Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université M'hamed Bougara de Boumerdès
Faculté des Sciences, Département : Informatique

TP 1 ► Module : TAI	**Année :** S1-Master I2A Pro.	GACEB

L'objectif de ce TP est de vous faire découvrir quelques bibliothèques Python spécialisées en vision par ordinateur, ainsi que l'IDE PyCharm qui sera utilisé pour réaliser les TP de ce module. Les travaux effectués vous permettront de vous initier à la représentation d'une image numérique en mémoire, d'apprendre à ouvrir, modifier (manipuler au niveau pixel), afficher et sauvegarder une image..

Exercice 1 : Installation de L'IDE Pycharm et création d'un nouveau projet python

Les codes qui seront développés durant toutes les séances de TP seront réalisés sur l'IDE PyCharm Community Edition en utilisant des bibliothèques spécialisées pour manipuler des images. Cet IDE est un environnement de développement intégré gratuit et open source conçu spécifiquement pour répondre aux besoins des développeurs Python de tous bords (sur Linux, Windows et MacOs). C'est un outil excellent pour le prototypage, facile d'utilisation pour les débutants en développement, tout en étant assez robuste pour satisfaire les exigences des développeurs chevronnés.
La toute première démarche, consiste à télécharger et installer Pycharm sur votre machine (windows) selon les étapes suivantes :

1. Télécharger l'installateur depuis la section de téléchargement de PyCharm sur le site de JetBrains et choisir la version Community.
   https://download.jetbrains.com/python/pycharm-community-2025.2.2.exe
2. Lancer l'installateur pycharm-community-2025.2.2.exe, Cliquer sur Next, choisir l'endroit où vous souhaitez installer le logiciel et cliquer à nouveau sur Next.
3. Suivre l'assistant en cochant toutes les options : comme "Open Folder as Project", l'ajout au PATH, etc.) et cliquer à nouveau sur Next, puis sur Install. PyCharm va maintenant s'installer. Une fois l'installation terminée, cliquer sur Finish.

La seconde démarche consiste à créer votre premier projet sur PyCharm :

1. Lancer PyCharm sur votre ordinateur.
2. Créer un nouveau projet : Sur l'écran de bienvenue ou à partir du menu, cliquer sur "New Project" (Nouveau projet).
3. Une nouvelle fenêtre de configuration s'ouvre : Dans le champ "Location" (Emplacement), spécifier le chemin où le projet sera enregistré et donner un nom à votre projet (Dans votre cas TP_TAI_Nom_Pénom_Groupe).
4. Version de Python : Assurez-vous que la version de Python sélectionnée dans les options de Virtualenv est bien celle que vous souhaitez utiliser.
5. Lancer la création : Cliquez sur le bouton "Create" pour générer le projet. PyCharm va créer le dossier du projet et l'environnement virtuel.
6. A partir du menu ou en cliquant droite sur le nom de projet (menu contextuel), cliquer sur « New » → « Python File », puis saisir dans Name « TP1_Exo1 » et Entrer. Le fichier TP1_Exo1.py est à présent ajouté à votre projet. Une fois le projet créé, vous pouvez commencer à écrire votre code Python dans le nouveau fichier créé.
7. Décompresser le dossier Images, ci-joint, copier le dans le dossier de votre projet.

Exercice 2 : Manipulations d'une image et bibliothèques spécialisées

Pour manipuler une image, plusieurs packages python existent. Pour commencer, on utilise la bibliothèque ***OpenCV-Python***. Cette dernière est un wrapper Python pour ***OpenCV*** (pour Open Computer Vision), une bibliothèque logicielle open source développée initialement par Intel pour faire la vision par ordinateur et l'apprentissage automatique. Elle fournit un ensemble complet d'outils pour diverses tâches de vision par ordinateur, tirant parti de la simplicité et de la lisibilité de Python tout en bénéficiant des performances de l'implémentation C++ sous-jacente. Dans ***OpenCV-Python***, les images sont fondamentalement représentées sous forme de tableaux ***NumPy***, où chaque élément du

tableau correspond à un pixel. Il est essentiel de comprendre comment accéder à chaque pixel et le manipuler pour effectuer diverses tâches de traitement d'images.

