Hoël LE CAPITAINE

sous la direction de
Carl FRELICOT (EA 3165 MIA)
et Luc VACHER (UMR 6250 LIENSs)

Détermination automatique d'une zone de plage utile aux usages de loisirs et de tourisme.

Rapport Action incitative 2010
Université de La Rochelle -
Fédération de Recherche en Environnement
pour le Développement Durable (FREDD)

UMR 6250 LIttoral,
ENvironnement et Sociétés
CNRS-Université de La Rochelle

Laboratoire Mathématiques, Image et
Applications (EA 3165 MIA) de
l'Université de La Rochelle

Décembre 2010

Table des matières

Chapitre **1**

Méthode de segmentation

La méthode de segmentation considérée repose sur l'utilisation d'algorithmes non supervisés, c'est à dire sans connaissance préalable des caractéristiques précises de chacune des zones à extraire. Cette approche non supervisée permet de réduire de manière significative l'intervention humaine lors du traitement de chaque image. Au sein des méthodes non supervisées de classification, de nombreuses approches ont été proposées au cours des trente dernières années. Par souci de simplicité, nous adoptons une démarche dite itérative, qui consiste à déterminer et faire évoluer les centres de chacune des classes en fonction des données dont on dispose. Ces centres de classes correspondent en fait au prototype de chacune des régions, c'est à dire un objet synthétique représentant au mieux l'ensemble des pixels qui s'y rapportent.

A l'heure actuelle, le défaut majeur de ces méthodes est leur temps de calcul. En effet, elles requièrent de parcourir $i \times c$ fois l'ensemble des pixels de l'image, où i est le nombre d'itérations nécessaires à la convergence de l'algorithme, et c est le nombre de classes désirées par l'utilisateur. Sur des images de taille réduite, ce n'est aujourd'hui plus un problème, mais sur les images de plage considérées, les temps de calcul augmentent de manière drastique. Si l'algorithme se termine, ce qui n'est pas garanti sur des images de grande taille, la segmentation d'une seule image peut prendre jusqu'à 2 jours. Ceci n'est clairement pas envisageable en pratique.

Par conséquent, une nouvelle approche, toujours itérative, est proposée. Celle-ci repose sur une réduction de l'espace de représentation des pixels par quantification uniforme. En d'autres termes, chaque canal d'information couleur est réduit de 255 valeurs possibles à un nombre fixé par l'utilisateur, que l'on appellera dorénavant *bin*. Le parcours de l'algorithme ne se fait plus sur l'ensemble des pixels, mais sur l'ensemble des bins possibles, ce qui réduit de manière considérable les opérations à effectuer, en particulier si l'image est de grande taille. Cela apporte aussi l'avantage d'une relative indépendance des temps de calcul par rapport à la taille de l'image. Cette dépendance est en effet relative, puisque certaines étapes de traitement, et notamment de lissage, sont toujours contraintes par l'information spatiale.

Les algorithmes classiques de classification se fondent généralement sur la distance Euclidienne entre deux points afin de discriminer et d'inférer une appartenance à une classe. Lorsque les classes sont linéairement séparables dans l'espace de description, cette représentation de l'espace ne présente pas d'inconvénients. A l'inverse, lors de problèmes pratiques réels, les classes ne sont que très peu souvent linéairement séparables, et se chevauchent dans l'espace de description. Afin de remédier à ce problème, nous proposons d'utiliser des fonctions dites *noyaux*. Brièvement, ces fonctions ont la particularité de procé-

der à des projections dans des espaces de dimension potentiellement infinie les points considérés. Dans cet espace, une distance Euclidienne est calculée, mais elle permet la discrimination non linéaire par rapport à l'espace d'origine. Pour un coût calculatoire équivalent, une distance dans un espace de dimension infinie est calculée, et permet d'obtenir de meilleures classifications, puisque non linéaires.

La plupart des images issues de photographies ne sont pas homogènes localement. Soit parce que l'acquisition à créé du bruit résiduel, soit parce que la réalité n'est elle-même homogène. Afin de pouvoir définir des zones homogènes sans constellations de nouvelles zones à l'intérieur de celles-ci, il est nécessaire de procéder à un lissage de l'image. Classiquement, un premier lissage de type Gaussien de l'image est opéré. Ce lissage prend un paramètre principal en compte : la taille du voisinage considéré. Plus cette taille est grande, plus l'image résultante sera lissée. A l'extrême, les contours de chacun des objets de l'image ne seront plus discernables. Il faut donc choisir précautionneusement cette taille. Une discussion plus détaillée du choix et de l'influence de ce paramètre est donnée en Section 2.1. A l'issu de la procédure de segmentation, une partition floue des données est obtenue. En d'autres termes, pour chaque pixel de l'image, lui correspond un nombre c de valeurs. Ces c valeurs constituent les degrés de ressemblance du pixel à chacune des zones homogènes. Afin de produire une segmentation spatiale plus fluide, nous proposons de régulariser cette matrice de partition floue. Ici encore, cette régularisation prend pour paramètre la taille de voisinage considérée. Afin de réduire le nombre de choix laissé à l'utilisateur, cette taille est fixée de dimension égale à la dimension du pré filtrage de l'image.

Un autre problème régulièrement pointé du doigt est l'initialisation des centres de chacune des classes. Cette initialisation se fait habituellement de manière aléatoire : soit on choisit de manière aléatoire c points de l'ensemble des données, et fixons ainsi les centres ; soit pour chacun des centres les valeurs sont déterminées de manière aléatoire. Cette façon d'initialiser l'algorithme présente plusieurs inconvénients. D'une part, pour les mêmes données, les résultats de l'algorithme ne donneront pas les mêmes résultats (non déterministe), et d'autre part la convergence n'est pas garantie pour des centres ne reflétant pas les données. Nous proposons donc un nouvel algorithme, déterministe, permettant l'initialisation des centres des classes. Cet algorithme repose sur un réordonnancement des données par le biais d'une fonction σ. Une fois les données ordonnées, nous procédons une répartition uniforme en c groupes, puis appliquons la fonction inverse σ^{-1} afin de déterminer les effectifs de chacun des groupes. Enfin, le centre de chacune des classes est obtenu en considérant la moyenne arithmétique des effectifs. Cette procédure est appelée *Séparation Ordonnée*.

L'algorithme de segmentation peut ainsi se décomposer de la façon suivante

- Lissage de l'image I suivant une fenêtre de taille $N \times N$ (N impair)
- Quantification de l'image en q^3 valeurs
- Initialisation des c centres par *Séparation Ordonnée*
- tant que *nombre itérations* $<$ *imax* ET *critère d'arrêt non rempli*
  - mise à jour des appartenances des pixels à chacune des classes
  - mise à jour des centres des classes
  - calcul de la norme matricielle entre deux matrices de centres successives
- fin tant que

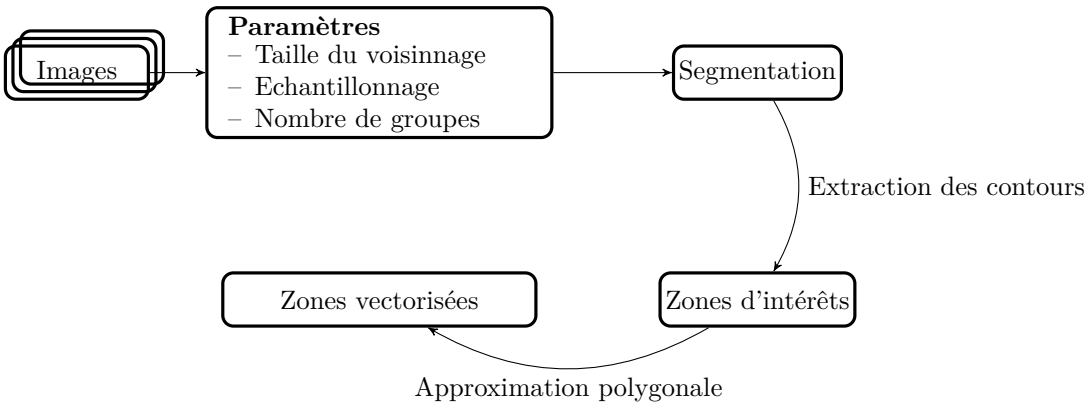


FIGURE 1.1: Schéma résumant les différentes étapes d'obtention des zones de plage utiles.

