Dorian

GROUTEAU

21904277

M2 VMI

18/10/2024

COMPTE RENDU

TP 1 Reconnaissance de formes :

Algorithme Fuzzy C Means

1. Objectif du TP

L'objectif de ce TP est d'implémenter l'algorithme de Fuzzy C-means et de l'appliquer à la segmentation d'une image en régions floues. Nous utiliserons une carte de chaleur (heatmap) pour visualiser le résultat de la segmentation. Le projet se concentrera dans un premier temps sur une image en niveaux de gris, avec deux clusters (K=2), puis sera étendu à une image en couleurs (RGB).

2. Ce que j'ai fait

Pour ce TP, j'ai décidé d'utiliser le langage de programmation Python pour sa simplicité et rapidité d'utilisation.

J'ai également utilisé 3 bibliothèques: numpy, matplotlib et cv2.

J'ai donc décidé d'utiliser des tableaux numpy pour accroître grandement la vitesse des calculs (même si cela peut parfois complexifier un peu le code).

Implémentation de l'algorithme Fuzzy C-means :

- J'ai dans un premier temps défini la fonction Initinitialize_membership_matrix, nous obtenons une matrice de forme (n_clusters, nombre_de_pixels).
- J'ai ensuite défini directement la fonction fuzzy_c_means (sans avoir les fonctions Update, pour deja avoir la structure du code) :

  Paramètres de la fonction :

  - img : l'image d'origine en niveau de gris.
  - n_clusters : le nombre de clusters.
  - m : le paramètre de fuzziness.
  - max_iter : le nombre maximal d'itérations de l'algorithme.
  - tol : la tolérance pour le critère d'arrêt.

  Étapes de l'algorithme :

  - Aplatir l'image
  - Initialisation de la matrice d'appartenance
  - Calcul des nouveaux centres de clusters = update_centroids.
  - Mise à jour de la matrice d'appartenance = update_membership_matrix.
  - Critère d'arrêt : L'algorithme vérifie si la différence entre la nouvelle et l'ancienne matrice d'appartenance est inférieure à une tolérance donnée.
  - Résultats retournés : L'algorithme retourne les centres finaux des clusters et la matrice d'appartenance finale, qui indique dans quelle mesure chaque pixel appartient à chaque cluster.

- Il me manquait alors simplement les deux fonctions de MAJ:
    - update_centroids : j’ai repris la formule donnée en utilisant numpy (dot pour le numérateur et sum pour le dénominateur).
    - update_membership_matrix : j’ai également repris la formule proposé, mais cette fois ci j'ai dû y réfléchir plus longuement. Je calcul dans un premier temps les distances puis j’obtiens les puissances (distances misent à la puissance (2.0 / (m - 1))). Il me suffit ensuite de normaliser les données pour obtenir la matrice d’appartenance.

