Année universitaire

2021 - 2022

4ETI - IMI — Traitement d'images

TP de ligne de partage des eaux

Marion Foare

Eric Van Reeth

Présentation du TP

Objectifs

Ce TP de 4 heures consiste à implémenter un algorithme de segmentation par ligne de partage des eaux (LPE), via la méthode des files d'attentes hiérarchiques (FAH).
La segmentation sera effectuée sur l'image contenant des smarties sur un fond clair (Figure 1). Il s'agira d'attribuer un label unique à chaque bonbon afin de pouvoir les séparer et les compter.

FIGURE 1 – Image à segmenter

Déroulement

Ce TP s'effectue en binôme, en Python et en utilisant de préférence l'IDE VSCode. L'image *smarties.png* à segmenter est fournie avec le TP, ainsi qu'un article décrivant le fonctionnement de la FAH dans le cadre de la LPE (déjà vu en cours).
Dans un premier temps, il s'agira de générer les marqueurs qui serviront d'initialisation à la segmentation. Enfin, l'algorithme de LPE par FAH sera implémenté en suivant le principe décrit dans les pages 19 à 24 de l'article.

Évaluation

Dans un délai précisé par l'encadrant, vous déposerez sur le dépôt e-campus ouvert un **compte-rendu au format pdf**, ainsi que le **code python** (fonctionnel) implémenté.
Le compte-rendu contiendra une description rigoureuse de la méthode implémentée, ainsi que les figures pertinentes permettant d'illustrer la démarche suivie. Les résultats obtenus seront clairement affichés, et une analyse critique des résultats est attendue (temps de convergence, pistes d'amélioration). Ne pas mettre de code dans le compte-rendu.
Le code sera commenté et séparé en parties clairement identifiées permettant au correcteur de comprendre et de reproduire tous vos résultats (les codes rendus seront comparés, et les éventuels doublons seront sanctionnées pour les deux groupes).

Marion Foare
Eric Van Reeth

Mise en place de l'environnement Python

Veillez à ce que l'**interpréteur Python** utilisé dans votre IDE soit le suivant :
`/opt/python/cpe/bin/python3`
Pour ce TP, vous aurez besoin des librairies suivantes :