- Régularisation des appartenances
- Détermination de la classe de degré maximum d'appartenance pour chacun des pixels

Un schéma global de fonctionnement est décrit en Figure 1.1.

Chapitre 2

Analyse des résultats

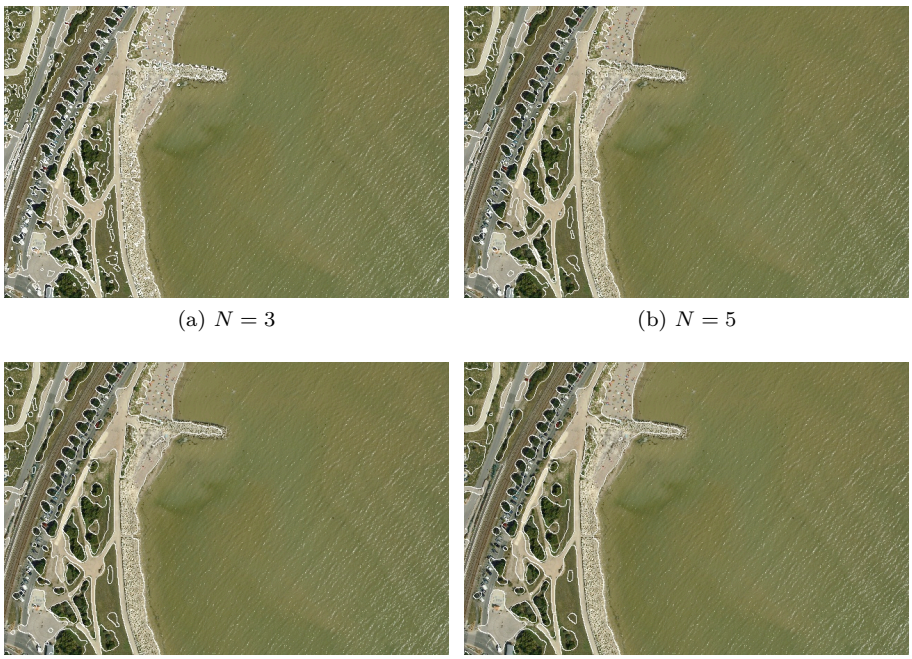
Dans cette section, nous proposons une analyse des résultats obtenus par l'algorithme de segmentation sur différentes images de plages présentant des caractéristiques bien distinctes. Afin de bien comprendre l'influence des paramètres de l'algorithme, une étude détaillée est conduite pour chacun d'eux. En particulier, pour analyser l'influence d'un paramètre, nous fixons les valeurs des deux autres paramètres, et faisons évoluer le paramètre étudié. Pour autant, afin de ne pas multiplier les images et rendre impossible une analyse, nous choisissons deux images particulières de la base de données. Par la suite, une analyse globale pourra être menée avec la connaissance de la vérité terrain. Ces deux images sont données en Figure 2.1.

Comme chacun peut le noter, les problématiques des deux images sont différentes. En ce qui concerne l'image de gauche, la difficulté principale est liée à la présence de rochers, ayant sensiblement la même couleur que le sable. Le second problème provient du chemin bordant la plage, également de la même couleur que le sable. Pour l'image de gauche, le problème est cette fois-ci la présence de zones comportant de l'herbe ou des arbres au beau milieu de la plage. Ces zones ne sont pas considérée comme utiles et doivent donc être détectées en tant que telles.

Dans ce qui suit, nous étudions successivement l'influence de la taille du voisinage, du nombre de bins et du nombre de classes. Naturellement, plus les valeurs de chacun de ces paramètres sont importantes, plus les temps de calcul sont élevés. En d'autres termes, il s'agit de minimiser la valeur de ceux-ci tout en gardant des performances de segmentation acceptables.

(a) Image *Aytré* (b) Image *Oléron*

Figure 2.1: Les deux images objets de l'analyse.



(a) $N = 3$ (b) $N = 5$

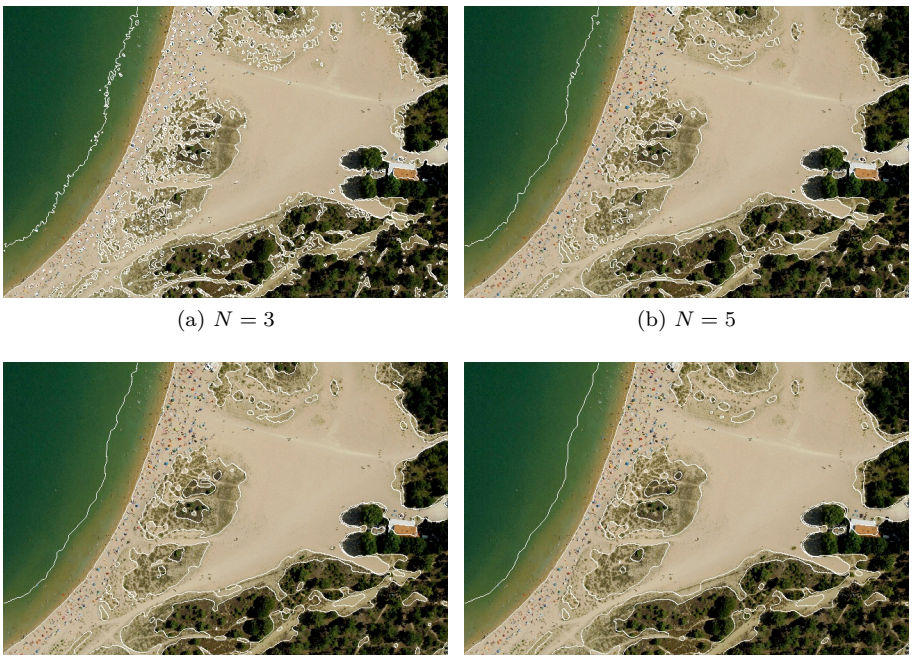
(c) $N = 7$ (d) $N = 9$

FIGURE 2.2: Segmentation de *Aytré* pour différentes tailles de voisinage N.

2.1 VOISINAGE

Dans un premier temps, nous étudions l'influence de la taille de voisinage considéré autour de chaque pixel. Cette taille de voisinage est utilisée à deux reprises lors de l'algorithme. Une première fois lors du lissage préalable de l'image, puis lors de la régularisation de la matrice d'appartenance. Nous faisons varier cette taille dans l'intervalle suivant : $N = \{3, 5, 7, 9\}$. Au delà de ces valeurs, le temps de calcul augmente de manière importante, pour des gains en terme de performance très relatifs comme nous le verrons par la suite. Les deux autres paramètres, c'est à dire le nombre de bins et le nombre de classes, sont fixés à $c = 3$ et $q = 5$, respectivement. Les résultats pour les deux images *Aytré* et *Oléron* sont donnés en Figure 2.2 et 2.3, respectivement. Les conséquences de l'augmentation de la taille du voisinage sont les suivantes :

- la régularisation est plus grande. En d'autres termes, les zones découvertes ont une surface de plus en plus grande. Ceci s'explique par le fait que deux zones différentes proches (spatialement) vont fusionner si le voisinage considéré est plus grand que l'écart maximum entre les deux zones.
- à l'inverse, des zones de faible surface seront supprimées et intégrées à la zone de plus grande dimension majoritaire dans le voisinage d'intérêt, et ce toujours pour le même raison.
- de manière générale, et comme l'on pouvait s'y attendre, plus la taille du voisinage considéré est importante, plus les détails de l'image sont supprimés, et donc non traités lors du processus de segmentation.



