Etude de méthodes de Clustering pour la segmentation d'images en couleurs

D'Hondt Frédéric, El Khayati Brahim
Faculté Polytechnique de Mons, 5ème Electricité, Certificat Applicatifs Multimédia

***Résumé*— Ce projet consiste en l'évaluation des performances de deux algorithmes de segmentation d'images en couleur : *Fuzzy C-means* et *K-means*. Dans un premier temps, nous présentons le fonctionnement des deux méthodes, ainsi que leurs avantages et inconvénients. Dans un second temps, nous utilisons les algorithmes implémentés sous Matlab afin d'évaluer les performances d'une part, de manière subjective et d'autre part, de manière objective, par l'utilisation d'un logiciel de reconnaissance optique de caractères.**

***Mots clefs*—Clustering Methods, Clusters, Fuzzy C-Means, K-Means.**

I. INTRODUCTION

Le *clustering*, ou segmentation, est une étape de base du traitement d'une image. Cette opération a pour but de séparer différentes zones homogènes d'une image, afin d'organiser les objets en groupes (*clusters*) dont les membres ont en commun diverses propriétés (intensité, couleur, texture, etc). On peut regrouper les méthodes de segmentation en deux catégories : la segmentation non supervisée, qui vise à séparer automatiquement l'image en clusters naturels, c'est-à-dire sans aucune connaissance préalable des classes ; et la segmentation supervisée, qui s'opère à partir de la connaissance de chacune des classes définies par une approche probabiliste.

Dans le cadre de ce projet, nous nous limiterons à l'étude des deux méthodes de segmentation non-supervisée suivantes : le *Fuzzy C-means* et le *k-means*.

Nous présenterons tout d'abord une brève description du fonctionnement de ces algorithmes, ainsi qu'une comparaison de leurs performances. Nous présenterons ensuite les toolbox Matlab disponibles. Nous mettrons finalement en application les deux algorithmes par une série d'exemples testés à l'aide des fonctions implémentées sous Matlab.

II. DESCRIPTION DES ALGORITHMES

Fuzzy C-Means (FCM) est un algorithme de classification non-supervisée floue. Issu de l'algorithme des C-moyennes (*C-means*), il introduit la notion d'ensemble flou dans la définition des classes : chaque point dans l'ensemble des données appartient à chaque cluster avec un certain degré, et tous les clusters sont caractérisés par leur centre de gravité.

Comme les autres algorithmes de classification non supervisée, il utilise un critère de minimisation des distances intra-classe et de maximisation des distances inter-classe, mais en donnant un certain degré d'appartenance à chaque classe pour chaque pixel. Cet algorithme nécessite la connaissance préalable du nombre de clusters et génère les classes par un processus itératif en minimisant une fonction objective. Ainsi, il permet d'obtenir une partition floue de l'image en donnant à chaque pixel un degré d'appartenance (compris entre 0 et 1) à une classe donnée. Le cluster auquel est associé un pixel est celui dont le degré d'appartenance sera le plus élevé.

Les principales étapes de l'algorithme Fuzzy C-means sont :

1. La fixation arbitraire d'une matrice d'appartenance.
2. Le calcul des centroïdes des classes.
3. Le réajustement de la matrice d'appartenance suivant la position des centroïdes.
4. Calcul du critère de minimisation et retour à l'étape 2 s'il y a non convergence de critère.

L'algorithme *k-means* est l'algorithme de clustering le plus connu et le plus utilisé, du fait de sa simplicité de mise en œuvre. Il partitionne les données d'une image en K clusters. Contrairement à d'autres méthodes dites hiérarchiques, qui créent une structure en « arbre de clusters » pour décrire les groupements, *k-means* ne crée qu'un seul niveau de clusters. L'algorithme renvoie une partition des données, dans laquelle les objets à l'intérieur de chaque cluster sont aussi proches que possible les uns des autres et aussi loin que possible des objets des autres clusters. Chaque cluster de la partition est défini par ses objets et son centroïde.

