**Introduction**

Ce code est un exemple de traitement d'image en Python utilisant OpenCV et Matplotlib. Il charge une image en niveaux de gris, applique une binarisation avec un seuil spécifique, et affiche les images originale et binarisée côte à côte.

**Bibliothèques Utilisées, nous avons**

* cv2 : OpenCV, une bibliothèque de traitement d'image.
* numpy : NumPy, utilisée ici pour les opérations sur les matrices (importée mais non utilisée dans ce script).
* matplotlib.pyplot : Matplotlib, utilisée pour l'affichage des images.

**Fonctions**

**Fonction im2bw**

def im2bw(image, threshold=100):

"""

Binarise l'image en utilisant un seuil spécifié.

Paramètres :

- image : ndarray

L'image à binariser, typiquement en niveaux de gris.

- threshold : int, optionnel

Le seuil de binarisation. Par défaut, il est fixé à 100.

Retour :

- binary\_image : ndarray

L'image binarisée, où les pixels au-dessus du seuil sont mis à 255 (blanc)

et les pixels en dessous du seuil sont mis à 0 (noir).

"""

# Applique une binarisation à l'image

\_, binary\_image = cv2.threshold(image, threshold, 255, cv2.THRESH\_BINARY)

return binary\_image

* **Description** : Cette fonction applique une binarisation à une image en niveaux de gris en utilisant un seuil spécifié. Les pixels avec une intensité supérieure ou égale au seuil sont convertis en blanc (255) et les autres en noir (0).
* **Paramètres** :
  + image : L'image à binariser.
  + threshold : Le seuil de binarisation (par défaut 100).
* **Retour** : L'image binarisée.

**Fonction show**

def show(original\_image, binary\_image):

"""

Affiche l'image originale et l'image binarisée côte à côte.

Paramètres :

- original\_image : ndarray

L'image originale en niveaux de gris.

- binary\_image : ndarray

L'image binarisée obtenue après application de la fonction im2bw.

"""

# Affiche l'image originale et l'image binaire côte à côte

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

plt.title('Image originale')

plt.imshow(original\_image, cmap='gray')

plt.axis('off')

plt.subplot(1, 2, 2)

plt.title('Image binaire')

plt.imshow(binary\_image, cmap='gray')

plt.axis('off')

plt.show()

* **Description** : Cette fonction affiche l'image originale et l'image binarisée côte à côte en utilisant Matplotlib.
* **Paramètres** :
  + original\_image : L'image originale en niveaux de gris.
  + binary\_image : L'image binarisée.

**Script Principal**

# Charger l'image en niveaux de gris

img = cv2.imread('images/image.tif', 0)

# Vérifier si l'image a été correctement chargée

if img is None:

print("Erreur : Impossible de charger l'image.")

else:

# Binarisation de l'image

img\_binari = im2bw(img, 100)

# Afficher les images

show(img, img\_binari)

* **Description** : Ce script principal charge une image en niveaux de gris à partir du chemin spécifié, vérifie si l'image a été correctement chargée, applique la binarisation en utilisant la fonction im2bw, et affiche les images originale et binarisée côte à côte en utilisant la fonction show.

**Étapes du Script Principal**

1. **Chargement de l'image** : L'image est chargée en niveaux de gris depuis le chemin spécifié.
2. **Vérification du chargement** : Le script vérifie que l'image a été correctement chargée. Si ce n'est pas le cas, un message d'erreur est affiché.
3. **Binarisation** : Si l'image est chargée avec succès, la binarisation est appliquée en utilisant la fonction im2bw.
4. **Affichage** : Les images originale et binarisée sont affichées côte à côte en utilisant la fonction show.

**II-2-Remarque :**

Calcul de l'Histogramme: Après la binarisation, l'histogramme de l'image binarisée aura seulement deux valeurs significatives correspondant aux pixels noirs et blancs. L'histogramme est un outil puissant pour analyser la distribution des pixels dans une image binarisée. Il peut être calculé en utilisant la fonction `np.histogram` de NumPy ou `cv2.calcHist` de OpenCV, par exemple**.**