TP1 : Traitement d'images Numériques

Affichage et conversion d'images, Opérations sur les images
Histogramme, amélioration du contraste, seuillage

Matière : Imagerie Embarquée Niveau : IIA4
Enseignant responsable : Mme Ben Saïd Salma

Le premier objectif de ce premier TP est de prendre en main les outils de traitement d'images les plus classiques à l'aide du logiciel OpenCV sous Python.

Le deuxième objectif est de générer l'histogramme d'une image numérique afin d'analyser sa distribution lumineuse et d'améliorer son contraste. Seuiller une image en se basant sur l'histogramme pour le choix du seuil, en utilisant aussi des fonctions prédéfinies OpenCV.

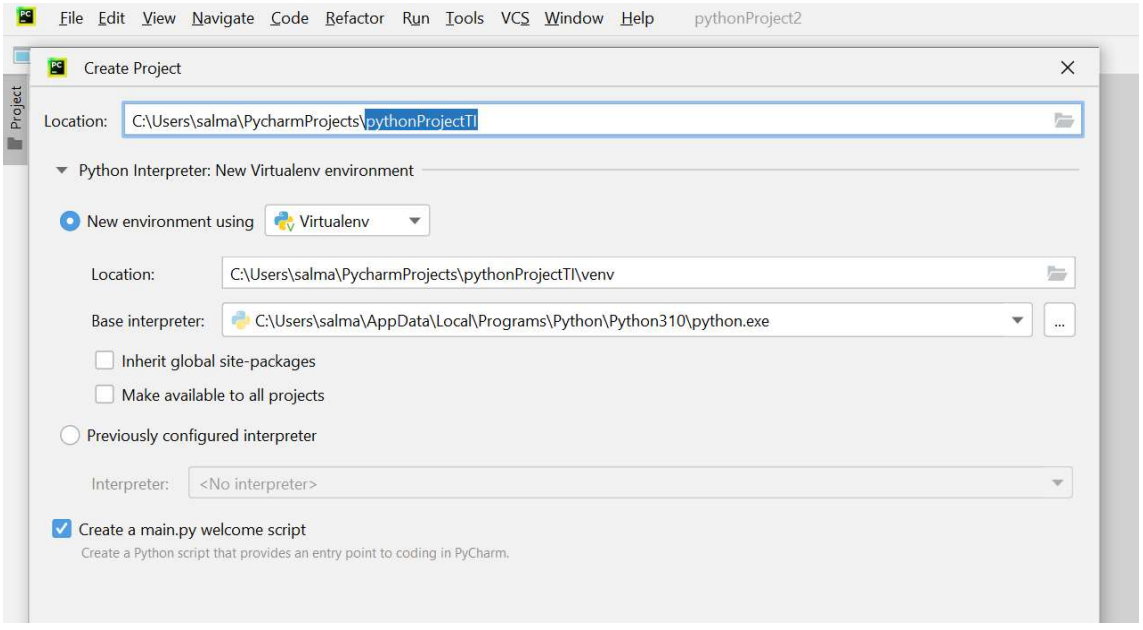
I. Présentation de OpenCV

OpenCV signifie Open Source Computer Vision Library. Il s'agit d'une bibliothèque open source gratuite utilisée pour la vision par ordinateur. Il offre un bon support en Machine Learning, Face Recognition, Deep Learning, etc. Cette bibliothèque est écrite en C/C++ optimisé et est prise en charge sous Windows, Linux, Android et MacOS.

https://download.jetbrains.com/python/pycharm-community-2022.2.2.exe

II. Création d'un projet PyCharm

Aller dans File puis New Project, la fenêtre suivante s'affichera :



Nommer le projet à créer, dans ce cas pythonProjectTI.

Installation des bibliothèques OpenCV et matplotlib sur Python. matplotlib est une bibliothèque Python capable de produire des graphes de qualité.

Les commandes : pip install opencv-python
pip install matplotlib
à exécuter sous la fenêtre **terminal** de PyCharm.

