M2 - VMI
UFR de Mathématiques et Informatique
Université Paris Cité

Rapport TP N°01
Implémentation de Fuzzy C-Mean

Réalisé par :
ABED Nada Fatima Zohra

Le lien git du TP

Date : 25 octobre 2024

1 Introduction

Dans ce rapport, Je décris l'implémentation de l'algorithme Fuzzy C-Means (FCM) pour la segmentation d'images. Cet algorithme permet de regrouper des données en clusters basés sur leur similarité, tout en autorisant une appartenance partielle à plusieurs clusters.

FCM est particulièrement utile dans le traitement d'images où les pixels peuvent ne pas appartenir strictement à un seul objet ou à un arrière-plan.

2 Description de l'algorithme

L'algorithme Fuzzy C-Means est un processus itératif qui :
- Calcule la distance entre chaque point de données (par exemple, les pixels d'une image) et les centres de cluster actuels.
- Met à jour les centres des clusters et les degrés d'appartenance pour chaque point de données.

La matrice d'appartenance est initialisée de manière aléatoire. L'algorithme ajuste ensuite cette matrice de manière itérative jusqu'à ce que la convergence soit atteinte, c'est-à-dire lorsque le changement dans la matrice d'appartenance est inférieur à un certain seuil ou un certain nombre d'iterations est atteint.

2.1 Fonctions principales

- `compute_distances` : Cette fonction calcule la distance euclidienne entre chaque pixel et chaque centre de cluster.
- `initialize_membership_matrix` : Cette fonction initialise de manière aléatoire le degré d'appartenance de chaque pixel à chaque cluster.
- `update_cluster_centers` : Cette fonction met à jour les centres des clusters en fonction des valeurs actuelles des degrés d'appartenance.
- `update_membership_matrix` : Cette fonction met à jour la matrice d'appartenance en fonction des nouveaux centres des clusters et des distances des pixels.
- `fuzzy_c_means` : La fonction principale qui applique le clustering flou en appelant les fonctions ci-dessus de manière itérative jusqu'à la convergence (epsilon ou nombre d'itération maximale).

3 Implémentation

L'implémentation de l'algorithme Fuzzy C-Means se fait selon les étapes suivantes :

1. Convertir l'image en un tableau à une dimension de pixels.
2. Initialiser la matrice d'appartenance de manière aléatoire.
3. Répéter les étapes suivantes jusqu'à la convergence :
   - Mettre à jour les centres des clusters en utilisant les valeurs floues des degrés d'appartenance.
   - Calculer la distance de chaque pixel par rapport aux centres des clusters.
   - Mettre à jour la matrice d'appartenance en fonction des nouvelles distances.
4. Reshaper le résultat pour retrouver une image segmentée visualisable.

```python
# Fuzzy C-Means Clustering
def fuzzy_c_means(image, n_clusters=3, fuzziness=2, max_iter=100, error=1e-5):
    # Convertir l'image en un tableau de pixels
    image_pixels = image.reshape((-1, 3)).astype(np.float64)
    n_pixels = len(image_pixels)
    # Initialisation de la matrice de degrés d'appartenance
    membership_matrix = initialize_membership_matrix(n_pixels, n_clusters)

    for iteration in range(max_iter):
        # Mise à jour des centres des clusters
        cluster_centers = update_cluster_centers(image_pixels, membership_matrix, n_clusters, fuzziness)

        # Calcul des distances
        distances = compute_distances(image_pixels, cluster_centers)
        old_membership_matrix = membership_matrix.copy()

        membership_matrix = update_membership_matrix(distances, n_clusters, fuzziness)

        # Vérification de la convergence
        if np.linalg.norm(membership_matrix - old_membership_matrix) < error:
            print(f"Convergence atteinte après {iteration+1} itérations.")
            break
    segmented_image = np.argmax(membership_matrix, axis=1).reshape(image.shape[:2])

    return segmented_image, cluster_centers
```

FIGURE 1 – Code qui implémente l'algorithme Fuzzy C-mean

Pour l'affichage de chaque cluster, on crée une nouvelle image où seuls les pixels appartenant à ce cluster sont visibles, tandis que les autres sont masqués (noirs). Chaque cluster est ensuite affiché dans une sous-figure distincte, ce qui permet de visualiser clairement la répartition des pixels entre les différents clusters.

```python
import cv2
# Fonction d'affichage
def display_image_per_cluster(image, segmented_image, cluster_centers):
    n_clusters = len(cluster_centers)
    plt.figure(figsize=(15, 5))

    for cluster_idx in range(n_clusters):
        # Créer image où seuls les pixels du cluster actuel sont affichés
        cluster_image = np.zeros_like(image)
        # Attribuer aux pixels du cluster actuel leur couleur d'origine
        cluster_image[segmented_image == cluster_idx] = image[segmented_image == cluster_idx]
        # Affichage du cluster
        plt.subplot(1, n_clusters, cluster_idx + 1)
        plt.title(f"Cluster {cluster_idx + 1}")
        plt.imshow(cv2.cvtColor(cluster_image.astype(np.uint8), cv2.COLOR_BGR2RGB))
        plt.axis('off')

    plt.tight_layout()
    plt.show()
```

FIGURE 2 – Code qui implémente l'algorithme d'affichage des clusters

4 Résultats

Après avoir exécuté l'algorithme Fuzzy C-Means sur une l'image milky way, on obtient les résultats suivants :

FIGURE 3 – Image Milky way RGB - nombre de clusters : 03

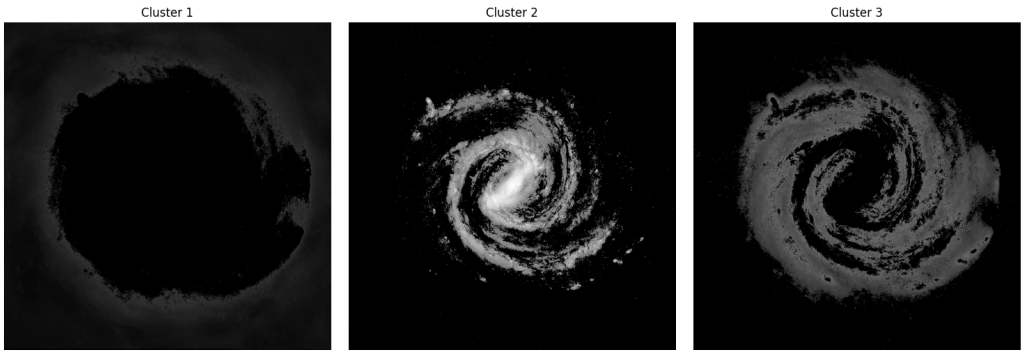

FIGURE 4 – Image Milky way niveaux de gris - nombre de clusters : 03

5 Conclusion

L'algorithme de clustering Fuzzy C-means fournit une méthode flexible pour la segmentation d'images, en particulier lorsque les limites entre les objets ne sont pas bien définies. Dans cette implémentation, on a démontré son utilité dans le domaine de segmentation d'image en regroupant les pixels similaires en clusters selon leurs valeurs de couleur.