Fuzzy Shell Cluster Analysis

F.Klawonn, R.Kruse and H.Timm
University of Magdeburg, Magdeburg, Germany

Abstract

In this paper we survey the main approaches to fuzzy shell cluster analysis which is simply a generalization of fuzzy cluster analysis to shell like clusters, i.e. clusters that lie in nonlinear subspaces. Therefore we introduce the main principles of fuzzy cluster analysis first. In the following we present some fuzzy shell clustering algorithms. In many applications it is necessary to determine the number of clusters as well as the classification of the data set. Subsequently therefore we review the main ideas of unsupervised fuzzy shell cluster analysis. Finally we present an application of unsupervised fuzzy shell cluster analysis in computer vision.

1 Introduction

Cluster analysis is a technique for classifying data, i.e. to divide the given data into a set of classes or *clusters*. In classical cluster analysis each datum has to be assigned to exactly one class. Fuzzy cluster analysis relaxes this requirement by allowing gradual memberships, offering the opportunity to deal with data that belong to more than one class at the same time.

Traditionally, fuzzy clustering algorithms were used to search for compact clusters. Another approach is to search for clusters that represent nonlinear subspaces, for instance spheres or ellipsoids. This is done using fuzzy shell clustering algorithms, which is the subject of this paper.

Fuzzy shell cluster analysis is based on fuzzy cluster analysis. Therefore we review the main ideas of fuzzy cluster analysis first, and present then some fuzzy shell clustering algorithms. These algorithms search for clusters of different shapes, for instance ellipses, quadrics, ellipsoids etc. Since in many applications the number of clusters, into which the data shall be divided, is not known in advance, subsequently the subject of unsupervised fuzzy shell clustering analysis is reviewed. Unsupervised fuzzy shell clustering algorithms determine the number of clusters as well as the classification of the data set. Finally an application of fuzzy shell cluster analysis in computer vision is presented.

2 Fuzzy Cluster Analysis

2.1 Objective Function Based Clustering

Objective function based clustering methods determine an optimal classification of data by minimizing an objective function. Depending on whether binary or gradual memberships are used, one distinguishes between hard and fuzzy clustering methods. In fuzzy cluster analysis data can belong to several clusters at different degrees and not only to one. In general the performance of fuzzy clustering algorithms is superior to that of the corresponding hard algorithms [1].

In objective function based clustering algorithms each cluster is usually represented by a prototype. Hence the problem of dividing a data set X, $X = \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^p$, into c clusters can be stated as the task of minimizing the distances of the datum to the prototypes. This is done by minimizing the following objective function $J(X, U, \beta)$

$$J(X, U, \beta) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^m d^2(\beta_i, x_j) \tag{1}$$

subject to

$$\textstyle\sum_{j=1}^{n} u_{ij} > 0 \quad \text{for all } i \in \{1, \ldots, c\} \tag{2}$$
$$\textstyle\sum_{i=1}^{c} u_{ij} = 1 \quad \text{for all } j \in \{1, \ldots, n\} \tag{3}$$

where $u_{ij} \in [0, 1]$ is the membership degree of datum x_j to cluster i, β_i is the prototype of cluster i, and $d(\beta_i, x_j)$ is the distance between datum x_j and prototype β_i. The $c \times n$ matrix $U = [u_{ij}]$ is also called the fuzzy partition matrix and the parameter m is called the fuzzifier. Usually $m = 2$ is chosen.

Constraint (2) guarantees that no cluster is empty and constraint (3) ensures that the sum of membership degrees for each datum equals 1. Fuzzy clustering algorithms which satisfy these constraints are also called *probabilistic clustering algorithms*, since the membership degrees for one datum formally resemble the probabilities of its being a member of the corresponding cluster.

The objective function $J(X, U, \beta)$ is usually minimized by updating the membership degrees u_{ij} and the prototypes β_i in an alternating fashion, until the change ΔU of the membership degrees is less than a given tolerance ε. This approach is also known as the alternating optimization method.