(a) $N = 3$ (b) $N = 5$

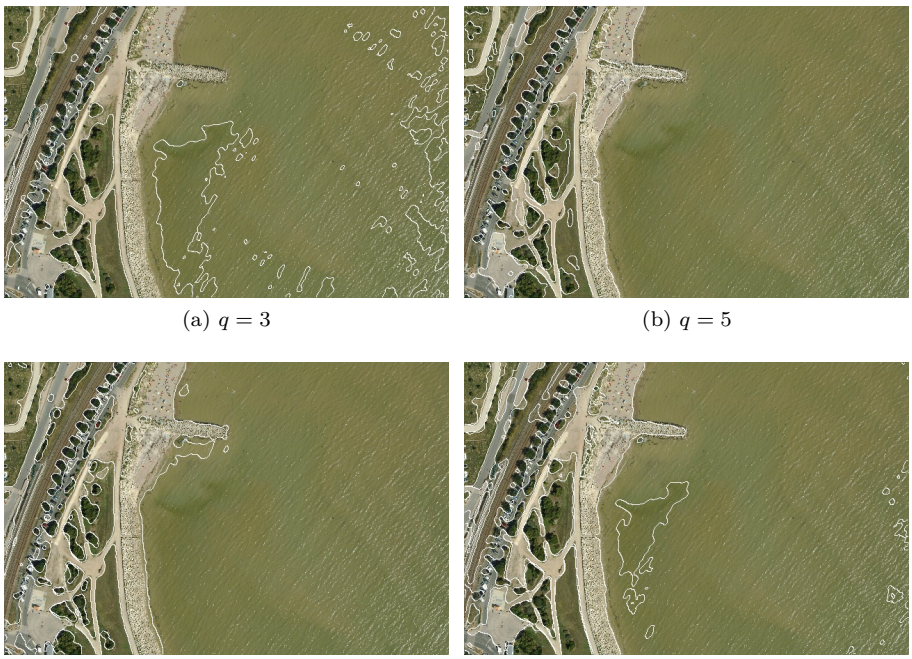
(c) $N = 7$ (d) $N = 9$

FIGURE 2.3: Segmentation de *Oléron* pour différentes tailles de voisinage N.

A noter que quels que soient les paramètres de voisinage fixés, les zones de rocher et de sable de l'image *Aytré* ne sont pas distinguées. A l'inverse, les zones d'herbes et d'arbre de l'image *Oléron* sont bien discriminées par rapport à la zone de sable.

2.2 ECHANTILLONNAGE

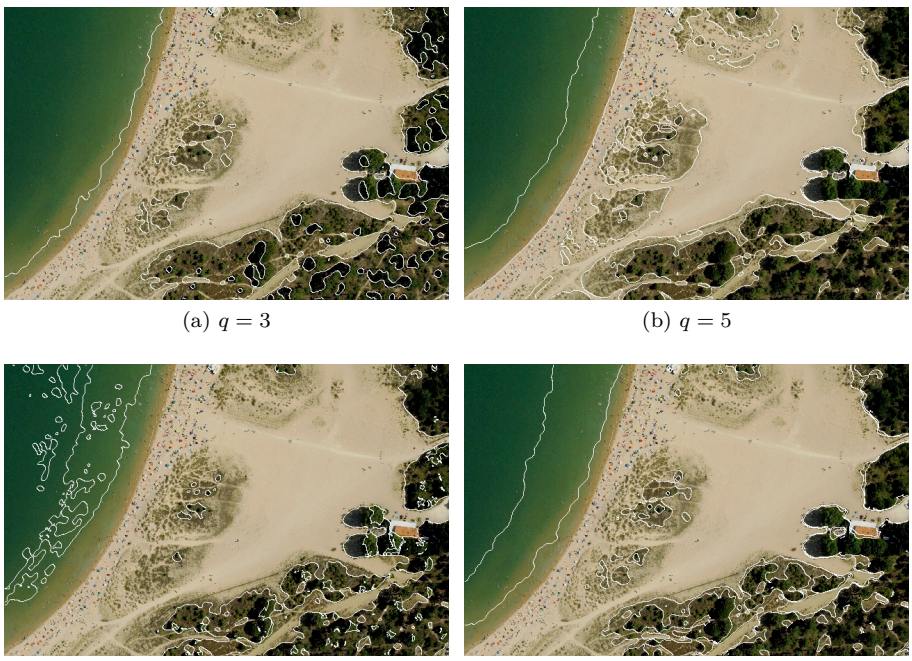
Dans cette section, la taille du voisinage est fixée à $N = 7$ et le nombre de classes à $c = 3$. Le nombre de bins utilisés pour décrire chacun des pixels est $q = \{3, 5, 7, 9\}$. Nous rappelons que cela conduit à q^3 valeurs possibles de pixels sur l'ensemble de l'image. Les résultats pour les deux images *Aytré* et *Oléron* sont donnés en Figure 2.4 et 2.5, respectivement. Lorsque q est fixé à 3, le nombre d'éléments descriptifs (27) est insuffisant, et les segmentation obtenues pour les deux images ne correspondent pas à la vérité terrain. En particulier, les zones extraites appartiennent à la mer, et la séparation mer/sable n'est pas distinguée. La valeur $q = 5$ semble la plus appropriée pour les deux images considérées, puisque les distinctions sont faites correctement. A noter toutefois, qu'une fois encore les rochers et le sable ne sont pas distingués. Lorsque la valeur de q est augmentée à 7 et 9, des problèmes similaires apparaissent, dans la mesure ou le nombre de valeurs différentes est importante, il existe de nombreuses solutions pour les partager en groupes. Le nombre réduit de couleurs utilisées (125) permet une exécution rapide de l'algorithme, et la différence majeure avec les algorithmes classiques de segmentation.



(a) $q = 3$ (b) $q = 5$

(c) $q = 7$ (d) $q = 9$

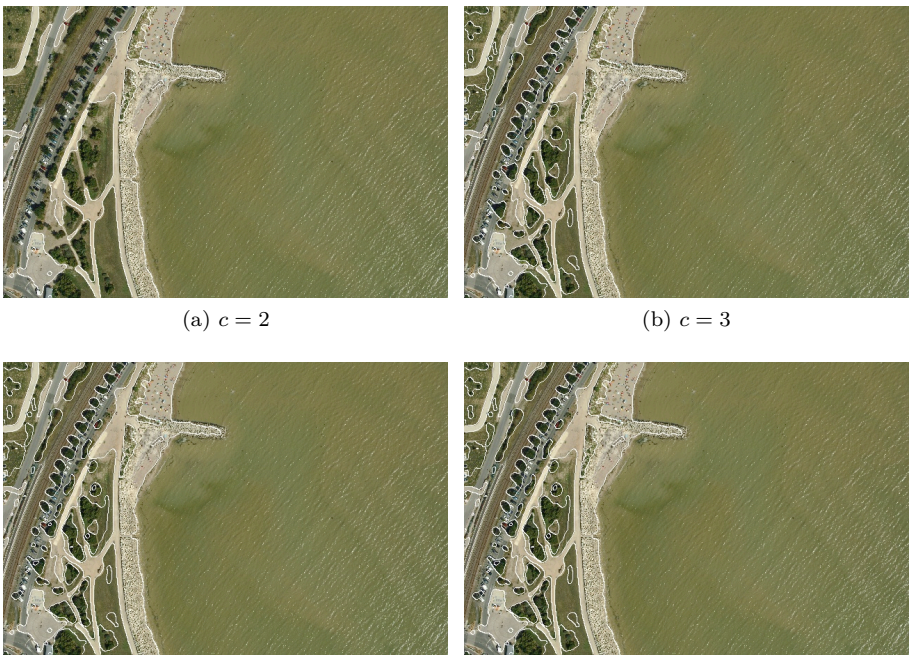
FIGURE 2.4: Segmentation de *Aytré* pour différentes quantifications.