III. Chargement, affichage et sauvegarde d'une image

Les types de fichiers suivants sont pris en charge dans la bibliothèque OpenCV :

- Bitmaps Windows – *.bmp, *.dib
- Fichiers JPEG – *.jpeg, *.jpg
- Graphiques réseau portables – *.png
- WebP – *.webp
- Rasters solaires – *.sr, *.ras
- Fichiers TIFF – *.tiff, *.tif
- Données géospatiales matricielles et vectorielles prises en charge par GDAL

Créer un fichier nommé load_display_save.py dans lequel vous allez écrire un script qui permet de charger, afficher et sauvegarder une image. En utilisant les fonctions imread, imshow et imwrite de la bibliothèque OpenCV.
Exemple :

```
import cv2
# image path
path = r'c:\imagesTP\nature.png'

# using imread()
img = cv2.imread(path)

# displaying the image
cv2.imshow('image', img)

# save image
imwrt = cv2.imwrite(r'c:\imagesTP\natureResult.jpg', img)
cv2.imshow('imageSaved', img)
if imwrt:
    print('Image is successfully saved.')
cv2.waitKey(0);
cv2.destroyAllWindows();
cv2.waitKey(1)
```

IV. Image matricielle

Une image matricielle sous OpenCV est représentée par une matrice. L'élément de base appelé pixel représente simultanément un point de l'image et un élément de la matrice. Les pixels sont accessibles par leurs coordonnées (x, y), Par exemple : soit une image nommée I, sachant que Python est zéro indéxé, I[3,4] donne la valeur du pixel qui se trouve à la 4ème ligne et la 5ème colonne de l'image I.

Dimensions et redimensionnement d'une image :

```
import cv2

# image path
path = r'c:\imagesTP\shape1.png'

# using imread()
img = cv2.imread(path)

# get dimensions of image
dimensions = img.shape

# height, width, number of channels in image
height = img.shape[0]
width = img.shape[1]
channels = img.shape[2]

print('Image Dimension    : ', dimensions)
print('Image Height       : ', height)
print('Image Width        : ', width)
print('Number of Channels : ', channels)
# Total number of pixels on the image
sizeImage=img.size
print('number of pixels   : ', sizeImage)
# reading image to be resized
path2 = r'c:\imagesTP\natureResult.jpg'
# using imread()
img2 = cv2.imread(path2)
# resize image then display
img2Res=cv2.resize(img2,(width,height))
cv2.imshow('img2Res', img2Res)

cv2.waitKey(0);
cv2.destroyAllWindows();
cv2.waitKey(1)
```

Définir une région d'intérêt : x rang du pixel en ligne et y rang du pixel en colonne.

```
ROI=img[xi:xj,ya:yb] avec i<j et a<b ;
```

Profil d'intensité

La fonction ***profile_line*** dans la bibliothèque `skimage` calcule et affiche l'intensité d'une image I le long d'un segment (horizontal ou vertical) partant d'un pixel de départ D de coordonnées x et y jusqu'à un pixel d'arrivé A de coordonnées x' et y'.

```
from skimage.measure import profile_line

…

p = profile_line(I, (Dx, Dy), (Ax', Ay')) plt.plot(p)

plt.show()
```

V. Espaces de couleurs et Conversion

Il existe quatre types d'images : Image en niveaux de gris, Image binaire, Image couleur vraie(RGB) et Image couleur indexée. Les couleurs de l'image couleur indexée peuvent être changées en changeant la carte de couleur qui lui est associée.
Syntaxe:

image=cv2. applyColorMap (imagesource, paletteCcouleurSouhaitée)

Les espaces colorimétriques sont un moyen de représenter les canaux de couleur présents dans l'image qui donnent à l'image cette teinte particulière. Certains des espaces colorimétriques populaires sont RVB (rouge, vert, bleu), CMJN (cyan, magenta, jaune, noir), HSV (teinte, saturation, valeur) .
Syntaxe:

imageEspaceConertie=cv2.cvtColor(imagesource, espaceCouleurConversion)

Exemple :

```
import cv2

# image path
path = r'c:\imagesTP\shape2R.jpg'
# using imread()
imgI = cv2.imread(path)
dim=(640,360)
img=cv2.resize(imgI,dim)
#changing colormap for idexed color image
im1 = cv2.applyColorMap(img, cv2.COLORMAP_AUTUMN)
im2 = cv2.applyColorMap(img, cv2.COLORMAP_BONE)
# Display
cv2.imshow('image', np.hstack((im1, im2)))

#Converting color space
imagegray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
imagehsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV )
imageyuv = cv2.cvtColor(img,cv2.COLOR_BGR2YUV)
imagecmyk = cv2.cvtColor(img,cv2.COLOR_BGR2YCR_CB)
# Displaying
cv2.imshow('gray', imagegray)
cv2.imshow('hsv', imagehsv)
cv2.imshow('YUV', imageyuv)
cv2.imshow('CMYK', imagecmyk)

cv2.waitKey(0);
cv2.destroyAllWindows();
cv2.waitKey(1)
```

VI. Opérations sur les images

Les opérations arithmétiques telles que l'addition, la soustraction et les opérations au niveau du bit (AND, OR, NOT, XOR) peuvent être appliquées aux images d'entrée. Les deux images doivent être de taille et de profondeur égales.