A Fuzzy Clustering Algorithm
Fix the number of clusters c
Fix m, $m \in (1, \infty)$
Initialize the fuzzy c-partition U
REPEAT
    Update the parameters of each clusters prototype

Update the fuzzy c-partition U using (4)
UNTIL $|\Delta U| < \varepsilon$

To minimize the objective function (1), the membership degrees are updated using (4). The following equation for updating the membership degrees can be derived by differentiating the objective function (1).

$$u_{ij} = \begin{cases} \dfrac{1}{\sum_{k=1}^{c} \left(\dfrac{d^2(x_j, \beta_i)}{d^2(x_j, \beta_k)} \right)^{\frac{1}{m-1}}} & \text{if } I_j = \emptyset, \\ 0 & \text{if } I_j \neq \emptyset \text{ and } i \notin I_j, \\ x, x \in [0,1] \text{ such that } \sum_{i \in I_j} u_{ij} = 1, & \text{if } I_j \neq \emptyset \text{ and } i \in I_j. \end{cases} \tag{4}$$

This equation is used for updating the membership degrees in every probabilistic clustering algorithm.

In contrast to the minimization of the objective function (1) the minimization of (1) varies with respect to the prototypes according to the choice of the prototypes and the distance measure. Therefore each choice leads to a different algorithm.

2.2 Possibilistic Clustering Algorithms

The prototypes are not always determined correctly using probabilistic clustering algorithms, i.e. only a suboptimal solution is found. The main source of the problem is constraint (3), which requires the membership degrees of a point across all clusters to sum up to 1. This is easily demonstrated by considering the case of two clusters. A datum x_1, which is typical for both clusters, has the same membership degrees as a datum x_2, which is not at all typical for any of them. For both data the membership degrees are $u_{ij} = 0.5$ for $i = 1, 2$. Therefore both data influence the updating of the clusters to the same extent.

An obvious modification is to drop constraint (3). To avoid the trivial solution, i.e. $u_{ij} = 0$ for all $i \in \{1, \ldots, c\}, j \in \{1, \ldots, n\}$, (1) is modified to (5).

$$J(X, U, \beta) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^m d^2(\beta_i, x_j) + \sum_{i=1}^{c} \eta_i \sum_{j=1}^{n} (1 - u_{ij})^m \tag{5}$$

where $\eta_i > 0$.

The first term minimizes the weighted distances while the second term avoids the trivial solution. A fuzzy clustering algorithm that minimizes the objective function (5) under the constraint (2) is called a *possibilistic clustering algorithm*, since the membership degrees for one datum resemble the possibility of its being a member of the corresponding cluster.

Minimizing the objective function (5) with respect to the membership degrees leads to the following equation for updating the membership degrees u_{ij} [11].

$$u_{ij} = \frac{1}{1 + \left(\frac{d^2(x_j, \beta_i)}{\eta_i} \right)^{\frac{1}{m-1}}} \tag{6}$$

Equation (6) shows, that η_i determines the distance at which the membership degree equals 0.5. If $d^2(x_j, \beta_i)$ equals η_i, the membership degree equals 0.5. So it is useful, to choose η_i for each cluster separately [11]. η_i can be determined by using the fuzzy intra cluster distance (7) for example.

$$\eta_i = \frac{K}{N_i} \sum_{j=1}^{n} (u_{ij})^m d^2(x_j, \beta_i). \tag{7}$$

where $N_i = \sum_{j=1}^{n} (u_{ij})^m$. Usually $K = 1$ is chosen.

It is recommended to initialize a possibilistic clustering algorithm with the results of the corresponding probabilistic version [12]. In case prior information about the clusters is available, it can be used to determine η_i for a further iteration of the fuzzy clustering algorithm to fine tune the results [10].

A Possibilistic Clustering Algorithm
Fix the number of clusters c
Fix m, $m \in (1, \infty)$ Initialize U using the corresponding fuzzy algorithm
Compute η_i using (7)
REPEAT
    Update prototype using U
    Compute U using (6)
UNTIL $|\Delta U| < \varepsilon_1$
Fix the values of η_i using a priori information
REPEAT
    Update prototype using U
    Compute U using (6)
UNTIL $|\Delta U| < \varepsilon_2$
} *optional*