Le *k-means* est un algorithme itératif qui minimise la somme des distances entre chaque objet et le centroïde de son cluster. La position initiale des centroïdes conditionne le résultat final, de sorte que les centroïdes doivent être initialement placés le plus loin possible les uns des autres de façon à optimiser l'algorithme. *K-means* change les objets de cluster jusqu'à ce que la somme ne puisse plus diminuer. Le résultat est un ensemble de clusters compacts et clairement séparés, sous réserve qu'on ait choisi la bonne valeur K du nombre de clusters.

Les principales étapes de l'algorithme *k-means* sont :

1. Choix aléatoire de la position initiale des K clusters.
2. (Ré-)Affecter les objets à un cluster suivant un critère de minimisation des distances (généralement selon une mesure de distance euclidienne).

3. Une fois tous les objets placés, recalculer les K centroïdes.
4. Réitérer les étapes 2 et 3 jusqu'à ce que plus aucune ré-affectation ne soit faite.

III. PRESENTATION DES TOOLBOX

Il est possible de télécharger gratuitement des toolbox Matlab implémentant les fonctions que nous recherchons. Parmi les toolbox que nous avons trouvées sur le net, figurent essentiellement les suivantes :

- *Fuzzy Clustering and Data Analysis Toolbox*, de Balazs Balasko, Janos Abonyi et Blazs Feil. Cette toolbox contient une fonction implémentant l'algorithme *Fuzzy C-Means*, que nous ne sommes toutefois pas parvenus à utiliser car elle nécessite l'utilisation de Visual Studio.
- *Fuzzy Logic Toolbox* implémentée sous Matlab 7. Cette toolbox contient la fonction *fcm.m*, implémentant le FCM.
- *Statistics Toolbox* implémentée sous Matlab 7. Cette toolbox contient la fonction *kmeans.m*, facile à prendre en main et bien documentée. Nous utiliserons cette fonction pour effectuer les tests sur l'algorithme *k-means*.
- *My Clustering Toolbox* de David Corney. Cette toolbox contient entre autre les fonctions *dcFuzzy.m*, *dcKmeans.m* qui implémentent les algorithmes que nous étudions. La fonction qui implémente FCM n'est pas évidente à prendre en main, c'est pourquoi notre choix s'est porté sur la fonction décrite ci-dessous.
- Nous avons trouvé un code exécutable sous Matlab (disponible sur le site de l'Ecole Nationale Supérieure des Télécommunications de Paris, dont le lien est mentionné dans les références), qui implémente l'algorithme FCM. La fonction Matlab fournie, *f_seg.m*, étant facile à utiliser, nous nous en sommes servi pour effectuer les tests.

IV. TESTS SUR MATLAB

Les tests sont effectués sous Matlab, par l'utilisation des fonctions *f_seg.m* et *kmeans.m*. Nous comparerons les deux algorithmes d'une part, de manière subjective, en commentant les images résultantes et d'autre part, de manière objective, par l'utilisation d'un logiciel de reconnaissance optique de caractères (OCR). Les tests ont été effectués avec un nombre K de clusters valant systématiquement 2, 3, 5 et 10. Nous avons pu observer que le meilleur résultat, tant d'un point de vue détection de caractères que d'un point de vue temps de calcul, correspond généralement à la détection de 2 clusters. Il apparaît tout à fait logique que cette valeur de K soit la plus adéquate, puisque l'on constatera dans toutes les images testées, qu'il y a essentiellement deux objets : le fond et les caractères (chiffres et mots). De ce fait, nous exposerons essentiellement les résultats pour K=2, les autres ne seront que brièvement décrits.

La première étape pour l'utilisation des fonctions est de convertir les images mises à notre disposition selon le code RGB, par la fonction *Imread()*.