Opérations arithmétiques sur les images

Opérer les fonctions arithmétiques d'addition, de soustraction et de multiplication par un ratio sur des images de votre choix. Afficher le résultat pour chaque opération.

```
cv2.add(image1, image2)
cv2.subtract(image1, image2)
```

Opérations logiques bit à bit sur les images binaires

Créer deux images binaires en créant deux matrices `img1`et `img2` de définition `300*500`, initialisées à zéro et telles que :

```
img1[100:160,200:260]=1
img2[140:220,220:300]=1
```

Opérer les fonctions logiques **and**, **or**, **xor**, **not** sur ces images et afficher le résultat.

```
resultat_or = cv2.bitwise_or(img2, img1)
```

VII. Histogramme

L'histogramme d'une image est la distribution des intensités dans cette image, il s'agit du nombre de pixels par niveau de gris.

La fonction intégrée dans Open CV cv2.calcHist() et la fonction hist() d'un array numpy permettent de calculer l'histogramme d'une image.

```
cv2.calcHist(images, channels, mask, histSize, range=[0,256])

plt.hist(n_img.ravel(), bins=256, range=[0,256])


Exemple :
```

```
f1=plt.figure(1)
plt.title('histogramme')
plt.hist(I.ravel(),256,[0,25
6]); plt.show()
```

```
%charger l'image pout.tif dans une variable I.
```

- Calculer et afficher l'histogramme en utilisant la fonction plt.hist().
- Déterminer le maximum de la distribution de l'intensité.
- Afficher l'histogramme de l'image I à l'aide de 32, 64 et 128 bins.
- Déterminer le maximum de la distribution de l'intensité dans chaque cas.

Egalisation de l'histogramme

L'égalisation de l'histogramme permet de distribuer uniformément le nombre de pixel par niveau de gris. Elle permet d'améliorer le contraste dans une image donnée.

La fonction `cv2.equalizeHist(I)` effectue l'égalisation de l'image I.

- Appliquer la fonction d'égalisation sur l'image pout.tif , afficher l'image obtenue et son histogramme.
- Quelles différences remarquez-vous entre les deux images et leurs histogrammes respectifs.

```
%charger une image de votre choix dans une variable I1.
```

- Afficher son histogramme. Interpréter.
- Eclaircir l'image I1 et l'enregistrer dans une matrice I2.
- Assombrir l'image I1 et l'enregistrer dans une matrice I3.
- Améliorer le contraste de chacune de ces images par l'opération d'égalisation de l'histogramme. Afficher le résultat. Interpréter.

Seuillage et binarisation

A. Le seuillage d'une image consiste à considérer les pixels de l'image dont la valeur est supérieure (ou inférieure) à un seuil. Une image binaire peut être obtenue par seuillage en remplaçant tout pixel dont la valeur est supérieure (ou inférieure) au seuil.

Pour un seuillage simple, l'histogramme peut être utilisé.

- Charger l'image coins.png en mémoire
- Utiliser l'histogramme pour choisir un seuil de façon à isoler les pièces de monnaie.
- Utiliser la fonction suivante pour effectuer le seuillage :

cv2.threshold(source, thresholdValue, maxVal, thresholdingTechnique)

- Afficher l'image coins, l'histogramme et l'image binarisée.

B. Le seuillage adaptatif est la méthode dans laquelle la valeur de seuil est calculée pour des régions plus petites. Cela conduit à différentes valeurs de seuil pour différentes régions en ce qui concerne le changement d'éclairage.

cv2.adaptiveThreshold(source, maxVal, **adaptiveMethod**, thresholdType)

cv2.ADAPTIVE_THRESH_MEAN_C : Valeur seuil = (Moyenne des valeurs de la zone de voisinage – valeur constante). En d'autres termes, c'est la moyenne du voisinage blockSize×blockSize d'un point moins une constante.

cv2.ADAPTIVE_THRESH_GAUSSIAN_C : Valeur seuil = (Somme pondérée gaussienne des valeurs de voisinage – valeur constante). En d'autres termes, il s'agit d'une somme pondérée du voisinage