2.3 The Fuzzy C Means Algorithm

The simplest fuzzy clustering algorithm is the *fuzzy c means algorithm (FCM)* [1]. The c in the name of the algorithm reminds that the data is divided into c clusters. The FCM searches for compact clusters which have approximately the same size and shape. Therefore the prototype is a single point which is the center of the cluster, i.e. $\beta_i = (c_i)$. The size and shape of the clusters are determined by a positive definite $n \times n$ matrix A. Using this matrix A the distance of a point x_j to the prototype β_i is given by

$$d^2(x_j, \beta_i) = \|x_j - c_i\|_A^2 = (x_j - c_i)^T A (x_j - c_i). \tag{8}$$

In case A is the identity matrix, the FCM looks for spherical clusters otherwise for ellipsoidal ones. In most cases the Euclidean norm is used, i.e. A is the identity matrix. Hence the distance reduces to the Euclidean norm, i.e.

$$d^2(x_j, \beta_i) = \|x_j - c_i\|^2. \tag{9}$$

Minimizing the objective function with respect to the prototypes leads to the following equation (10) for updating the prototypes [7].

$$c_i = \frac{1}{N_i} \sum_{j=1}^{n} (u_{ij})^m x_j \tag{10}$$

where $N_i = \sum_{j=1}^{n} (u_{ij})^m$.

A disadvantage of the FCM is, that A is not updated. Therefore the shape of the clusters cannot be changed. Besides, when the clusters are of different shape, it is not appropriate to use a single matrix A for all clusters at the same time.

2.4 The Gustafson-Kessel Algorithm

The *Gustafson-Kessel algorithm (GK)* searches for ellipsoidal clusters [6]. In contrast to the FCM, a separate matrix A_i, $A_i = (\det C_i)^{1/n} C_i^{-1}$, is used for each cluster. The norm matrices are updated as well as the centers of the corresponding clusters. Therefore the prototypes of the clusters are a pair (c_i, C_i), where c_i is the center of the cluster and C_i the covariance matrix, which defines the shape of the cluster.

Like the FCM the GK computes the distance to the prototypes by

$$d^2(x_j, \beta_i) = (\det C_i)^{1/n} (x_j - c_i)^T C_i^{-1} (x_j - c_i). \tag{11}$$

To minimize the objective function with respect to the prototypes, the prototypes are updated according to the following equations [7]

$$c_i = \frac{1}{N_i} \sum_{j=1}^{n} (u_{ij})^m x_j, \tag{12}$$

$$C_i = \frac{1}{N_i} \sum_{j=1}^{n} (u_{ij})^m (x_j - c_i)(x_j - c_i)^T. \tag{13}$$

The GK is a simple fuzzy clustering algorithm to detect ellipsoidal clusters with approximately the same size but different shapes. In combination with the FCM it is often used to initialize other fuzzy clustering algorithms. Besides the GK can also be used to detect linear clusters. This is possible, because lines and planes can also be seen as degenerated ellipses or ellipsoids, i.e. at least in one dimension the radius nearly equals zero.

2.5 Other Algorithms

There are many fuzzy clustering algorithms besides the FCM and the GK. These algorithms search for clusters with different shape, size and density of data and use different distance measures. For example, if one is interested in ellipsoidal clusters of varying size the Gath and Geva algorithm can be used [5]. It searches for ellipsoidal clusters, which can have different shape, size, and density of data.

If one is interested in linear clusters, for instance lines, linear clustering algorithms, for example the fuzzy c-varieties algorithm [1] or the adaptive fuzzy clustering algorithm [3], can be used. Another linear clustering algorithm is the compatible cluster merging algorithm (CCM) [8, 7]. This algorithm uses the property of the GK to detect linear clusters and improves the results obtained by the GK by merging compatible clusters. Two clusters are considered compatible, if the distance between these clusters is small compared to their size and if they lie in the same hyperplane.

A common application of the CCM is line detection. The advantage of the CCM in comparison to other line detection algorithms is its ability to detect significant structures while neglecting insignificant ones.

3 Fuzzy Shell Cluster Analysis

The fuzzy clustering algorithms discussed up to now search for clusters that lie in linear subspaces. Besides, it is also possible to detect clusters that lie in nonlinear subspaces, i.e. resemble shells or patches of surfaces with no interior points. These clusters can be detected using fuzzy shell clustering algorithms.

The only difference between fuzzy clustering algorithms and fuzzy shell clustering algorithms is that the prototypes of fuzzy shell clustering algorithms resemble curves resp. surfaces or hypersurfaces. Therefore the algorithm for probabilistic clustering and the algorithm for possibilistic clustering are both used for fuzzy shell cluster analysis.