(a) $q = 3$ (b) $q = 5$

(c) $q = 7$ (d) $q = 9$

FIGURE 2.5: Segmentation de *Oléron* pour différentes quantifications.



(a) $c = 2$ (b) $c = 3$

(c) $c = 4$ (d) $c = 5$

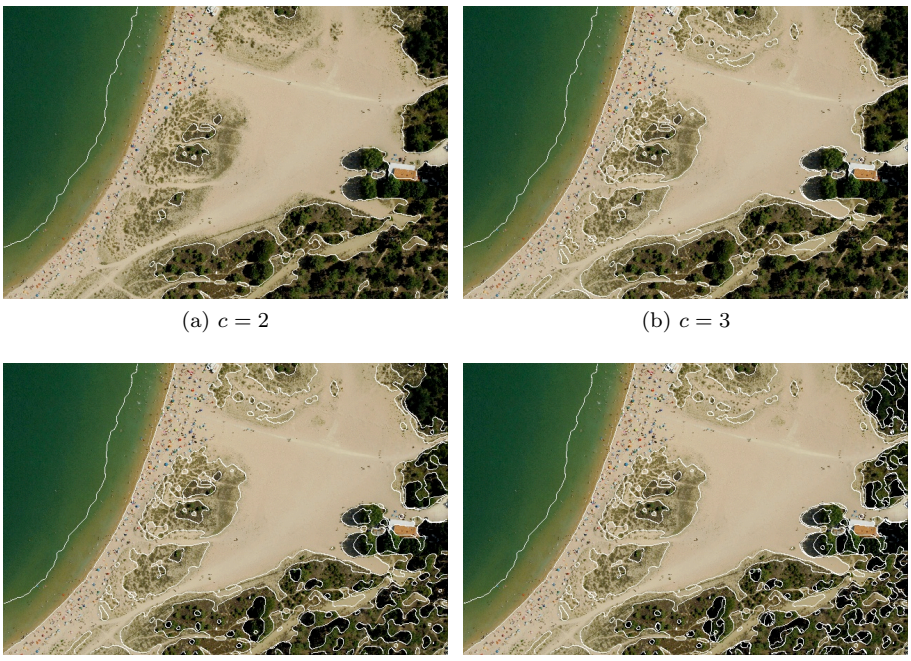
FIGURE 2.6: Segmentation de *Aytré* pour différents nombres de classes.

2.3 NOMBRE DE GROUPES

Dans cette section, la taille du voisinage est fixée à $N = 7$ et le nombre de bins à $q = 5$. Nous faisons varier le nombre de classes désirées $c = \{2, 3, 4, 5\}$. Les résultats pour les deux images *Aytré* et *Oléron* sont donnés en Figure 2.6 et 2.7, respectivement.

Sur l'image *Aytré*, quel que soit le nombre de classes considéré, la distinction entre pierres et plage n'est pas faite. Au vu des résultats, les choix $c = 2$ et $c = 3$ semblent donner les résultats intéressants. Au delà, des zones internes à des zones déjà considérées comme homogène apparaissent, et n'apportent donc aucun information supplémentaire.

Sur l'image *Oléron*, lorsque le nombre de groupes est fixé à deux, la séparation entre la mer et la plage n'est pas faite : les deux centres des classes correspondent en fait à la couleur vert foncé (incluant une partie de la mer et les arbres) et beige (correspondant à la plage ainsi qu'au début de la mer). Augmenter le nombre de classes permet d'obtenir au moins un centre supplémentaire, d'où la séparation entre la mer et la place pour $c > 2$. La valeur $c = 3$ apparait ici encore comme un choix intéressant. Au delà de 3, les principales différences apparaissent dans les zones de foret, où une plus grande spécification des zones est observée. Cette spécification ne présente pas d'intérêt notoire dans l'application considérée.



(a) $c = 2$ (b) $c = 3$

(c) $c = 4$ (d) $c = 5$

FIGURE 2.7: Segmentation de *Oléron* pour différents nombres de classes.

CHAPITRE **3**

Appendice technique

3.1 INTRODUCTION

Image segmentation can be defined as the process of merging pixels having similar features into the same groups, or regions. The segmented image is then the union of distinct groups, where pixels of homogeneous regions are associated to the same groups. Numerous techniques have been proposed in the literature, where color, texture or edges features are used to populate each group. At the very beginning, only gray level images were considered in the analysis. However, as color images are becoming the norm, and due to advancements in both color technology and computation power, the interest of proposing color image segmentation techniques has grown. Moreover, image segmentation is needed and used in a wide variety of applications such as geographical imaging, medical imaging, or video surveillance.

This paper is organized as follows. Section 3.2 first recalls some basic knowledge on iterative clustering algorithms, focusing on their fuzzy approaches. Then, a new initialization of cluster centers is proposed, along with some numerical examples exhibiting its efficiency in terms of both accuracy and convergence speed.

3.2 ITERATIVE CLUSTERING ALGORITHMS

Algorithms

Let $X = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ be a data set having c clusters. Each sample $\mathbf{x}$ is represented by a set of p features, leading to a $(n \times p)$ *pattern matrix*. A partition of the data set X into c clusters is presented as mutually disjoint subsets X_i of X such that $X_i \cup \cdots \cup X_c = X$ and $X_i \cap X_j = \emptyset$ for any $i \neq j$.

Then, the hard c-mean algorithm minimizes the following objective function defined by the within-cluster distances :

$$J = \sum_{i=1}^{c} \sum_{\mathbf{x} \in X_i} \|\mathbf{x} - \mathbf{v}_i\|^2, \tag{3.1}$$

which can be rewritten as

$$J = \sum_{k=1}^{n} \sum_{i=1}^{c} u_{ik} \|\mathbf{x}_k - \mathbf{v}_i\|^2, \tag{3.2}$$

where $u_{ik} = 1$ if $\mathbf{x}_k \in X_i$, and 0 otherwise.

However, in many real situations, overlapping clusters reduce the effectiveness of crisp clustering methods. Ruspini first proposed the notion of fuzzy

partition [1], where samples may partially belong to several clusters through the idea of partial membership. Some years later, Dunn [2] introduced a modification of the crisp objective function J by adding an exponent $m = 2$ to the individual membership degrees :

$$J_2 = \sum_{k=1}^{n} \sum_{i=1}^{c} u_{ik}^2 \|\mathbf{x}_k - \mathbf{v}_i\|^2, \tag{3.3}$$

that have been generalized by Bezdek in [3], where the exponent m is strictly greater than 1 :

$$J_m = \sum_{k=1}^{n} \sum_{i=1}^{c} u_{ik}^m \|\mathbf{x}_k - \mathbf{v}_i\|^2, \tag{3.4}$$

The solution of (3.4) is generally obtained by an alternating optimization procedure that successively updates the cluster centers and the partition matrix as follows :

$$\mathbf{v}_i = \frac{\sum_{k=1}^{n} u_{ik}^m \mathbf{x}_k}{\sum_{k=1}^{n} u_{ik}^m} \tag{3.5}$$

$$u_{ik} = \frac{1}{\sum_{j=1}^{c} \left(\frac{\|\mathbf{x}_k - \mathbf{v}_i\|}{\|\mathbf{x}_k - \mathbf{v}_j\|} \right)^{2/(m-1)}} \tag{3.6}$$

The choice to first initialize a random partition matrix or cluster centers is let to the user, both being used in the literature. The algorithm stops when the centroids stabilize, *i.e.* the matrix norm between two successive V is below a given threshold. Equivalently, the entire procedure can be shifted one half cycle, so that initialization and termination is done on U. Naturally, in terms of speed and storage, there are some advantages to initialize and terminating with V.

