CPE LYON

Année universitaire

2020 - 2021

4ETI — Traitement d'images

TP de contours actifs (Snakes)

Marion Foare

Eric Van Reeth

Objectif. Ce TP de 4 heures consiste à implémenter une segmentation par l'algorithme de *snakes* vu en cours sur des images simples. Á partir d'un contour initial, la segmentation sera actualisée itérativement à partir du schéma itératif vu en cours.

Déroulement. Ce TP s'effectue en binôme, en Python et en utilisant de préférence l'IDE VSCode. Vous trouverez avec le TP un dossier contenant les images sur lesquelles vous travaillerez, à savoir :

- l'image *im_goutte.png*, que tous les groupes devront segmenter dans un premier temps
- les autres images (numérotées), qui seront attribuées de façon individuelle à chaque groupe

Évaluation. Dans un délai précisé par l'encadrant, vous déposerez sur le dépôt ouvert un **compte-rendu au format pdf**, ainsi que le **code python** (fonctionnel) implémenté. Le tout sera compressé dans une archive (.zip) contenant les noms des membres du binômes.
Le compte-rendu contiendra une description rigoureuse des méthodes implémentées, une discussion sur le(s) critère(s) d'arrêt choisi(s) et les résultats obtenus.
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).

1 Mise en place de l'environnement Python

Pour ce TP, vous aurez besoin des librairies suivantes :

```
import numpy as np
from matplotlib import pyplot as plt
from scipy import sparse as sp
import cv2
```

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.

2 Algorithme à implémenter

2.1 Trame du code

Voici la trame du code à implémenter :

1. Lecture de l'image à segmenter (en niveaux de gris)
2. Calculer de la norme du gradient de l'image à segmenter
3. Définition des paramètres de l'algorithme (α, β, γ, et K : nombre de points qui composent le *snake*)
4. Initialisation du *snake* comme un cercle englobant l'objet à segmenter. Le *snake* sera défini à partir de ses coordonnées en x et en y que vous pourrez stocker dans des variables
5. Calcul de D_2, de D_4 et de A sous forme de matrices parcimonieuses
6. Lancement de la procédure itérative pour faire évoluer les coordonnées du *snake* jusqu'à convergence (voir formule du cours)
7. Affichage de la convergence du *snake* au fur et à mesure des itérations. Pour plus de fluidité, vous pouvez n'affichez l'évolution du *snake* que, par exemple, toutes les 10 itérations
8. Affichage des informations relatives à la convergence (nombre d'itérations, critère d'arrêt, temps de calcul, évolution des différentes énergies, ...)

2.2 Question subsidiaire

Tester votre algorithme sur une image bruitée.

3 Aide pour le code

3.1 Commandes utiles

```
cv2.imread('imageName.ext', 0) # ouverture d'une image et
 ↪  conversion en niveaux de gris
np.meshgrid() # retourne des matrices de coordonnées
np.square() # élévation au carré de chaque élément d'un array
cv2.Sobel() # applique un filtre de Sobel à une image
cv2.GaussianBlur() # applique un flou gaussien sur une image
sp.diags() # pour la création de matrices parcimonieuses dont
 ↪  on définit les termes diagonaux
np.linalg.inv() # pour calculer l'inverse d'une matrice
```

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
```

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()`