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
```

Une aide des fonctions Python peut être obtenue via la commande `help(functionName)`.
Les commandes openCV nécessaires au TP sont fournies dans la dernière partie du sujet.

1 Pré-traitement

1.1 Obtention des marqueurs

La première étape de l'algorithme LPE consiste à placer des marqueurs à l'intérieur de chaque élément à segmenter.
Note : le fond de l'image doit également contenir un marqueur. Il aura donc son propre label.
L'objectif est d'obtenir une image similaire à la Figure 2, dans laquelle chaque couleur correspond à un label.

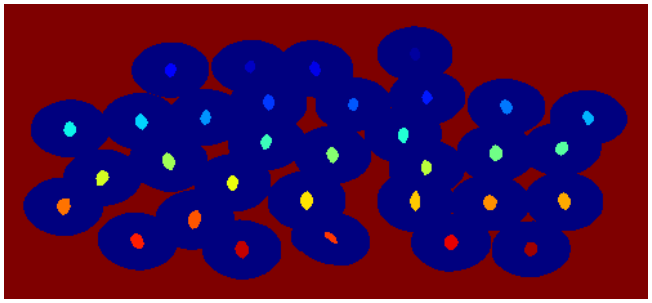

FIGURE 2 – Marqueurs initiaux utilisés pour la LPE

L'obtention de cette image nécessite une série d'opérations simples qu'il vous faudra implémenter :

1. Lecture de l'image (en niveau de gris)
2. Seuillage de l'image (smarties en blanc, fond en noir)
3. Grâce à des opérations morphologiques à définir, isoler un marqueur unique dans chaque bonbon
4. Attribution d'un label unique à chaque marqueur

Une image de marqueurs correcte est primordiale au bon fonctionnement de la LPE. Vous ferez donc attention aux points suivants :

- Avoir un et un seul marqueur par grain (pas de marqueurs "parasites" qui ne correspondent à aucun bonbon)
- Les marqueurs ne doivent pas être connectés
- Un marqueur du fond doit être présent dans toutes les régions entourées de bonbons

1.2 Obtention de la carte des distances

La carte des distances servira à définir le niveau de priorité associé à chaque pixel. On considérera ici que les **faibles intensités** de cette carte de distances correspondent aux niveaux de **priorités élevées**.
Le calcul de cette carte se fera en suivant les étapes suivantes :

1. Calcul de la carte des distances à partir de la segmentation smarties/fond obtenue dans la partie précédente
2. On veillera à ce que cette carte ait des valeurs entières (codée en `uint8`) répartie sur l'intervalle $[0, 255]$.
3. Enfin, on fera en sorte d'affecter un niveau de priorité élevé au centre de chaque bonbon. Le résultat doit être semblable à celui de la figure 3.

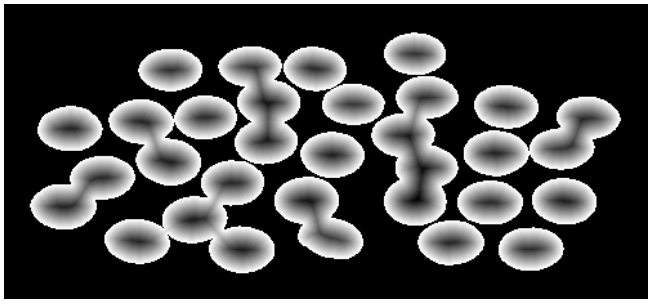

FIGURE 3 – Carte des distances à obtenir, permettant d'établir le niveau de priorité des pixels

2 Algorithme LPE

La file d'attente hiérarchique telle que décrite en cours et résumées dans l'article fourni sera initialisée à partir des marqueurs obtenus (Figure 2). Cette image correspond à la variable g de l'article. La variable f de l'article est l'image qui contient les priorités que l'on associera à chaque pixel, c'est à dire, la carte de distances (Figure 3). La démarche attendue est donc la suivante :

1. Initialiser la FAH. Il est conseillé de créer une FAH par coordonnée (FAH_x et FAH_y par exemple), et de les créer sous forme de listes afin de bénéficier des propriétés de celles-ci (*pop* et *append* notamment).

2. Implémenter l'algorithme itératif décrit en partie 3.2.2.2 de l'article sera implémenté jusqu'à vider la FAH.
3. Afficher l'évolution de la segmentation au fur et à mesure des itérations afin de vérifier que l'algorithme ait bien le comportement attendu. Vous afficherez également le résultat de la segmentation ainsi que le nombre de bonbons détectés.

Application sur un cas concret : L'image 4 contient des structures qui proviennent de phénomènes de réaction-diffusion, et apparaissent lors de la cristallisation spontanée de la pérovskite. La pérovskite est un minéral composé de titane et de calcium qui a des propriétés électriques et semi-conductrices prometteuses dans les domaines du photovoltaïque, de la photonique et de l'opto-électronique. La structure aléatoire en apparence, présente en réalité un ordre caché (e.g. aléatoire à longue distance mais avec des corrélations à courte distance). Cette organisation dite hyper-uniforme est connue depuis longtemps dans la nature (tache de léopard, rayures de zèbre), et Alan Turing est à l'origine de leur étude.

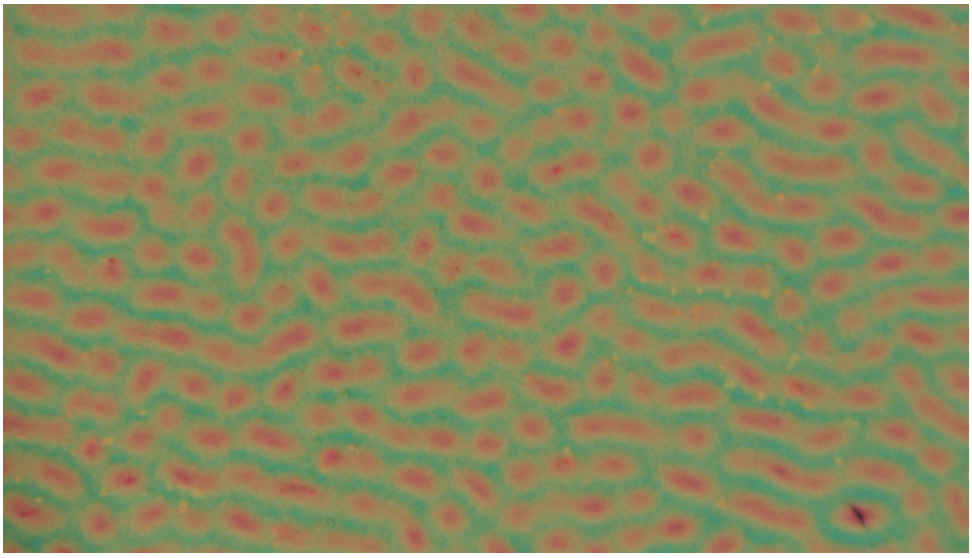

Figure 4 – Image microscope de pérovskite

Travail demandé : Afin de faciliter l'étude de l'organisation des structures, il est nécessaire de segmenter chaque structure de pérovskite (apparaissant en rouge sur l'image). Vous veillerez à ce que les structures adjacentes soient segmentées avec deux labels différents.
Les modifications effectuées sur la phase de pré-traitement seront expliquées et justifiées en détail, et les résultats obtenus clairement présentés et analysés.

3 Aide pour le code

3.1 Commandes utiles