1. Avant de commencer, vérifier si les packages NumPy, OpenCV-Python sont installés dans votre environnement : File > Settinges > Interpreter de votre projet, en cherchant dans la liste des packages déjà installer. Si le package n'existe pas, cliquer sur le bouton +, puis en recherchant et installant le package désiré.
2. Dans le même projet, créer un fichier Python sous le nom « TP1_Exo2.py », puis saisir les codes suivants.
3. ***Lecture et affichage d'images :*** Les images sont lues à l'aide de cv2.imread() et peuvent être affichées à l'aide de cv2.imshow(). cv2.waitKey(0) est utilisé pour attendre une pression sur une touche, et cv2.destroyAllWindows() ferme toutes les fenêtres OpenCV. Saisir et executer dans TP1_Exo2.py le code suivant :

```
import cv2
# Lecture d'une image
img = cv2.imread('Images/BoatsColor.bmp')

# Charger une image couleur avec sa conversion en niveaux de gris
gray_img = cv2.imread('Images/BoatsColor.bmp', cv2.IMREAD_GRAYSCALE)
# Transformer une image couleurs en une image en niveaux de gris
gray_img2 = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Affichage d'une image
cv2.imshow('Input Image', img)
cv2.imshow(' Image chargée en niveau de gris', gray_img)
cv2.imshow(' Image tranformée en niveau de gris', gray_img2)
cv2.waitKey(0)

# Lecture des couleurs d'un pixel à la position at (100, 150)
pixel = img[100, 150]
print(f"Les valeurs RGB de Pixel à (100, 150) sont : {pixel}")

""" Lecture des dimensions des images : dimensions = img.shape
Pour une image couleur, dimensions contient trois valeurs :
(hauteur, largeur, canaux).  hauteur : nombre de lignes de pixels dans
l'image. largeur : nombre de colonnes de pixels dans l'image. canaux :
nombre de canaux de couleur (par exemple, 3 pour RGB, 1 pour les niveaux de
gris). Pour une image en niveaux de gris, dimensions contient deux valeurs
: (hauteur, largeur)"""

height, width, channels = img.shape
print(f"Height: {height} pixels")
print(f"Width: {width} pixels")
print(f"Channels: {channels}")

#Modification des couleurs de pixel de la ligne de centre vers une couleur rouge
for i in range(0, width-1):
    img[int (height/2), i] = [0, 0, 255] # [B,G,R] modifier en rouge
cv2.imshow('Image modifiée', img)

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

Exercice 3 : Seuillage fixe d'une image

Dans un nouveau fichier « TP1_Exo3.py », On souhaite coder la binarisation de l'image « compteur.jpg ». Tous les pixels de NG >80 sont convertis en Blanc et les pixels de NG≤80 sont convertis en noir. Pour cela, développer un code permettant d'obtenir l'image binaire dans imgb1 en utilisant les fonctions openCV et l'image binaire imgb2 en utilisant votre code personnel. Enregistrer imgb1 dans le dossier 'Images/imgb1.jpg'.

Exercice 4 : Manipulations simples sur des images numériques

Dans un nouveau fichier *«TP1_Exo4.py »*, On souhaite faire diverses manipulations sur l'image *im1*, lue à partir du fichier « Images/pepper.bmp ». Implémenter les instructions suivantes :

1) Lire l'image *im1*
2) Lire la taille de l'image *im1* dans *h* et *w*
3) Redimensionner l'image *im1* en divisant sa taille sur 2, stoker le résultat dans l'image im2
4) Cropper dans *im3* la région de l'image *im1* délimitée par le rectangle: *[y_start=30:y_end =150, x_start=200:x_end =400]*.
5) Copier *im1* dans *im4*, dessiner dans *im4* un rectangle bleu dans la zone *[y_start=30:y_end =150, x_start=200:x_end =400]*, puis saisir sur *im4* le texte suivant :« *Mon rectangle* » en couleur verte à la position *[x=200, y=20]*.
6) Appliquer à *im1* une rotation de 90°, charger le résultat dans *im5*
7) Appliquer un flou Gaussien à l'image *im1*, stoker le résultat dans *im6*,
8) Afficher toutes les images créées

Exercice 5 : Histogramme d'une image

Un histogramme d'une image en niveaux de gris est une représentation graphique qui illustre la répartition de l'intensité lumineuse d'une image, en spécifiant le nombre de pixels présents pour chaque niveau de gris (de 0 à 255). Il facilite l'évaluation de la qualité de l'image (sur-exposition, sous-exposition) et la modification de son contraste en étendant ou comprimant l'histogramme (Égalisation d'histogramme).

Un histogramme d'une image en couleur est un diagramme qui représente la distribution des couleurs ou des intensités lumineuses d'une image en comptant le nombre de pixels pour chaque couleur ou niveau tonal. Il se compose de trois diagrammes, chacun représentant la proportion de pixels pour les composantes rouges, vertes et bleues (RVB) de l'image, indiquant quels tons sont dominants. Il permet d'analyser rapidement la prédominance des couleurs et d'identifier les zones sous-exposées (pic à gauche) ou surexposées (pic à droite) dans chaque canal. En analysant les histogrammes de chaque couleur, il est possible d'identifier les zones de perte d'information et de corriger l'exposition ou le contraste lors du traitement de l'image.

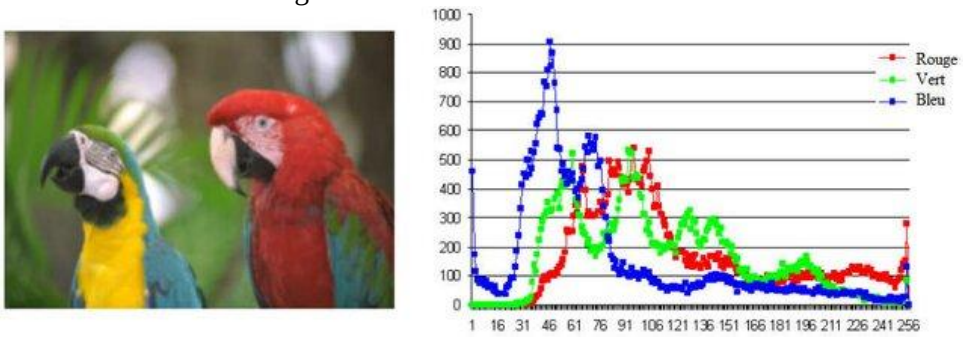

Figure 1. Histogrammes d'une image en couleurs

Tâche à accomplir : Créer un nouveau fichier python «TP1_Exo5.py», puis implémenter les instructions ci-dessous :

1) Lire l'image en niveaux de gris *«Images/cameraman.bmp »* dans *im1* et l'image en couleurs *« Images/pepper.bmp»* dans *im2*
2) Utiliser la méthode *cv2.calcHist()* pour effectuer les calculs d'histogrammes des deux images (*hist1 pour im1 et hist2_B, hist2_G et hist2_R pour im2*).
3) Utiliser le package *matplotlib* pour afficher tous les histogrammes
4) Développer votre propre algorithme pour calculer les histogrammes des deux images (*histp1 pour im1 et histp2_B, hispt2_G et histp2_R pour im2*), puis vérifier que les valeurs de ces histogrammes correspondent à celles obtenues dans la question 2.