Pour la fonction kmeans.m, il est nécessaire d'effectuer un pré-traitement qui consiste à convertir la matrice de pixels en un vecteur constitué des niveaux de gris pour chaque couleur de base (rouge, vert et bleu). Lorsque l'algorithme a été appliqué au vecteur, on peut ensuite le reconvertir en une matrice constituée de la valeur associée à chaque cluster, pour chaque pixel de l'image.
En ce qui concerne *f_seg.m*, il suffit de coder l'image au format RGB, par la fonction *Imread()* de Matlab. La fonction renvoie une matrice de pixels constituée de la valeur moyenne de la couleur (codée en RGB) associée au cluster qui leur correspond.

La première image est constituée de 130x42 pixels. Les caractères qui forment le mot « study » contiennent essentiellement du vert. Le fond de l'image est constitué de plusieurs nuances de bleu. Les tests sur cette image donnent les résultats suivants :

Figure 1 Image originale (au dessus), application de Fuzzy C-means (gauche) et k-means (droite) avec K=2.

Les deux algorithmes donnent des résultats très similaires (Figure 1). On remarque cependant l'apparition d'une tâche sous le « d » de « study » pour le FCM. La reconnaissance à l'aide de l'OCR du nom de Fine Reader 8 (disponible gratuitement en version d'essai) marche correctement pour FCM, alors que le résultat est « studv » pour k-means. Lorsque l'on augmente le nombre de clusters, on constate que les contours des caractères forment de nouveaux clusters et le fond de l'image est progressivement divisé en plusieurs parties. La figure 2 montre le résultat obtenu par l'application de k-means avec 2, 3, 5 et 10 clusters sur l'image précédente :

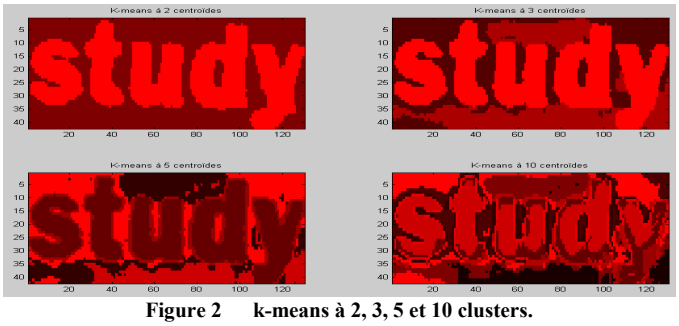

Figure 2 k-means à 2, 3, 5 et 10 clusters.

La deuxième image que nous avons testée comporte 1355x370 pixels. On distingue le mot « pepsi » en caractères blanc, sur un fond constitué de diverses nuances de bleu, ainsi que du rouge, du noir et du blanc. On peut a priori se douter qu'une

détection du mot sera difficile, compte tenu de la confusion entre les parties blanches du fond et les caractères. L'application des deux algorithmes nous donne le résultat suivant :

Figure 3 Image originale (au dessus), application de Fuzzy C-means (gauche) et k-means (droite) avec K=2.

FCM donne une segmentation peu performante (Figure 3), étant donné la similitude entre certaines parties du fond de l'image et des caractères. K-means donne un résultat un peu plus intéressant, mais tous deux ne permettent cependant pas d'identifier le mot « pepsi » à l'aide de l'OCR utilisé.

La troisième image testée représente une affiche de publicité pour « Apple Bank », de résolution 1280x960.

Figure 4 Image originale (au dessus), application de Fuzzy C-means (gauche) et k-means (droite) avec K=2.

Les deux algorithmes donnent un très bon résultat. La détection par OCR permet d'identifier tous les caractères. Etant donné la résolution de l'image, les tests effectués pour k-means avec 5 et 10 clusters nécessitent un temps de calcul très élevé. Nous avons remarqué que FCM est un peu plus rapide et ne donne un temps de calcul trop important qu'à des valeurs de K supérieures à environ 15. Le problème du temps de calcul et de la grande quantité de mémoire nécessaire pour les deux algorithmes s'explique par le fait qu'il s'agit de méthodes itératives. Ceci ne porte cependant pas à conséquence dans notre application, puisque le nombre d'objets présents sur les images testées n'excède jamais 10.

La quatrième image (352x150) est constituée d'un fond rose sur lequel on distingue le mot « race » en bleu. On remarque l'image est un peu floue, mais ceci ne pose aucun problème lors de la segmentation.