Centroids initialization

Iterative clustering methods do not guarantee a unique final clustering because we obtain different results with different initial clusters (or partitions). In particular, it has been shown that these algorithms give better results when the initial input is sufficiently close to the final solution [4]. However, most of the practitioners initialize their centers in a random manner, which heavily affects the results. Several methods have been proposed for the initialization of the cluster centers, *e.g.* [5]. An additional reason to correctly initialize the cluster centers is that it allows to reach the local extrema of the objective function much more quickly, resulting in a more usable algorithm for large scale real world problems. However, most of the methods that allow to initialize cluster centers are computationally expensive. For instance, in [5], the method require to run the c-means algorithm p times on n 1-dimensional samples, and then p times the same algorithm on n p-dimensional samples, which is intractable for large scale data sets.

In this subsection, we propose a new, efficient, yet simple, manner of initializing the c cluster centers, called *Ordering-split*. For each sample $\mathbf{x}_k$, we define its relative mean by

$$m_k = \frac{1}{p} \sum_j x_{kj} \tag{3.7}$$

Note that we are working on features coming from each channel of an image so that the scale of individual features does not differ. If the features do not hold this property, a normalization of the data is required. We introduce the permutation function σ such that the sequence $m_{\sigma(k)}$ is increasing. We propose to split the n relative means as follows. Starting from the hypothesis that the clusters are equally distributed, we uniformly split the vector $\mathbf{m}$ into c groups. In other terms, we set $c-1$ indices, say $\ell_1, \cdots \ell_{c-1}$, such that the $\ell_i - \ell_{i-1}$ are roughly equals [1]. More formally, we set

$$\ell_i = i * \lfloor n/c \rfloor \tag{3.8}$$

where $\lfloor . \rfloor$ is the floor function. Then we iteratively build c subsets S of n corresponding to each index :

$$S_i = \{\ell_{i-1} + 1, \cdots, \ell_i\} \tag{3.9}$$

where $\ell_0 = 0$ by convention. We obtain the subset of indices in each cluster by applying the inverse function :

$$C_i = \sigma^{-1}(S_i) \tag{3.10}$$

Finally, each cluster center is obtained as follows :

$$\mathbf{v}_i = \frac{1}{|C_i|} \sum_{j \in C_i} \mathbf{x}_j \tag{3.11}$$

where $|C_i|$ is the cardinality of the set C_i. We give in Algorithm 1 the procedure of determination of centers

Algorithm 1 Initialization of cluster centers

Require: X : dataset, c : number of clusters
procedure ORDERING-SPLIT(X, c)
  compute $\mathbf{m}$ by using (3.7) for each $k \in \{1, \cdots, n\}$
  apply to $\mathbf{m}$ the ordering function σ
  for $i = 0$ to $c - 1$ **do** ▷ uniform splitting
    $\ell_i \leftarrow i * \lfloor n/c \rfloor$
  end for
  for $i = 1$ to c **do** ▷ build the subsets
    $S_i \leftarrow \{\ell_{i-1} + 1, \cdots, \ell_i\}$
    $C_i \leftarrow \sigma^{-1}(S_i)$
    $\mathbf{v}_i \leftarrow \frac{1}{|C_i|} \sum_{j \in C_i} \mathbf{x}_j$
  end for
  return v ▷ **v** is the matrix of centers
end procedure

Example 1 *Let us consider the well known Iris data set [6]. The real (physical) cluster centers are*

$$\begin{pmatrix} 5.006 & 3.418 & 1.464 & 0.244 \\ 5.936 & 2.770 & 4.260 & 1.326 \\ 6.588 & 2.974 & 5.552 & 2.026 \end{pmatrix}$$

1. Note that other hypothesis could be provided, where some information about the clusters distribution is used.

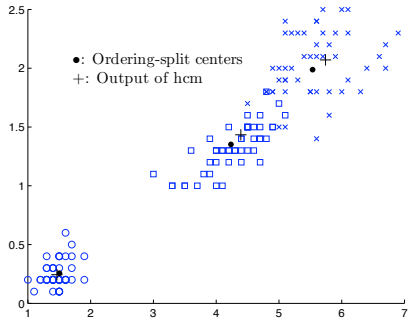


FIGURE 3.1: Features 3 and 4 of the iris data, the centers obtained by the ordering-split method (•) and the centers obtained by applying the hard c-means algorithm (+).

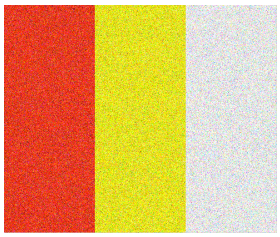


FIGURE 3.2: A 500×600 (i.e. $n = 300,000$) color image, with Gaussian white noise ($\sigma = 0.1$).

The cluster centers obtained by the Ordering-split method are

$$\begin{pmatrix} 4.992 & 3.370 & 1.502 & 0.258 \\ 5.824 & 2.748 & 4.236 & 1.352 \\ 6.714 & 3.044 & 5.536 & 1.988 \end{pmatrix}$$

We run the hard c-mean algorithm 100 *times, the average number of iterations needed for convergence is 12, while the same algorithm initialized with the Ordering-split method needs only 5 iterations to stabilize. On a reduced number of samples, the gain is not very important, but for large data sets, the computation time is greatly reduced.*

For comparison purpose, we give the results obtained with our method and randomly selected cluster centers. We use the hard c-means algorithm for simplicity. We use as data X a simple color image (see Figure 3.2), where we add a Gaussian white noise with various variances σ. We run each algorithm 100 times, and we give in Table 3.1 the mean computation times. The time needed for the computation of initial cluster centers is obviously added to the clustering time of the hard c-means algorithm. We also provide the cluster center proximity index (CCPI) defined by

$$CCPI = \frac{1}{c \times p} \sum_i \sum_j \left| \frac{v_{ij}^\star - v_{ij}}{v_{ij}^\star} \right|, \tag{3.12}$$

TABLE 3.1: Comparison of centers initialization.

	comp. time (sec.)	CCPI	Accuracy
$\sigma = 0.05$			
Random	1.286	0.729	98.33%
Ordering-split	0.578	0.088	98.33%
$\sigma = 0.1$			
Random	1.870	0.714	92.18%
Ordering-split	0.787	0.118	92.27%
$\sigma = 0.5$			
Random	2.776	0.787	53.13%
Ordering-split	1.670	0.181	65.90%

which measures the degree of closeness between the final cluster centers and the desired cluster centers, where $\mathbf{v}_i^\star$ is the desired cluster center of the ith cluster. Clearly, the less the CCPI, the better the result. Note that this index differs from the one proposed in [5] where the index is computed on initial cluster centers and desired cluster centers. We finally give the accuracy performance defined by the ratio of correctly labeled pixels over the total number of pixels.

According to Table 3.1, we see that the Ordering split method allows to obtain a faster algorithm, with an output that is more close to the reality, and producing a better segmented image than the method which consists in randomly initializing cluster centers.

Finally, we propose to compare our method with the random initialization method and with the one proposed in [5] for 4 data sets : iris, wine, Ruspini and letter image. The results of the random and CCIA methods are taken from [5].