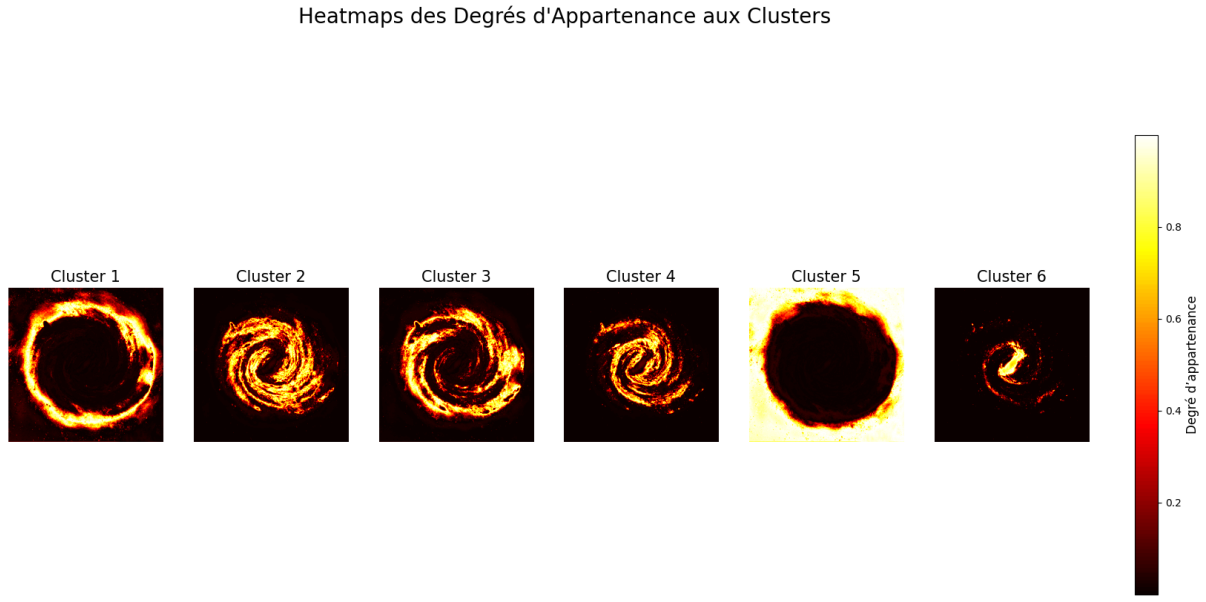
3. Affichage des cartes de chaleurs

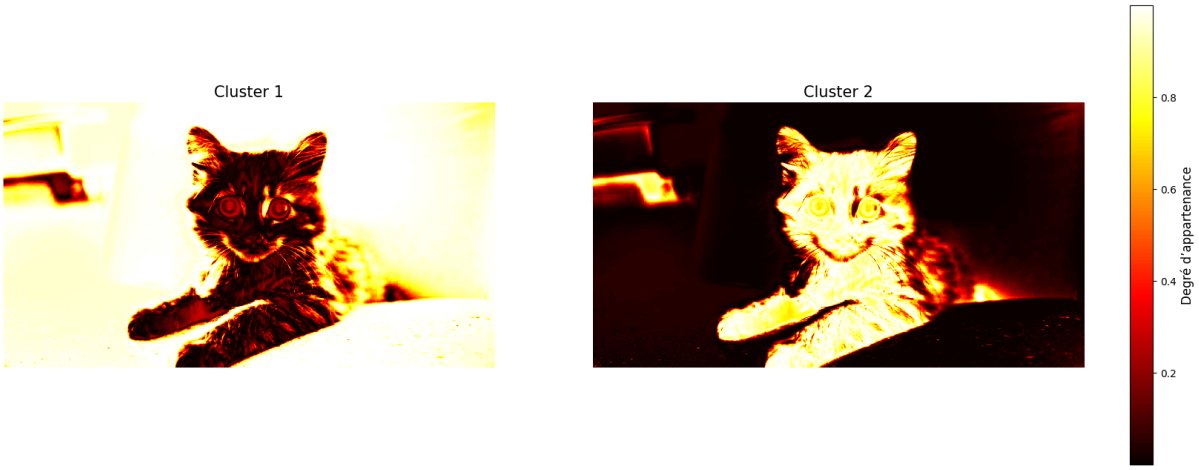
Enfin, pour l’affichage nous avons utilisé des cartes de chaleurs (plot avec matplolib et une colormap).

4. Passage en RGB

Le passage de niveau gris à RGB a été relativement simple, nous considérons maintenant que chaque pixel a 3 canaux. L’image aplatie a donc 3 canaux (nous aplatissons sur chaque canal). Lorsque nous calculons les distances nous utilisons la distance euclidienne.

5. Résultats





J’ai utilisé deux images pour les tests : l’image fournie milky_waves.jpg et une image d’un chat.

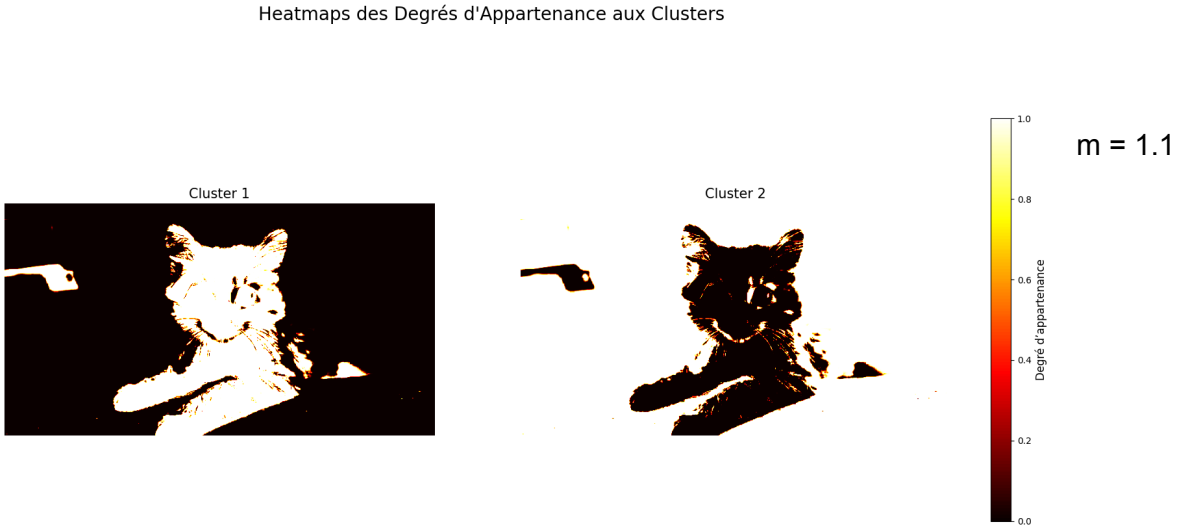
Sur la première image, la segmentation est plutôt bien réussie. Nous pouvons classer les clusters du plus proche au centre au plus éloigné : 6, 4, 2, 3, 1, 5.

Sur la deuxième image, la partie avant du chat est bien segmentée, mais la partie arrière l’est moins. Il serait utile d'ajuster les hyperparamètres, notamment le paramètre de fuzziness et le nombre de clusters, pour améliorer ce résultat.

