
7. Chargez l'image « crochet.jpg ».
8. Affichez votre image dans la *première* case de l'image « Résultat ».
9. Convertissez votre image en HSV.
10. Affichez l'image HSV dans la *deuxième* case de la fenêtre « Résultat ». Obtenez-vous la même chose qu'à l'étape 8 ?

Attention : une image HSV doit être converti en BGR avant d'être visualisée avec « imshow »

11. Affichez la version retournée horizontalement de l'image dans la *troisième* case de l'image « Résultat ».

12. Affichez chaque canal de l'image HSV dans les cases du bas de la fenêtre « Résultat ».

Afin de réaliser cette tâche, vous devez coder une fonction qui copie un canal d'une image à plusieurs canaux (e.g., RGB,HSV, etc.) vers une image avec un seul canal.



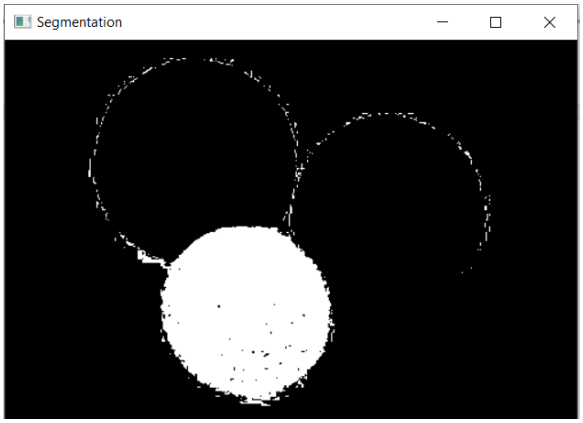
Nous allons maintenant utiliser notre image en HSV afin de segmenter chaque objet de l'image individuellement. Quel canal de l'image HSV choisirez-vous afin de segmenter votre image ?

13. Calculez et affichez les histogrammes de différents canaux de l'image HSV.

14. Ecrivez une fonction qui réalise une segmentation par intervalle.

15. Analysez les images et les histogrammes produits afin d'identifier les intervalles de couleurs qui segmentent chaque boule de crochet. Notez ces valeurs sur votre script en mode commentaire.

16. Utilisez l'intervalle qui produit une segmentation similaire à celle affichée ci-dessous dans votre fonction, et affichez le résultat dans une fenêtre OpenCV.



17. Déposez votre travail sur la plateforme de cours

Défis : filtrage d'artefacts dans l'image

Vous remarquerez que nous sommes capables de segmenter les objets de l'image précédente sans beaucoup de difficulté dans l'espace HSV. Par contre, plusieurs artefacts persistent. Vous pouvez utiliser différentes techniques de filtrage afin d'essayer de les supprimer. Dans le cadre de cet exercice, nous vous proposons d'exploiter les opérations morphologiques à cette fin :

Source : https://dpt-info.u-strasbg.fr/~cronse/TIDOC/MM/deof.html

i. Opérations d'érosion

```
elementStructurant = np.ones((taille_element, taille_element),np.uint8)
resultat_erosion = cv2.erode(img, elementStructurant)
```

ii. Opérations de dilatation

```
elementStructurant = np.ones((taille_element, taille_element),np.uint8)
resultat_erosion = cv2.dilate(img, elementStructurant)
```

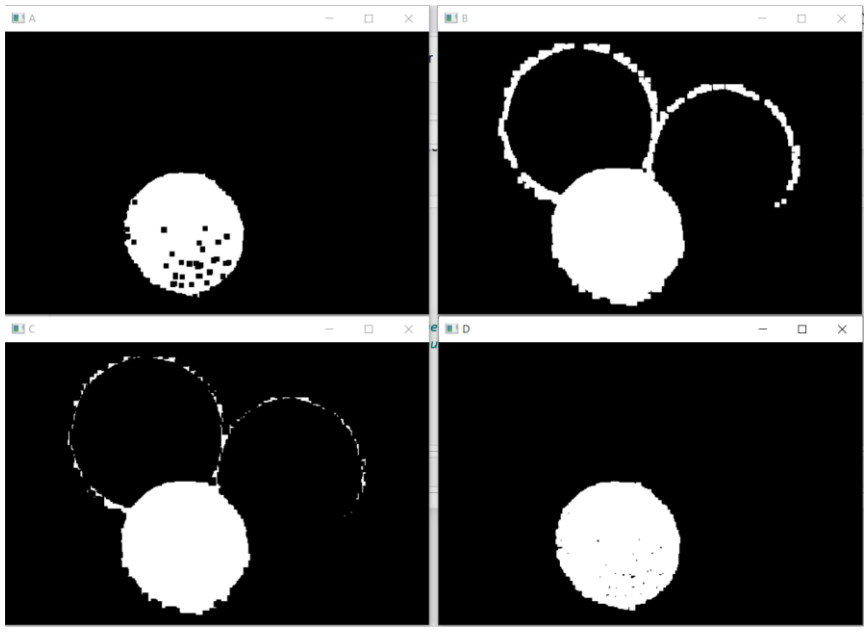
1. Ecrivez des fonctions qui utilisent les opérations d'érosion et de dilatation à une image.

2. Appliquez séparément ces fonctions au résultat de la segmentation précédente et affichez les résultats dans des fenêtres OpenCV.

3. Etudiez l'influence du paramètre « taille de l'élément structurant » sur le résultat (*e.g.*, 5, 7 et 9). Que pouvez-vous conclure à propos de ces opérations et de l'influence de ce paramètre ?

4. Ecrivez des fonctions qui réalisent les opérations d'ouverture (érosion puis dilatation) et de fermeture (dilatation puis érosion) sur une image (binaire). Appliquez ces fonctions séparément au résultat de la segmentation et affichez les résultats.

5. Comparez les résultats des quatre fonctions développées et identifiez celle qui vous semble la plus adaptée afin de minimiser l'influence des artefacts dans notre image.



6. Déposez votre travail sur la plateforme de cours.