Figure 5 Image originale (au dessus), application de Fuzzy C-means (gauche) et k-means (droite) avec K=2.

Le fond de l'image étant très uniforme, la segmentation par FCM et k-means donnent un excellent résultat (Figure 5). La détection par OCR nous donne bien le mot « RACE ».

La cinquième (75x55 pixels) est constituée des chiffres « 22 ». On voit que les contours des chiffres sont très flous et la partie supérieure de l'image est légèrement foncée.

Figure 6 Image originale (au dessus), application de Fuzzy C-means (gauche) et k-means (droite) avec K=2.

Figure 7 Fuzzy C-means (gauche) et k-means (droite) avec K=3.

La partie sombre de l'image rend l'utilisation des deux algorithmes difficile ; d'une part, avec k=2 (Figure 6), on ne distingue pas clairement la partie supérieure de « 22 » et d'autre part, quand on augmente le nombre de clusters (exemple avec k=3 en Figure 7), on résout ce problème mais on perd alors la détection de la partie inférieure du « 22 », de sorte que le résultat de la détection par OCR donne « ?? ». L'augmentation du contraste, par la fonction Matlab *imadjust()*, ne permet pas la résolution de ce problème. Cet exemple illustre le fait que les objets à segmenter doivent avoir une couleur suffisamment uniforme pour que la détection soit envisageable.

La sixième image testée (426x94 pixels) contient le mot « redback ». Intuitivement, on peut déjà imaginer que le reflet sur la partie « red » va avoir un très mauvais effet sur la détection. L'application de FCM et k-means donne le résultat suivant (Figure 8) :

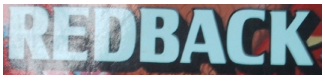

Figure 8 Image originale (au dessus), application de Fuzzy C-means (gauche) et k-means (droite) avec K=2.

On voit une sorte de reflet sur la partie « red » de l'image qui comme explicité précédemment, l'efface littéralement de l'image résultant de la segmentation. Dès lors, il n'est possible d'identifier que les caractères « back » sur cette image. L'augmentation du nombre de clusters améliore l'identification visuelle des caractères, mais n'apporte néanmoins aucune amélioration quant à la détection par OCR.
Le problème du reflet peut être résolu par action sur la luminance sur l'image. Aucune fonction Matlab ne permet de modifier la luminance de façon à faire disparaître le reflet, mais nous pouvons toutefois agir sur le contraste de l'image par la fonction *imadjust()*. En augmentant le contraste (voir Figure 9), on constate que les caractères sont de plus en plus distincts, de sorte que la segmentation donne un bien meilleur résultat. Dès lors, la détection par OCR du mot complet « redback » devient possible.

Figure 9 Application de Fuzzy C-means (gauche) et k-means (droite) avec K=2, après augmentation du contraste.

La septième et dernière image testée (de résolution 315x67) est composée d'un fond noir et du mot « DANCE », de couleur orange. La segmentation donne les résultats suivants (Figure 9) :

Figure 10 Image originale (au dessus), application de Fuzzy C-means (gauche) et k-means (droite) avec K=2.

Cette image est l'une des plus faciles à segmenter. En effet, il suffit de créer un cluster pour le fond noir et un cluster pour les caractères oranges. Le nombre de clusters le plus logique dans ce cas-ci est donc K=2. Ceci se confirme par les résultats obtenus après segmentation ; on distingue clairement le mot « DANCE » et on peut dès lors détecter par OCR le mot précité.

