TP1 : Opérations de base sur les images avec OpenCV

Objectifs :

- Charger une image
- Accéder aux dimensions, types, canaux
- Accéder et modifier les valeurs des pixels
- Découper, redimensionner, pivoter
- Convertir en niveaux de gris
- Appliquer des transformations simples
- Découper et fusionner des images

1. Chargement et propriétés de l'image

```
import cv2

# Charger l'image
img = cv2.imread(r'I:\Mon Drive\ENSEIGNEMENT SUPERIEUR ODC\Informatique\Python\paysage.jpg')

# Vérification du chargement
if img is None:
    print("Image non chargée.")
else:
    # Dimensions
    print("Dimensions :", img.shape)      # (hauteur, largeur, canaux)
    print("Taille en pixels :", img.size)
    print("Type de données :", img.dtype)

    cv2.imshow("Image originale", img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

- img.shape donne (hauteur, largeur, 3) pour une image couleur.
- img.size donne le **nombre total de pixels x 3 (si RGB)**.
- img.dtype : le **type** des pixels (uint8 = 0 à 255).

2. Accès et modification des valeurs de pixels

Pour accéder à la valeur d'un pixel, il suffit de fournir ses coordonnées (x, y) comme suit :
`pixel_value = image[y, x]`
La valeur du pixel est un tableau contenant les canaux de couleur (B, G, R). Pour modifier la valeur d'un pixel, il suffit d'affecter une nouvelle valeur aux coordonnées correspondantes.
Voici un exemple pour définir le pixel [0:100, 0:100] en rouge :

```
if img is None:
    print("Image non chargée.")
else:
    for x in range(100):
```

```
    for y in range(100):
        img[x, y] = [0, 0, 255]  ## Définit le pixel en rouge

cv2.imshow("Pixels en rouge", roi)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

2. Découper une région de l'image (ROI)

```
if img is None:
    print("Image non chargée.")
else:
    # Découper une zone : y1:y2, x1:x2
    roi = img[100:300, 150:350]

    cv2.imshow("Région découpée (ROI)", roi)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

- Utilise la **notation numpy** [ligne, colonne] pour découper un rectangle.
- C'est utile pour extraire un visage, un objet, etc.

---

3. Redimensionnement (resize)

```
# Vérification du chargement
if img is None:
    print("Image non chargée.")
else:
    resized = cv2.resize(img, (200, 200))  # Largeur x Hauteur

    cv2.imshow("Image redimensionnée", resized)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

- `cv2.resize()` change la taille de l'image à la dimension souhaitée.
- Peut être utilisé pour uniformiser la taille des images en entrée.

4. Rotation de l'image

```
# Vérification du chargement
if img is None:
    print("Image non chargée.")
```

```
else:
    # Rotation de 90 degrés dans le sens des aiguilles d'une montre
    rotated = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)

    cv2.imshow("Image pivotée", rotated)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

---

5. Conversion en niveaux de gris

```
# Vérification du chargement
if img is None:
    print("Image non chargée.")
else:
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    cv2.imshow("Image en niveaux de gris", gray)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

- `cv2.cvtColor(..., cv2.COLOR_BGR2GRAY)` : convertit l'image couleur en **niveaux de gris** (0 à 255).
- Requis pour plusieurs algorithmes comme Canny, Hough, etc.

---

6. Redimensionnement proportionnel (zoom)

```
# Vérification du chargement
if img is None:
    print("Image non chargée.")
else:
    # Zoom x2
    zoomed = cv2.resize(img, None, fx=2.0, fy=2.0, interpolation=cv2.INTER_LINEAR)

    cv2.imshow("Zoom x2", zoomed)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

- `fx` et `fy` : facteurs d'échelle sur l'axe X et Y.
- `INTER_LINEAR` : interpolation bilinéaire (qualité moyenne).

---

7. Symétrie (miroir)

```
# Vérification du chargement
if img is None:
    print("Image non chargée.")
else:
    # Image retournée horizontalement
    flip_horizontal = cv2.flip(img, 1)

    # Image retournée verticalement
    flip_vertical = cv2.flip(img, 0)

    cv2.imshow("Symétrie horizontale", flip_horizontal)
    cv2.imshow("Symétrie verticale", flip_vertical)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

- cv2.flip(img, 1) : miroir gauche/droite.
- cv2.flip(img, 0) : miroir haut/bas.

---

8. Découpage et fusion d'images

Pour découper une image en ses canaux de couleur (B, G, R), utilisez la fonction `cv2.split()` :

```
if img is None:
    print("Image non chargée.")
else:
    b, g, r = cv2.split(img)

    cv2.imshow("Cannal Bleu", b)
    cv2.imshow("Cannal Vert", g)
    cv2.imshow("Cannal Rouge", r)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

Pour fusionner les canaux en une seule image, utilisez la fonction `cv2.merge()` :

```
if img is None:
    print("Image non chargée.")
else:
    b, g, r = cv2.split(img)

    merged_image = cv2.merge((b, g, r))
    cv2.imshow("Cannal Bleu", merged_image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

Par exemple, pour échanger les canaux rouge et bleu, on peut faire ce qui suit :

```
if img is None:
    print("Image non chargée.")
```

```
else:
    b, g, r = cv2.split(img)

    merged_image = cv2.merge((r, g, b))
    cv2.imshow("Canal Bleu", merged_image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

Résumé des opérations :

Action	Fonction OpenCV
Charger une image	cv2.imread()
Afficher une image	cv2.imshow()
Accès aux pixels	img[y, x]
Découper une zone	img[y1:y2, x1:x2]
Redimensionner	cv2.resize()
Convertir en gris	cv2.cvtColor(..., GRAY)
Tourner (90°)	cv2.rotate()
Retourner (miroir)	cv2.flip()
Découpage des canaux de couleurs	cv2.split()
Fusionner les canaux	cv2.merge()