There is a large number of fuzzy shell clustering algorithms which use different kinds of prototypes and different distance measures. Fuzzy shell clustering algorithms can detect ellipses, quadrics, polygons, ellipsoids, hyperquadrics etc. In the following the fuzzy c ellipsoidal shells algorithm, which searches for ellipsoidal clusters, and the fuzzy c quadric shells algorithm, which searches for quadrics, are presented. Further fuzzy shell clustering algorithms are described in [7].

3.1 The Fuzzy C Ellipsoidal Shells Algorithm

The *fuzzy c ellipsoidal shells algorithm (FCES)* searches for shell clusters with the shape of ellipses, ellipsoids or hyperellipsoids [7, 4]. In the following we present the algorithm to find ellipses.

An ellipse is given by

$$(x - c_i)^T A_i (x - c_i) = 1, \tag{14}$$

where c_i is the center of the ellipse and A_i is a positive symmetric matrix, which determines the major and minor axes lengths as well as the orientation of the ellipse. From that description of an ellipse the prototypes β_i, $\beta_i = (c_i, A_i)$, for the clusters are derived.

The fuzzy c ellipsoidal shells algorithm uses the radial distance. This distance measure is a good approximation to the exact (perpendicular) distance, but easier to compute. The radial distance d^2_{Rij} of a point x_j to a prototype β_i is given by

$$d^2(x_j, \beta_i) = d^2_{Rij} = \|x_j - z\|^2, \tag{15}$$

where z is the point of the intersection of the ellipse β_i and the line through c_i and x_j that is near to the cluster.

Using (14) d^2_{Rij} can be transformed to

$$d^2_{Rij} = \frac{(\sqrt{(x_j - c_i)^T A_i (x_j - c_i)} - 1)^2 \|x_j - c_i\|^2}{(x_j - c_i)^T A_i (x_j - c_i)}. \tag{16}$$

Minimizing the objective function with respect to the prototypes leads to the following system of equations [7]:

$$\sum_{j=1}^{n} u_{ij}^m (x_j - c_i)(x_j - c_i)^\top \left(\frac{||x_j - c_i||}{d_{ij}}\right)^2 \left(\sqrt{d_{ij}} - 1\right) \;=\; 0, \tag{17}$$

$$\sum_{j=1}^{n} \frac{u_{ij}^m \left(\sqrt{d_{ij}} - 1\right)}{d_{ij}^2} \cdot \left[||x_j - c_i||^2 A_i + \left(\sqrt{d_{ij}} - 1\right) d_{ij} I\right] (x_j - c_i) \;=\; 0, \tag{18}$$

where $d^2_{ij} = (x_j - c_i)^T A_i (x_j - c_i)$ and I is the identity matrix.

This system of equations has to be solved using numerical techniques. To update the prototypes e.g. the Levenberg-Marquardt algorithm [13] can be used.

3.2 The Fuzzy C Quadric Shells Algorithm

The fuzzy c quadric shells algorithm ($FCQS$) searches for clusters with the shape of a quadric or a hyperquadric. A quadric resp. a hyperquadric is defined by

$$p_i^T q = 0, \tag{19}$$

where
$p_i^T = (p_{i1}, p_{i2}, \ldots, p_{in}, p_{i(n+1)}, \ldots, p_{ir}, p_{ir+1}, \ldots, p_{is})$,
$q^T = (x_1^2, x_2^2, \ldots, x_n^2, x_1 x_2, \ldots, x_{n-1} x_n, \ldots, x_1, x_2, \ldots, x_n, 1)$,
$s = n(n+1)/2 + n + 1 = r + n + 1$,
n is the dimension of the feature vector of a datum and $r = n(n+1)/2$.
Hence the prototypes of the fuzzy c quadric shell clustering algorithm are s-tuples.

The FCQS uses the algebraic distance. The algebraic distance of a point x_j to a prototype β_i is defined by

$$d^2(x_j, \beta_i) = d^2_{Qij} = p_i^T q_j q_j^T p_i = p_i^T M_j p_i, \tag{20}$$

where $M_j = q_j q_j^T$.