```
cv2.imread('imageName.ext', 0) # ouverture d'une image et
 ↪  conversion en niveaux de gris
cv2.threshold() # pour le seuillage d'image
cv2.distanceTransform() # pour le calcul de la carte de
 ↪  distance
cv2.multiply() # multiplication élément par élément
cv2.erode() # pour effectuer une érosion
cv2.dilate() # pour effectuer une dilatation
cv2.getStructuringElement() # génèration d'éléments
 ↪  structurants
cv2.connectedComponents() # attribution d'un label unique à
 ↪  chaque élément de l'image dont l'intensité vaut 1
np.ones() # création d'un np.array rempli de 1
np.meshgrid() # retourne des matrices de coordonnées
np.size() # retourne un tuple contenant les dimensions d'un
 ↪  numpy array
np.uint8() # Conversion d'un numpy array au format uint8
list.pop() # permet d'extraire le dernier élément de la liste
list.append() # permet d'ajouter un élément à une liste
```

3.2 Commandes pour l'affichage

Affichage standard en niveaux de gris :

```
plt.figure() # ouvre une nouvelle figure
plt.imshow(I,'gray') # affichage de l'image I en niveau de
 ↪  gris
plt.show() # déclenche l'affichage
```

Sauvegarde d'une image sur le disque :

```
plt.imsave('im.png',I,cmap='gray') # ou autre colormap
```

Commandes pour l'affichage de 3 images en *subplot* avec chacune leur *colormap* spécifique :

```
plt.figure() # ouvre une nouvelle figure
plt.subplot(131) # Image 1
plt.imshow(img1, 'binary') # colormap 'binary'
plt.title('Thresholded Image')
plt.subplot(132) # Image 2
plt.imshow(img2, 'gray') # colormap 'gray'
plt.title('Distance Transform')
plt.subplot(133) # Image 3
plt.imshow(img3, 'jet') # colormap 'jet'
plt.title('Labels')
plt.show()
```