3.3 CLUSTERING FOR IMAGE SEGMENTATION

Spatial consideration in clustering algorithms

Whilst the conventional FCM algorithm works well on noise-free images, it is very sensitive to local irregularities, which occur very often in real images. This sensitivity is due to the absence of consideration of the spatial context of each pixel. In [7], Ahmed *et al.* modify the original objective function by adding a penalty term that allows the memberships of each pixel to be influenced by its neighborhood. Practically, the new objective function is defined as

$$J_S = J_m + \frac{\alpha}{N_R} \sum_{k=1}^{n} \sum_{i=1}^{c} u_{ik}^m \sum_{r \in N_k} \|\mathbf{x}_r - \mathbf{v}_i\|^2 \tag{3.13}$$

where N_R is the cardinality of N_k, which stands for the set of neighbors in a window around a pixel $\mathbf{x}$. The parameter α is used to control the effect of the neighbor terms. However, computing the neighbor term in each iteration is computationally expensive. Moreover, setting the parameter α is not easy,

TABLE 3.2: Comparison of cluster centers proximity index (CCPI).

	CCPI	Error rate
Iris		
Random	0.8909	23.6%
CCIA	0.0396	11.33%
Ordering-split	0.0175	10.66%
Wine		
Random	0.3557	5.61%
CCIA	0.1869	5.05%
Ordering-split	0.1387	4.49%
Ruspini		
Random	1.2274	8.8%
CCIA	0.0361	4.0%
Ordering-split	0.0173	0.0%
Letter		
Random	0.1572	8.47%
CCIA	0.0608	8.55%
Ordering-split	0.0416	8.22%

because a slight variation of α produce very different segmentations. This algorithm is denoted *FCM_S* in the sequel.

In [8], the author propose another objective function where the relationship between neighboring pixels is taken into account. In particular, they replace the usual Euclidean distance between pixels and centroids by a new distance. This distance is a weighted mean distance of the pixel and its neighbors to each centroid. However, here again, in each iteration, all the pixels of the image are considered, leading to a large computation time.

In [9], the author propose to reduce the computation time of the solutions derived from (3.13) by computing in advance the mean value of the pixel within the specified window :

$$J_{S1} = J_m + \alpha \sum_{k=1}^{n} \sum_{i=1}^{c} u_{ik}^m \|\overline{\mathbf{x}}_k - \mathbf{v}_i\|^2 \tag{3.14}$$

where $\overline{\mathbf{x}}_k$ is the mean of neighboring pixels in the window around $\mathbf{x}_k$. Additionally, they also propose another objective function J_{S2} where $\overline{\mathbf{x}}_k$ is the median value of the neighboring pixels. Then, they propose to use kernel-induced distances (see Section 3.4) instead of the usual Euclidean one. The corresponding algorithms are respectively denoted as *KFCM_S1* and *KFCM_S2* in the sequel. More recently, Yang and Tsai [10] proposed to adapt the parameter α of (3.14) to each cluster, namely α_i :

$$J_G = J_m + \sum_{k=1}^{n} \sum_{i=1}^{c} \alpha_i u_{ik}^m \|\overline{\mathbf{x}}_k - \mathbf{v}_i\|^2 \tag{3.15}$$

Moreover, since they use kernel-induced distance (see Section 3.4), they also propose to automatically set the parameters of the Gaussian kernel. The corresponding algorithm is presented as a generalized version of *KFCM*. Here again, the authors introduce the same possibility to use mean-based or median-based spatial consideration. Consequently, the algorithms are respectively denoted as *GKFCM_S1* and *GKFCM_S2* in the sequel.

Finally, in [11], the authors propose another modification of the objective function somewhat similar to (3.13). A penalty term G_{ik}, defined by

$$G_{ik} = \sum_{j \in N_i, i \neq j} \frac{1}{d_{ij}+1}(1-u_{jk})^m \|\mathbf{x}_j - \mathbf{v}_i\|^2, \tag{3.16}$$

is added to J_m. The term d_{ij} is the spatial Euclidean distance between pixels i and j. The obtained corresponding updating functions are called *FLICM*. Naturally, *FLICM* suffers of the same problem as all the previous algorithms, a high computation time.

Speeded up clustering algorithms

The main drawback of such iterative clustering algorithms is their running time. In [12], the authors proposed a fast and accurate clustering method of images. The reduction is operated by aggregating similar examples and using the weighted prototype in the clustering, giving the *brFCM* algorithm. In order to speed up the segmentation, Szilagyi *et al.* [13] used the idea of Eschrich *et al.* and proposed the *EnFCM* algorithm, which is the *brFCM* algorithm applied on a smoothed image as follows. They first construct a linearly-weighted sum image with local neighbors of each pixel as follows :

$$\xi_i = \frac{1}{1+\alpha}\left(\xi_i + \frac{\alpha}{N_R}\sum_{j \in N_i}\xi_j\right) \tag{3.17}$$

where N_i is the set of neighbors of the pixel I_j, and the parameter α controls the influence of the neighbors. Instead of considering each pixel of the image, the objective function uses the number of gray levels in the image, which drastically reduces the computation time, since the number of gray level of an image is generally much lower than the number of pixels. In [14], Cai *et al.* propose an improvement of the *EnFCM* algorithm by adding a local similarity measure S_{ij}. The new image to be clustered is then defined as

$$\xi_i = \frac{\sum_{j \in \mathcal{N}(\mathbf{x})} S_{ij}\mathbf{x}_j}{\sum_{j \in \mathcal{N}(\mathbf{x})} S_{ij}} \tag{3.18}$$

where S_{ij} is a factor incorporating the spatial and gray level relationships in the neighborhood of $\mathbf{x}$. They propose various definitions of the local similarity measure. The first one is given by

$$S_{ij} = \begin{cases} \exp\left(-\frac{\max(|p_i-p_j|,|q_i-q_j|)}{\lambda_s} - \frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{\lambda_g \sigma_i^2}\right) & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases} \tag{3.19}$$

Using this similarity measure leads to the algorithm *FGFCM*. They also propose two other local similarity measures, that we denote *FGFCM_S1* and

FGFCM_S2. The corresponding similarity measure are respectively defined by $S_{ij} = 1$ for all i and j, so that ξ_i is equal to the mean of the neighbors, and $S_{ij} = \text{median}(\mathbf{x}_j)$, so that ξ_i is the median value of the neighbors. *EnFCM*, *FGFCM* and their variants provide quite good segmentation. However, they heavily depend on the internal parameters λ_s and λ_g, or a. Moreover, to the best of our knowledge, they are restricted to gray level images, and there are no propositions dedicated to color images.

A first proposition

Nowadays, processing an image without pre-processing and regularization is pointless. In [15], the authors proposed the bilateral filtering procedure, which is an anisotropic approach based on both spatial and photometric considerations. Formally, the filtered image I' is obtained by

$$I' = \frac{\sum_{j \in \mathcal{N}(\mathbf{x})} w(x,j) I(j)}{\sum_{j \in \mathcal{N}(\mathbf{x})} w(x,j)} \tag{3.20}$$

where $w(x,j)$ are the weights applied to every pixel j in $\mathcal{N}(\mathbf{x})$. The weights are decomposed by a conjunction into two weights corresponding to the spatial and the color weights, $w(x,j) = w_s(x,j) \times w_c(x,j)$. For instance, in [15], w_s and w_c are respectively defined by

$$\begin{aligned} w_s(x,j) &= \exp\Big(- \frac{d_s^2(x,j)}{2\sigma_s^2} \Big) \\ w_c(x,j) &= \exp\Big(- \frac{d_c^2(x,j)}{2\sigma_c^2} \Big) \end{aligned} \tag{3.21}$$

In fact, the proposition of [14] consists in adding a bilateral filter process before the clustering algorithm. It can be shown that (3.18) reduces to (3.20) if we use $d_s = L_\infty$ and $d_c = L_2$, where L_p stands for the p-norm. Extending the gray level algorithm [] to color spaces is not immediate. If we use the same resolution, and take the RGB space, it would lead to compute membership and center matrices $2^8 * 2^8 * 2^8 = 2^{24}$ times. In other terms, this would be equivalent to run the usual fuzzy c-means algorithm on a (4096×4096) color image. This is intractable in practice. In order to reduce the computation time, we propose a quantization of the color space into q_i bins, i corresponding to the number of the channel. Generally, most of the color spaces use three channels, so that we have to define q_1, q_2 and q_3. Obviously, this can be extended to any multi spectral images, or images where several color spaces are used to describe each pixel (e.g. RGB, HSV and CIE $L * a * b*$). In the sequel we consider that each color component is divided into the same number of bins, implying $q_1 = q_2 = q_3 = q$.