An additional constraint is needed to avoid the trivial solution $p_i^T = (0, \ldots, 0)$. For two dimensional data the constraint

$$\|p_{i1}^2 + p_{i2}^2 + \ldots + p_{in}^2 + \frac{1}{2}p_{i(n+1)}^2 + \ldots + \frac{1}{2}p_{ir}^2\|^2 = 1 \tag{21}$$

is recommended, because it is a good compromise between performance and result quality [10]. However this constraint prevents the algorithm from finding linear clusters. Linear clusters are detected as hyperbolas or ellipses with a large ratio of major to minor axis. Therefore an additional algorithm for line detection is needed, which is executed after the FCQS. For that purpose the CCM is well suited. Good results are obtained by initializing the CCM, using those clusters, which probably represent linear clusters, i.e. hyperbolas and ellipses with a large ratio of major to minor axis [10].

Defining $a_i = (a_{i1}, \ldots, a_{in}), b_i = (b_{i1}, \ldots, b_{in})$ by

$$a_{ik} = \begin{cases} p_{ik} & 1 \le k \le n \\ \dfrac{p_{ik}}{\sqrt{2}} & n+1 \le k \le r \end{cases} \tag{22}$$

$$b_{ik} = \; p_{i(r+k)} \quad 1 \le k \le s - r \tag{23}$$

constraint (21) simplifies to $\|a_i\|^2 = 1$. To minimize the objective function with respect to the prototypes, a_i and b_i are computed by
a_i = eigenvector corresponding to the smallest eigenvalue of $(F_i - G_i^T H_i^{-1} G_i)$,
$b_i = -H_i^{-1} G_i a_i$,
where

$$F_i = \sum_{j=1}^{n} u_{ij}^m R_j, \quad G_i = \sum_{j=1}^{n} u_{ij}^m S_j, \quad H_i = \sum_{j=1}^{n} u_{ij}^m T_j,$$
$$R_j = r_j r_j^T, \qquad S_j = r_j t_j^T, \qquad T_j = t_j t_j^T,$$

$r_j^T = [x_{j1}^2, x_{j2}^2, \ldots, x_{jn}^2, \sqrt{2}x_{j1}x_{j2}, \ldots, \sqrt{2}x_{jk}x_{jl}, \ldots, \sqrt{2}x_{jn-1}x_{jn}]$,
$t_j^T = [x_{j1}, x_{j2}, \ldots, x_{jn}, 1]$.
Therefore updating the prototypes reduces to an eigenvector problem of size $n(n+1)/2$, which is trivial. However the chosen distance measure d^2_{Qij} is highly nonlinear in nature and is sensitive to the position of a datum x_j with respect to the prototype β_i [10]. Therefore the membership degrees computed using the algebraic distance are not very meaningful. Depending on the data, this sometimes leads to bad results.

Since this problem of the FCQS is caused by the particular distance measure, the modified FCQS uses the shortest (perpendicular) distance d_{Pij}^2. To compute this distance, we first rewrite (19) as $x^T A_i x + x^T b_i + c_i = 0$. Then the shortest distance between a datum x_j and a cluster β_i is given by [10]

$$d^2(x_j, \beta_i) = d_{Pij}^2 = \min_z \|x_j - z\|^2, \tag{24}$$

subject to

$$z^T A_i z + z^T b_i + c_i = 0, \tag{25}$$

where z is a point on the quadric β_i. By using the Lagrange multiplier λ, the solution is found to be

$$z = \frac{1}{2}(I - \lambda A_i)^{-1}(\lambda b_i + 2x_j), \tag{26}$$

where I is the identity matrix. Substituting (26) in (25) yields a forth degree equation in λ. Each real root λ_k of this polynomial represents a possible value for λ. Calculating the corresponding z vector z_k, d_{Pij}^2 is determined by

$$d_{Pij}^2 = \min_k \|x_j - z_k\|^2. \tag{27}$$

The disadvantage of using the exact distance is, that the modified FCQS is computationally very expensive, because updating the prototypes can be achieved only by numeric techniques such as the Levenberg-Marquardt algorithm [13, 10, 4]. Therefore using a simplified modified FCQS is recommended. In this simplified algorithm the prototypes are updated using the algebraic distance d_{Qij} and the membership degrees are updated using the shortest distance d_{Pij} [10].

