Année universitaire

2021 - 2022

4ETI — Traitement d'images

TP de morphologie mathématique

Marion Foare

Eric Van Reeth

Objectif. Ce TP de 4 heures consiste à mettre en application des outils de morphologie mathématique appliqués sur différentes images en niveaux de gris. Il s'agira d'avoir un aperçu de la variété des traitements pouvant être réalisés avec ces opérations.

Déroulement. Ce TP s'effectue en binôme par poste informatique, sous Ubuntu. Le langage utilisé sera Python3, car de nombreuses fonctions de morphologie mathématique y sont implémentées. Vous trouverez avec ce sujet l'ensemble des images à traiter, ainsi qu'un poly décrivant le principe de fonctionnement des opérations morphologiques en niveaux de gris.

Évaluation. Ce TP n'est pas noté, mais la rédaction d'un compte-rendu est encouragée. Les notions abordées sont évidemment au programme du DS.

Mise en place de l'environnement Python

Pour ce TP, vous aurez besoin des librairies suivantes :

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

L'utilisation de l'IDE VSCode est fortement recommandée pour développer les codes de ce TP. Veillez à ce que l'interpréteur Python utilisé soit le suivant :
`/opt/python/cpe/bin/python3`
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 Gradient morphologique

1. Calculer le gradient morphologique de l'image *Ampoule.png* avec un élément structurant isotrope
2. Proposer un traitement afin d'éliminer le reflet blanc de l'ampoule, tout en limitant l'impact sur les autres éléments de l'image
3. Calculer à nouveau le gradient morphologique de l'image sans le reflet
4. Analyser l'effet de la taille de l'élément structurant

2 Comptage d'éléments

L'objectif est de calculer automatiquement le nombre de cercles noirs contenus dans l'image *blobs2.png*. Pour cela, dans un premier temps, il s'agira de générer un marqueur unique par cercle. Puis, la fonction `cv2.connectedComponents()` sera utilisée pour compter le nombre d'éléments distincts dans l'image obtenue.

1. Calculer la carte des distances de l'image originale, c'est-à-dire pour chaque pixel appartenant à un élément, la distance au contour le plus proche.
2. Détecter les maxima locaux de cette carte de distance afin d'isoler un marqueur par cercle
3. En déduire le nombre de cercles.

3 Détection de structures d'intérêts

On travaillera sur l'image *angiogram.png*.

1. Quelles opérations appliquer pour faire ressortir les structures vasculaires de l'image originale.
2. Comparer et commenter l'efficacité des différentes opérations implémentées.

4 Reconstructions morphologiques

Cette partie sera consacrée à l'implémentation de reconstructions morphologiques. Le poly de cours rappelle le principe ces méthodes, qui s'apparente fortement aux reconstructions géodésiques par marqueurs vues pour des images binaires en TSI.

4.1 Cas 1

Pour cette partie, nous travaillerons sur l'image *calculator.png*. L'objectif de cet exercice est de supprimer les caractères ne contenant pas de longues lignes verticales (à vous de décider de la longueur de ces lignes).

1. Obtenir l'image de marqueur M permettant la suppression des caractères souhaités
2. Effectuer la reconstruction morphologique pertinente à partir de cette image de marqueur jusqu'à convergence, en prenant pour masque l'image originale.
3. Afficher et discuter le résultat. Faire varier la dimension de l'élément structurant pour étudier son impact.

4.2 Cas 2

Dans l'exemple suivant, on cherche à éliminer les points noirs sur les dés blancs de l'image *des.jpg*

1. Obtenir l'image de marqueur M permettant la suppression des éléments souhaités
2. Effectuer la reconstruction morphologique pertinente à partir de cette image de marqueur jusqu'à convergence, en prenant pour masque l'image originale.

5 Aide pour le code

5.1 Commandes utiles

```
cv2.imread('imageName.ext', 0) # ouverture d'une image et
 ↪  conversion en niveaux de gris
S = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(7,7)) #
 ↪  création d'un élément structurant circulaire de rayon 7
 ↪  pixels
cv2.erode(I, S, iterations = n) # n érosions successives de I
 ↪  par S
cv2.dilate(I, S, iterations = n) # n dilatations successives
 ↪  de I par S
cv2.morphologyEx(I, cv2.MORPH_OPEN, S) # ouverture de I par S
```

```
cv2.morphologyEx(I, cv2.MORPH_CLOSE, S) # fermeture de I par S
cv2.morphologyEx(I, cv2.MORPH_TOPHAT, S) # top-hat de I par S
cv2.morphologyEx(I, cv2.MORPH_BLACKHAT, S) # bottom-hat de I
 ↪ par S
cv2.distanceTransform() # pour le calcul de la carte de
 ↪ distance
np.minimum(A,B) # retourne le minimum élément par élément des
 ↪ arrays A et B
```

5.2 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
```

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()
```

Pour afficher une image dont le contenu évolue à l'intérieur d'une boucle, penser à autoriser le mode interactif de `matplotlib` via la commande : `plt.ion()`