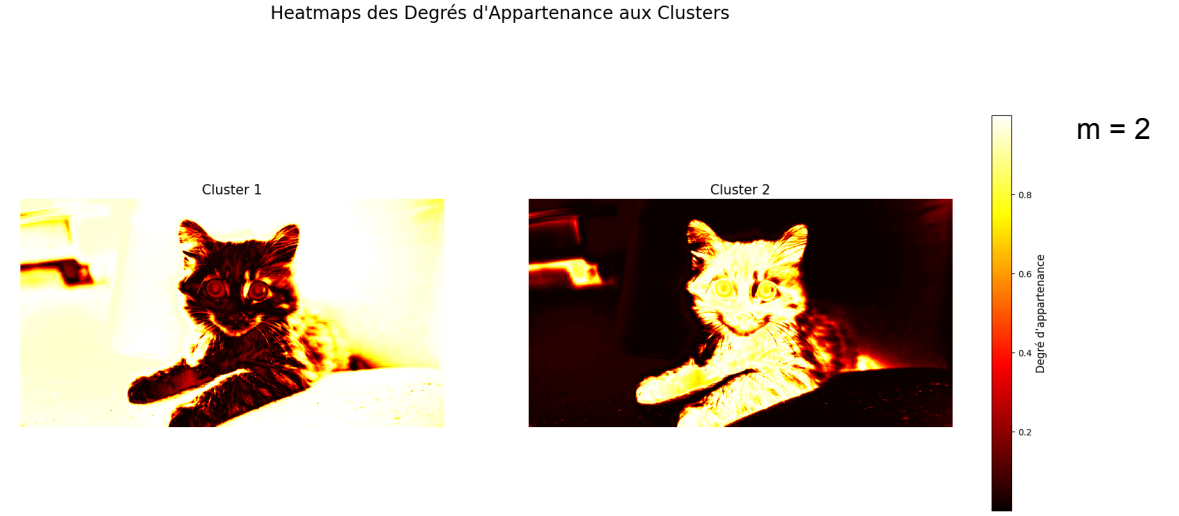
4. Conclusion

L'implémentation de l'algorithme Fuzzy C-means avec NumPy a permis d'accélérer significativement le processus de segmentation, rendant l'algorithme efficace même sur des images de taille modérée. Les résultats obtenus sont globalement satisfaisants, notamment sur l'image milky_waves.jpg où la segmentation est bien réalisée. Cependant, sur certaines images, comme celle du chat, des ajustements des hyperparamètres, tels que le paramètre de fuzziness et le nombre de clusters, seraient nécessaires pour améliorer la qualité de la segmentation, notamment pour des images plus complexes. Ces ajustements pourraient également permettre une meilleure granularité des résultats.

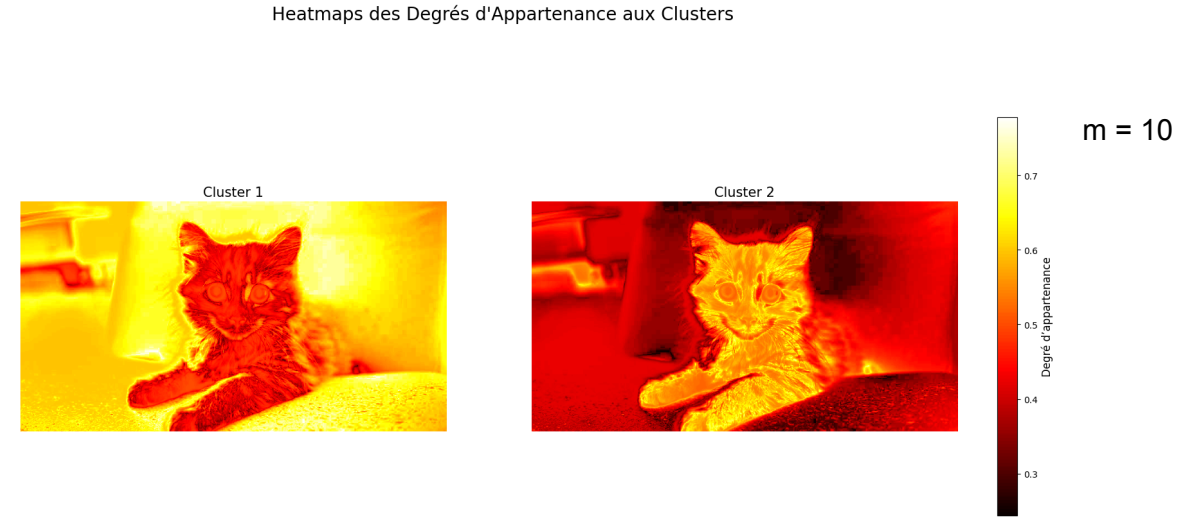
5. Annexes (autres tests)



m = 1.1



m = 2



m = 10

Lorsque le paramètre de fuzziness mmm est proche de 1, la segmentation devient plus stricte, et chaque pixel est presque totalement assigné à un seul cluster, avec des frontières nettes. À mesure que mmm augmente, les pixels ont une appartenance plus répartie entre plusieurs clusters, créant un flou plus important dans la segmentation. Les transitions entre les régions deviennent donc plus progressives et moins marquées.