In higher dimensions the approximate distance d_{Aij} is used instead of the geometric distance d_{Pij}. It is defined by:

$$d^2(x_j, \beta_i) = d_{Aij}^2 = \frac{d_{Qij}^2}{|\nabla d_{Qij}|^2} = \frac{p_i^T M_j p_i}{p_i^T (D(q_j) D(q_j)^T) p_i} \tag{28}$$

where $\bigtriangledown d_{Qij}$ is the gradient of the functional $p_i^T q$ evaluated in x_j and $D(q_j)$ the Jacobian of q evaluated in x_j. The corresponding variant of the FCQS is called the fuzzy c plano-quadric shells algorithm (FCPQS) [10].

The reason for using the approximate distance is that there is no closed form solution for d_{Pij} in higher dimensions. Hence in higher dimensions the modified FCQS cannot be applied.

Updating the prototypes of the FCPQS requires solving a generalized eigenvector problem, for instance on the basis of the QZ algorithm [10].

4 Unsupervised Fuzzy Shell Cluster Analysis

The algorithms discussed so far are based on the assumption that the number of clusters is known beforehand. However, in many applications the number of clusters c into which a data set shall be divided is not known.

This problem can be solved using unsupervised fuzzy clustering algorithms. These algorithms determine automatically the number of clusters by evaluating a computed classification on the basis of validity measures.

There are two kinds of validity measures, local and global. The former evaluates single clusters while the latter evaluates the whole classification. Depending on the validity measure, unsupervised fuzzy clustering algorithms are divided into algorithms based on local validity measures and algorithms based on global validity measures.

In this section the ideas of unsupervised fuzzy clustering are presented. A detailed discussion can be found in [7].

4.1 Global Validity Measures

An unsupervised fuzzy clustering algorithm based on a global validity measure is executed several times, each time with a different number of clusters. After each execution the clustering of the data set is evaluated. Since global validity measures evaluate the clustering of a data set as a whole, only a single value is computed. Usually the number of clusters is increased until the evaluation of the clustering indicates that the solution becomes worse.

However it is very difficult to detect a probably optimal solution as is easily demonstrated. A very simple global validity measure is the objective function of the fuzzy clustering algorithm. But it is obvious that the global minimum of that validity measure is unusable, because the global minimum is reached, if the number of data equals the number of clusters. Therefore often the apex of the validity function is used instead.

Unfortunately it is possible that the classification as a whole is evaluated as good, although no cluster is recognized correctly.

Some validity measures use the fuzziness of the membership degrees. They are based on the idea that a good solution of a fuzzy clustering algorithm is characterized by a low uncertainty with respect to the classification. Hence the algorithms based on these measures search for a partition which minimizes the classification uncertainty. For example this is done using the *partition coefficient* [1].

Other validity measures are more related to the geometry of the data set. For example the *fuzzy hypervolume* is based on the size of the clusters [5]. Because in probabilistic clustering each datum is assigned to a cluster, a low value of this measure indicates small clusters which just enclose the data.

For fuzzy shell clustering algorithms other validity measures are used. For example the *fuzzy shell thickness* measures the distance between the data and the corresponding clusters [10].

4.2 Local validity measures

In contrast to global validity measures, local validity measures evaluate each cluster separately. Therefore it is possible to detect some good clusters even if the classification as a whole is bad.

An unsupervised fuzzy clustering algorithm based on local validity measures starts with a number of clusters greater than the expected number of clusters. A good recommendation is to use twice as many clusters as expected [10]. After each execution of the used fuzzy clustering algorithm the number of clusters and the data is reduced. Good clusters and the assigned data are temporarily removed. Bad clusters and data that represent noise are deleted. In a further iteration the fuzzy clustering algorithm is executed on the remaining data using the number of the remaining clusters. This is repeated until no cluster is left or no cluster is removed. After the number of clusters is determined, the temporarily removed clusters are fine tuned by running the clustering algorithm again.

Some local validity measures are derived from global validity measures. For example the fuzzy shell thickness can also be determined for each cluster separately [10]. However, sometimes the evaluation of a clusters varies depending on its size and completeness. For example it is difficult to distinguish between sparse clusters and cluster segments using these validity measures.

