Université Constantine 2 – Abdelhamid Mehri

Faculté des Nouvelles technologies de l’Information et de la Communication

Département : Informatique Fondamentale et ses Application

Master Sciences et Technologie de l’Infoarmation et de la Communication **STIC** Matière : Imagerie et Vision Artificielle **ImVA**

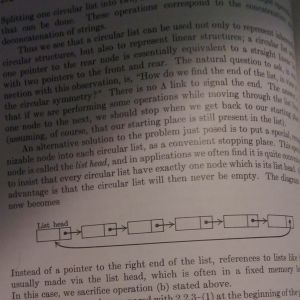
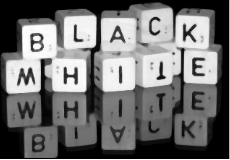
2023-2024

**TP N°5**

**Exercice 1 :**

1. Soient les deux images im1.jpg et im2.png , on veut appliquer la segmentation par seuillage. Choisissez aléatoirement 4 seuils et afficher le résultat correspondant

ret,th=cv2.threshold(gray,150,255,cv2.THRESH\_BINARY)



**Im1.jpg Im2.png**

1. On veut appliquer la segmentation par seuillage adaptatif sur les images im1.jpg et im2.png. La commande openCV utilisée est :

**cv.AdaptiveThreshold(src,dst,maxValue,adaptive\_method=CV\_ADAPTIVE\_THRESH\_MEAN\_C,thresholdType=CV\_THRESH\_BINARY,blockSize=3,param1=5)**

**Where :**

1. **src** - Source 8-bit single-channel image.
2. **dst** - Destination image of the same size and the same type as src.
3. **maxValue** - Non-zero value assigned to the pixels for which the condition is satisfied.
4. **adaptiveMethod** - Adaptive thresholding algorithm to use, ADAPTIVE\_THRESH\_MEAN\_C (threshold value is the mean of neighbourhood area) or ADAPTIVE\_THRESH\_GAUSSIAN\_C (threshold value is the weighted sum of neighbourhood values where weights are a gaussian window). **adaptiveMethod** decides how thresholding value is calculated.
5. **thresholdType** - Thresholding type that must be either THRESH\_BINARY or THRESH\_BINARY\_INV .
6. **blockSize** - size of a pixel neighborhood that is used to calculate a threshold value for the pixel: 3, 5, 7, and so on.
7. **C** - Constant subtracted from the mean or weighted mean.

**Solution : Code python :**

import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread('C:/Users/intel i7/Desktop/im2.png')

img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

img = cv2.medianBlur(img,5)

ret,th1 = cv2.threshold(img,127,255,cv2.THRESH\_BINARY)

th2 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE\_THRESH\_MEAN\_C,cv2.THRESH\_BINARY,11,2)

th3 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C,cv2.THRESH\_BINARY,11,2)

titles = ['Original Image', 'Global Thresholding (v = 127)',

'Adaptive Mean Thresholding', 'Adaptive Gaussian Thresholding']

images = [img, th1, th2, th3]

for i in range(4):

plt.subplot(2,2,i+1),plt.imshow(images[i],'gray')

plt.title(titles[i])

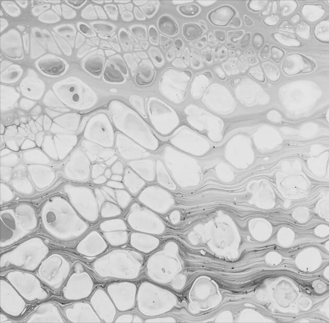
plt.xticks([]),plt.yticks([])

plt.show()

**Exercice 2 :**

On veut appliquer la segmentation par seuillage en utilisant la méthode d’otsu

Sur les deux images : im3.png et im4.jpg



**im3.png im4.jpg**

La commande openCV utilisée est :

cv2.threshold(grayImg, 0, 255, cv2.THRESH\_OTSU)

**Solution : Code python :**

import cv2

img = cv2.imread('C:/Users/intel i7/Desktop/im4.jpg')

grayImg = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

thresh, binaryImg = cv2.threshold(grayImg, 0, 255, cv2.THRESH\_OTSU)

print(thresh)

cv2.imshow('Grayscale image', grayImg)

cv2.imshow('Binary image', binaryImg)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Exercice 3 :**

Implémenter l’algorithme de seuillage par calcul de moyenne et appliquez-le sur les images im1.jpg, im2.png, im3.png et im4.jpg

**Exercice 4 :**

Implémenter la méthode de seuillage par classification (k-means) avec application sur les images im1.jpg, im2.png, im3.png et im4.jpg

**Exercice 5 :**

Implémenter la méthode d’**otsu** pour la segmentation par seuillage avec application sur les images im1.jpg, im2.png, im3.png et im4.jpg

Voir lien : <https://www.baeldung.com/cs/otsu-segmentation>