Various studies have shown that many color spaces proposed for computer graphic applications are not well adapted to image processing. As pointed out in [16], a convenient representation should yield distances and provides independence between chromatic and achromatic components. For this reason and comparison purpose, we use the CIE $L * a * b*$ color space. Moreover, the advantage of quantizing the $L*a*b*$ information rather than RGB information is that, if approximately uniform $L*a*b*$ is uniformly quantized, a constant distance between and any two quantization levels will result in small variation of

perceptual color difference, unlike nonuniform RGB data where this variation can be very large.

Therefore we define the new objective function to be minimized as

$$J_Q = \sum_{i=1}^{c} \sum_{q_1=1}^{q} \sum_{q_2=1}^{q} \sum_{q_3=1}^{q} h_{q_1,q_2,q_3} u_{i,q_1,q_2,q_3}^m \|\mathbf{x}_{q_1,q_2,q_3} - \mathbf{v}_i\|^2 \tag{3.22}$$

where $\|.\|$ is a convenient norm in the quantized space, e.g. the Euclidean one. For convenience, we rewrite the objective function as

$$J_Q = \sum_{i=1}^{c} \sum_{l=1}^{q^3} h_l u_{il}^m \|\mathbf{x}_l - \mathbf{v}_i\|^2 \tag{3.23}$$

Since the gradient of J_Q with respect to u_{il} and $\mathbf{v}_i$ vanishes when reaching the local optima, and knowing that the vector $u_{.l}$ sum up to one for any l, we obtain

$$u_{il} = \frac{\|\mathbf{x}_l - \mathbf{v}_i\|^{-2/(m-1)}}{\sum_{j=1}^{c} \|\mathbf{x}_l - \mathbf{v}_j\|^{-2/(m-1)}} \tag{3.24}$$

and

$$\mathbf{v}_i = \frac{\sum_{l=1}^{q^3} h_l u_{il}^m \mathbf{x}_l}{\sum_{l=1}^{q^3} h_l u_{il}^m} \tag{3.25}$$

Although the introduction of a bilateral filtering process before clustering improves the effectiveness of segmentation, it still lack enough robustness and neighborhood importance should be taken into account in the clustering algorithm. To this aim, we propose, instead of considering the entire image, to regularize the partition matrix, based on the neighborhood of each pixel. The advantage of this proposition is to get rid of the selection of the crucial weight parameter α in the methods of []. This parameter ensures a balance between robustness to noise and effectiveness of preserving details. Hence it is hard to set and have considerable impact on the performances. However, due to quantization, the elements of the partition matrix do not have spatial relationships. To overcome this problem, we introduce a mapping of the quantized partition matrix of size $(c \times q^3)$ to the usual partition matrix of the pixels, having size $c \times (m \times n)$, where m and n are the width and the height of the image. The basic idea of this mapping is, from a given pixel of the image, to obtain its corresponding bin, say l_i, in the quantized space. Then, this pixel inherits from the membership degrees of l_i obtained by $u_{.l_i}$. In order to avoid ambiguities, an element of the usual partition matrix is denoted u_{ik}, while an element of the quantized partition matrix is denoted u_{il}.

Each element of the regularized partition is obtained by

$$u_{ik} = \frac{r_{ik}}{\sum_{j=1}^{c} r_{jk}} \tag{3.26}$$

where r is a spatial function. We define two general form of r :

- $r_{ik} = \sum_{j \in \mathcal{N}(\mathbf{x}_k)} u_{ij}$,
- $r_{ik} = median_{j \in \mathcal{N}(\mathbf{x}_k)} u_{ij}$,

which respectively correspond to a mean and a median filter applied on the membership degrees of the neighborhood of the pixel. However, in order to keep the algorithm fast, this regularization cannot be done in each updating step, so that U is smoothed when the local optima has been reached.

Algorithm 2 Color segmentation algorithm

procedure SEGMENTATION(Image I, No. of clusters c, No. of bins q)
  Pre-process the image I using (3.20).
  initialize prototype matrix V using the Ordering-split procedure (Algorithm 1).
  repeat
    Update partition matrix U using (3.24).
    Update prototypes matrix V using (3.25).
  until $\|V - V_{old}\| < \epsilon$ ▷ $\|.\|$ is a matrix norm, e.g. Frobenius.
  Regularize the partition U using (3.26).
  return (U, V) ▷ Partition and prototypes matrices returned.
end procedure

3.4 KERNEL BASED OBJECTIVE FUNCTIONS

Kernel-induced distances try to embed the data into a Hilbert space, and to search linear separations into this space. Hilbert spaces can be viewed as a generalization of Euclidean spaces with a dimension that is possibly infinite. Formally, having a mapping $\Phi : X \to F$, where F is an embedding space, each element of the *kernel matrix* $\mathbf{K}_{ij}$ is defined by the *kernel function* $\kappa(\mathbf{x}_i, \mathbf{x}_j)$, which is equal to the inner product in the Hilbert space $\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$. The kernel trick allows to not explicitly map the vectors with Φ, which would lead to high computational effort. In fact, calculating the inner product in the Hilbert space is the same as calculating the kernel function. Among the most used kernel functions, we have

- linear kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle + c$,
- polynomial kernel of degree d $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\alpha \langle \mathbf{x}_i, \mathbf{x}_j \rangle + c)^d$,
- Gaussian kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$,
- sigmoid kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\alpha \langle \mathbf{x}_i, \mathbf{x}_j \rangle + c)$.

Moreover, an infinite number of kernels can be defined, since various combinations of existing ones may provide a new kernel function (see for instance []).

The most natural way to introduce kernel-induced distances into iterative clustering algorithms is to replace the Euclidean squared norm $\|\mathbf{x}_k - \mathbf{v}_i\|^2$ by the squared norm in the Hilbert space $\|\Phi(\mathbf{x}_k) - \Phi(\mathbf{v}_i)\|^2$. In other terms, each feature vector is mapped into a possibly infinite dimension space, where we try to find a linear separation between samples. We have

$$\begin{aligned} \|\Phi(\mathbf{x}_k) - \Phi(\mathbf{v}_i)\|^2 &= \langle \Phi(\mathbf{x}_k) - \Phi(\mathbf{v}_i), \Phi(\mathbf{x}_k) - \Phi(\mathbf{v}_i) \rangle \\ &= \langle \Phi(\mathbf{x}_k), \Phi(\mathbf{x}_k) \rangle - 2 \langle \Phi(\mathbf{x}_k), \Phi(\mathbf{v}_i) \rangle \\ &\quad + \langle \Phi(\mathbf{v}_i), \Phi(\mathbf{v}_i) \rangle \end{aligned} \tag{3.27}$$

Assuming that we use a radial basis function (i.e. $\kappa(\mathbf{x}_k, \mathbf{x}_k) = 1$), we finally obtain $\|\Phi(\mathbf{x}_k) - \Phi(\mathbf{v}_i)\|^2 = 2 - 2k(\mathbf{x}_k, \mathbf{v}_i)$.