For two dimensional data good results are obtained by using the *surface density* [10]. The surface density measures the relation between the number of data assigned to a cluster and the number of data, if that cluster would be perfect. For example, for a full circle the number of assigned data is related to the circumference. It is also possible to distinguish between cluster segments and sparse clusters using the surface density.

For applications in computer vision it has been proven successful to use the surface density in combination with other local validity measures.

4.3 Initialization

The performance and quality of a classification computed by a fuzzy clustering algorithm depends to a high degree on the initialization of the fuzzy clustering algorithm. Especially fuzzy shell clustering algorithms are very sensitive concerning the initialization, because they tend to get stuck in local minima.

A widely used procedure for initialization is to start with some iterations of the FCM and, if applicable, the GK and the FCES. For instance, a good initialization of the FCQS and its modifications can be achieved by using 10 iterations of the FCM, 10 iterations of the GK and 5 iterations of the FCES [10]. In case only ellipsoidal shell clusters are searched, better results are achieved by omitting the GK because it sometimes tends to search for lines.

An alternative approach is to apply methods from computer vision. For example, some techniques from boundary detection can be used to initialize fuzzy shell clustering algorithms. One such algorithm is introduced in [7]. For shell cluster analysis it is superior to the initialization procedure using the FCM, the GK, and the FCES described above. The reason is that the initialization relates more to the nature of the searched clusters. In addition this algorithm estimates the expected number of

clusters at the beginning of the fuzzy shell clustering algorithm. Hence it can reduce the computation time of the unsupervised fuzzy shell clustering algorithm. The results presented in section 5 are computed using this algorithm.

5 Fuzzy Shell Cluster Analysis in Computer Vision

Fuzzy clustering techniques can be applied in numerous fields. One of them is computer vision, in which clustering methods have been used for region segmentation for years. Another application in computer vision is contour detection and fitting or surface detection and fitting. Unsupervised fuzzy shell clustering algorithms seem to be well suited for this task.

The detection and recognition of boundaries in two dimensional pictures or surfaces in three dimensional scenes is one of the major problems in computer vision. A common method is the use of the generalized Hough transform which is able to deal with noisy and sparse boundaries or surfaces. However the disadvantage of the generalized Hough transform are its computational complexity and its high memory requirements[1] if there are only few assumptions concerning the boundaries or surfaces searched for. An alternative approach is to use fuzzy shell clustering algorithms, which perform boundary detection and fitting or surface detection and fitting simultaneously. These algorithms require far less computations and memory compared with the generalized Hough transform. Besides this algorithms are insensitive to local aberrations and deviations in shape as well as to noise.

Fuzzy shell clustering algorithms searching for quadrics or hyperquadrics can detect a large variety of curves, which they are able to detect. For many applications this is sufficient. It is recommended to use an unsupervised version of the FCQS that is based on the local validity criteria of the surface density [10]. However, the validity measure of surface density has to be slightly modified, because in applications of computer vision the aspect of digitization of images must be considered. That can be done using a correction factor [10].

Finally we present some results of boundary detection and recognition obtained by an unsupervised fuzzy shell clustering algorithm. Fig. 3 and 4 are obtained from fig. 1 and 2 respectively, by using an edge detection algorithm and an algorithm for line thinning. The data shown in these figures are divided into clusters by using an unsupervised fuzzy shell clustering algorithm. Fig. 5 and 6 show the clusters obtained by the FCQS.

It is obvious that the significant boundaries are determined correctly. Besides, like the CCM an unsupervised fuzzy shell clustering algorithm is able to distinguish

[1] The computational complexity is $O(n \times N_{p_1} \times N_{p_2} \ldots \times N_{p_{s-1}})$ and the memory requirement is $O(n \times N_{p_1} \times N_{p_2} \ldots \times N_{p_{s-1}})$, where n is the number of points, N_{p_i} is the number of quantization levels of the i-th parameter, and s is the total number of parameters [10].

between significant and insignificant structures. However, it is always important to bear in mind, that the computed classification is based on the choice of the validity and the distance measures. Therefore an optimal solution computed by an unsupervised fuzzy clustering algorithm sometimes differs from a classification obtained by a human. For example a human might describe the right border of the cup in fig. 4 using an extra line which is missing in fig. 6.