V. Observations generales

De manière générale, les deux algorithmes étudiés donnent un bon résultat mais présentent cependant deux inconvénients : d'une part, ils nécessitent le choix préalable du nombre K de clusters, ce qui rend impossible l'automatisation de la méthode ; d'autre part, ils requièrent un temps de calcul souvent élevé, du fait de leur nature itérative.
Nous avons pu constater que ces deux algorithmes s'avèrent efficaces lorsque les objets de l'image (essentiellement le fond et les caractères) sont clairement séparés.
La cinquième image nous indique que lorsque les contours ne sont pas bien définis, la segmentation devient difficile.
La sixième image testée montre clairement qu'une variation trop importante de la luminance (reflet sur une partie de l'image,…) rend impossible l'identification complète des caractères après segmentation. Dans ce cas-ci, le problème a pu être résolu en augmentant significativement le contraste de l'image.
Nous avons mesuré le temps d'exécution des deux algorithmes pour chaque image et il en ressort que FCM est plus rapide pour les images de grande résolution et que *k-means* est plus approprié dans le cas des images à faible résolution. On remarque que le temps de calcul n'est jamais identique pour chaque application des algorithmes, étant donné qu'il est influencé par la position initiale des centroïdes, qui est calculée aléatoirement.

VI. Conclusion

Nous avons testé les segmentations *Fuzzy C-means* et *K-means* sur une série d'images en couleur, sous Matlab. Après un traitement des fonctions utilisées, nous avons pu observer que, de manière générale, les algorithmes étudiés permettent une bonne segmentation. Néanmoins, les images comportant des défauts (reflet, partie sombre, flou,…) donnent un résultat qui semble visuellement correct mais ne permet pas une bonne détection des caractères par OCR. Nous avons constaté que les deux algorithmes nécessitent une connaissance préalable du nombre de clusters à déterminer, ce qui rend impossible une éventuelle automatisation du processus. Par leur caractère itératif, ils s'avèrent inefficaces lorsque le nombre de clusters devient important.
Nous pouvons donc conclure que les algorithmes *Fuzzy C-means* et *K-means* sont efficaces pour la détection de caractères, mais ne sont pas appropriés à des images contenant un grand nombre d'objets. D'un point de vue temps de calcul, FCM s'avère être plus efficace pour des images à haute résolution alors que *k-means* est plus adapté aux images de faible résolution.

REFERENCES

- Mohamed Hachama et Bohoua-Nasse Franck-Olivier, *Une segmentation grossière et rapide des images en couleurs*, Ecole Nationale Supérieure des Télécommunications de Paris, DEA MVA 2003-2004.
http://www.tsi.enst.fr/tsi/enseignement/ressources/mti/gric/discuss.htm

- Samuel Drulhe et Martial Hue, *Segmentation des images couleurs : méthode d'Ohlander, Price et Reddy*, Telecom Paris, 2004.
http://www.tsi.enst.fr/tsi/enseignement/ressources/mti/OPR/index.php#opr

- David Corney, *Clustering with Matlab*, UCL Computer Science Department.
http://www.cs.ucl.ac.uk/staff/D.Corney/ClusteringMatlab.html

- Martine Collard et Nicolas Pasquier, *Fouille de données : Classification non supervisée*, Université de Nice Sophia-Antipolis, Laboratoire I3S.
http://deptinfo.unice.fr/~pasquier/m2miage/fd/clustering.pdf

- Jean-Marie Beaulieu, *Reconnaissance des formes : Classification et regroupement*, département d'informatique, Université Laval, 2002.
http://www.ift.ulaval.ca/Info_Courante/Vitrine/Syllabus_IFT64321.pdf

- H. D. Cheng, X.H. Jiang, Y. Sun & J. Wang, *Color image segmentation : advances and prospects*, Pattern Recognition, vol. 34, no. 9, 2001, pp. 2259-2281.
http://www.mathworks.com/products/fuzzylogic/demos.html?file=/products/demos/shipping/fuzzy/fcmdemo_codepad.html

- P. Lambert, H. Grecu, *A quick and coarse color image segmentation*, ICIP, 14-17 Septembre, 2003.
http://www.elsevier.com/wps/find/bookdescription.cws_home/675242/description#description

- Lilia Lazli et Mohamed Tayeb Laskri, *Nouvelle méthode de fusion de données pour l'apprentissage des systèmes hybrides*, Université Badji Mokhtar d'Annaba, 2003.
http://www-direction.inria.fr/international/arima/CARI04/pdf/CARI04_07.pdf