With this formulation, the kernel based fuzzy c-means objective function can be written as

$$J_K(U, V) = 2 \sum_{k=1}^{n} \sum_{i=1}^{c} u_{ik}^m \Big(1 - \kappa(\mathbf{x}_k, \mathbf{v}_i)\Big),$$

which gives, by zeroing the gradient with respect to u_{ik} and $\mathbf{v}_i$,

$$u_{ik} = \frac{\big(1-\kappa(\mathbf{x}_k,\mathbf{v}_i)\big)^{-\frac{1}{m-1}}}{\sum_{j=1}^{c}\big(1-\kappa(\mathbf{x}_k,\mathbf{v}_j)\big)^{-\frac{1}{m-1}}} \tag{3.28}$$

and

$$\mathbf{v}_i = \frac{\sum_{k=1}^{n} u_{ik}^m \kappa(\mathbf{x}_k,\mathbf{v}_i)\mathbf{x}_k}{\sum_{k=1}^{n} u_{ik}^m \kappa(\mathbf{x}_k,\mathbf{v}_i)} \tag{3.29}$$

The equations obtained are robust to noise and outliers according to Huber's robust statistics, (see []). Moreover, when updating the cluster centers matrix, we note that the term $\kappa(\mathbf{x}_k,\mathbf{v}_i)$, which measures the similarity between $\mathbf{x}_k$ and $\mathbf{v}_i$, serves as a weight. When this similarity is low, the point $\mathbf{x}_k$ is considered as an outlier from the perspective of the center $\mathbf{v}_i$. Therefore, the contribution of $\mathbf{x}_k$ when computing $\mathbf{v}_i$ is very small. Inversely, when the similarity is large, the contribution is large.

The use of kernel-based fuzzy c-means in image segmentation is now a well studied problem. However, it still suffers of several drawbacks. The vast majority of the proposed algorithms restrict to gray level images, and the computation time is extremely high for large images.

3.5 THE FAST KERNEL-BASED FUZZY C-MEANS ALGORITHM

In this section, we give the new objective function, and derive the updating equations of membership and center matrices. Then, we describe the step by step procedure to segment a given image. We now propose the Quantized Kernel Fuzzy c-means (QKFCM) clustering algorithm. The factor 2 is omitted because it does not change the updating equations.

$$J_{QK}(U,V) = \sum_{i=1}^{c}\sum_{l=1}^{q^3} u_{il}^m \Big(1-\kappa(\mathbf{x}_l,\mathbf{v}_i)\Big),$$

Using Lagrange multipliers, we respectively obtain

$$u_{il} = \frac{(1-\kappa(\mathbf{x}_l,\mathbf{v}_i))^{-1/(m-1)}}{\sum_{j=1}^{c}(1-\kappa(\mathbf{x}_l,\mathbf{v}_j))^{-1/(m-1)}} \tag{3.30}$$

and

$$\mathbf{v}_i = \frac{\sum_{l=1}^{q^3} h_l u_{il}^m \kappa(\mathbf{x}_l,\mathbf{v}_i)\mathbf{x}_l}{\sum_{l=1}^{q^3} h_l u_{il}^m \kappa(\mathbf{x}_l,\mathbf{v}_i)} \tag{3.31}$$

for the update of the partition matrix and the centroids. The corresponding clustering procedure is described in Algorithm 3. Using the mean operator in (3.20) and (3.26) gives the so called QKFCM_S1 algorithm, while using the median gives the QKFCM_S2 algorithm. The kernel induced-distance used in the algorithm needs one parameter, namely σ. A commonly used value is obtained by computing the sample variance of the data :

$$\sigma^2 = \frac{1}{n}\sum_{k=1}^{n} \|\mathbf{x}_k - \overline{\mathbf{x}}\|^2 \tag{3.32}$$

Algorithm 3 Kernel color segmentation algorithm

procedure SEGMENTATION(Image I, No. of clusters c, No. of bins q)
    Pre-process the image I using (3.20).
    initialize prototype matrix V using the Ordering-split procedure (Algorithm 1).
    repeat
        Update partition matrix U using (3.30).
        Update prototypes matrix V using (3.31).
    until $\|V - V_{old}\| < \epsilon$ ▷ $\|.\|$ is a matrix norm, e.g. Frobenius.
    Regularize the partition U using (3.26).
    return (U, V) ▷ Partition and prototypes matrices returned.
end procedure

where $\overline{\mathbf{x}}$ is the mean vector of the data set given by

$$\overline{\mathbf{x}} = \frac{1}{n} \sum_{k=1}^{n} \mathbf{x}_k \tag{3.33}$$

[1] E. H. Ruspini, A new approach to clustering, Information and control 15 (1) (1969) 22–32.

[2] J. C. Dunn, A fuzzy relative isodata process and its use in detecting well-separated clusters, Journal of Cybernetics 3 (3) (1974) 32–57.

[3] J. C. Bezdek, Pattern Recognition with fuzzy objective function algorithm, Plenum Press, 1981.

[4] A. Jain, M. Murty, P. Flynn, Data clustering : A review, ACM Computing Surveys 31 (1999) 264–323.

[5] S. K. Khan, A. Ahmad, Cluster center initialization algorithm for k-means clustering, Pattern Recognition Letters 25 (2004) 1293–1302.

[6] R. A. Fisher, The use of multiple measurements in taxonomic problems, Annals of Eugenics 7 (1936) 179–188.

[7] M. N. Ahmed, S. M. Yamany, N. Mohamed, A. A. F. ant T. Moriarty, A modified fuzzy c-means algorithm for bias field estimation and segmentation of mri data, IEEE Transactions on Medical Imaging 21 (3) (2002) 193–199.

[8] A. Liew, S. H. Leung, W. H. Lau, Segmentation of color lip images by spatial fuzzy clustering, IEEE Transactions on Fuzzy Systems 11 (4) (2003) 542–549.

[9] S. Chen, D. Zhang, Robust image segmentation using fcm with spatial constraints based on new kernel-induced distance measure, IEEE Transactions on Systems, Man and Cybernetics, Part B. 34 (4) (2004) 1907–1916.

[10] M.-S. Yang, H.-S. Tsai, A gaussian kernel-based fuzzy c-means algorithm with a spatial bias correction, Pattern Recognition Letters 29 (12) (2008) 1713–1725.

[11] S. Krinidis, V. Chatzis, A robust fuzzy local information c-means clustering algorithm, IEEE Transactions on Image Processing 19 (2010) 1328–1337.

[12] S. Eschrich, J. KE, L. O. Hall, D. B. Goldgof, Fast accurate fuzzy clustering through data reduction, IEEE Transactions on Fuzzy Systems 11 (2) (2003) 262–270.

[13] L. Szilagyi, Z. Benyo, S. Szilagyi, H. Adam, Mr brain image segmentation using an enhanced fuzzy c-means algorithm, in : Proc. of 25th Annual International Conference of IEEE EMBS, 2003, pp. 724–726.

[14] W. Cai, S. Chen, D. Zhang, Fast and robust fuzzy c-means clustering algorithms incorporating local information for image segmentation, Pattern Recognition 40 (2007) 825–838.

[15] C. Tomasi, R. Manduchi, Bilateral filtering for gray and color images, in : International Conference on Computer Vision, 1998, pp. 839–.

[16] J. Angulo, J. Serra, Color segmentation by ordered mergings, in : IEEE International Conference on Image Processing, 2003, pp. 125–128.