Summarizing unsupervised fuzzy shell cluster analysis is an interesting method for line detection and recognition or contour detection and recognition. Its advantage compared to other algorithms, e.g. the generalized Hough transform, is its lower computational complexity and its ability to distinguish between significant and insignificant structures. However, this ability is also a disadvantage because they cannot detect small and fine structures. Finally it is to remark that an optimal solution cannot be guaranteed.

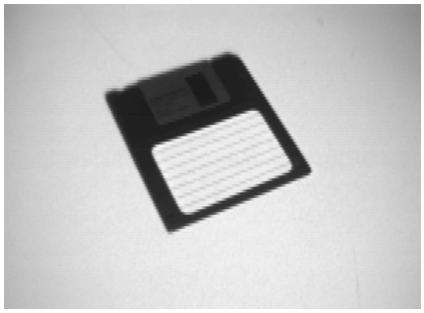

Figure 1: picture of a disk

Figure 2: picture of a cup

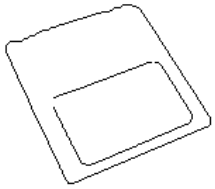

Figure 3: contour of a disk

Figure 4: contour of a cup

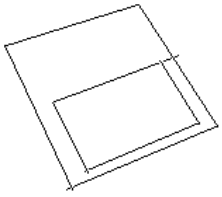

Figure 5: prototypes found by the FCQS

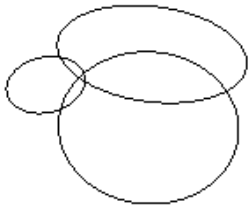

Figure 6: prototypes found by the FCQS

References

[1] Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum, New York 1981.

[2] Bock, H.H.: Classification and Clustering: Problems for the Future, in: New Approaches in Classification and Data Analysis (Ed. Diday, E., Lechevallier, Y., Schrader, M., Bertrand, P. and Burtschy, B.), Springer, Berlin, 1994, 3-24.

[3] Davé, R.N.: Use of the Adaptive Fuzzy Clustering Algorithm to Detect Lines in Digital Images, Proc. Intelligent Robots and Computer Vision VIII, 1192 (1989), 600-611.

[4] Frigui, H. and Krishnapuram, R.: A Comparison of Fuzzy Shell-Clustering Methods for the Detection of Ellipses, IEEE Transactions on Fuzzy Systems, 4 (1996), 193-199.

[5] Gath, I. and Geva, A. B.: Unsupervised Optimal Fuzzy Clustering, IEEE Transactions on Pattern Analysis and Machine Intelligence, 11 (1989), 773-781.

[6] Gustafson, E.E. and Kessel, W.C.: Fuzzy Clustering with a Fuzzy Covariance Matrix, IEEE CDC, San Diego, Californien, 1979, 761-766.

[7] Höppner, F., Klawonn, F. and Kruse, R.: Fuzzy-Clusteranalyse. Verfahren für die Bilderkennung, Klassifikation und Datenanalyse, Vieweg, Braunschweig 1996.

[8] Krishnapuram, R. and Freg, C.P.: Fitting an Unknown Number of Lines and Planes to Image Data through Compatible Cluster Merging, Pattern Recognition, 25 (1992), 385-400.

[9] Krishnapuram, R., Frigui, H. and Nasraoui, O.: The Fuzzy C Quadric Shell clustering algorithm and the detection of second-degree curves, Pattern Recognition Letters 14 (1993), 545-552.

[10] Krishnapuram, R., Frigui, H. and Nasraoui, O.: Fuzzy and Possibilistic Shell Clustering Algorithms and Their Application to Boundary Detection and Surface Approximation — Part 1 & 2, IEEE Transactions on Fuzzy Systems, 3 (1995), 29-60.

[11] Krishnapuram, R. and Keller, J.: A Possibilistic Approach to Clustering, IEEE Transactions on Fuzzy Systems, 1 (1993), pp. 98-110.

[12] Krishnapuram, R. and Keller, J.: Fuzzy and Possibilistic Clustering Methods for Computer Vision, Neural Fuzzy Sytems 12 (1994), 133-159.

[13] Moore, J.J.: The Levenberg-Marquardt Algorithm: Implementation and Theory, in: Numerical Analysis (Ed. Watson, G.A.), Springer, Berlin, 1977, 105-116.