A Fuzzy Clustering Algorithm to Detect Planar and Quadric shapes

Raghu Krishnapuram, Hichem Frigui and Olfa Nasraoui
Department of Electrical and Computer Engineering
University of Missouri
Columbia, MO 65211, USA

ABSTRACT

In this paper, we introduce a new fuzzy clustering algorithm to detect an unknown number of planar and quadric shapes in noisy data. The proposed algorithm is computationally and implementationally simple, and overcomes many of the drawbacks of the existing algorithms that have been proposed for similar tasks. Since the clustering is performed in the original image space, and since no features need to be computed, this approach is particularly suited for sparse data. The algorithm may also be used in pattern recognition applications.

1. Introduction

Boundary detection and surface approximation are important components of intermediate-level vision. They are the first step in solving problems such as object recognition and orientation estimation. Recently, it has been shown that these problems can be viewed as clustering problems with appropriate distance measures and prototypes [1-4]. Dave's Fuzzy *C* Shells (FCS) algorithm [1] and the Fuzzy Adaptive C-Shells (FACS) algorithm [4] have proven to be successful in detecting clusters that can be described by circular arcs, or more generally by elliptical shapes. Unfortunately, these algorithms are computationally rather intensive since they involve the solution of coupled nonlinear equations for the shell (prototype) parameters. These algorithms also assume that the number of clusters are known. To overcome these drawbacks we recently proposed a computationally simpler Fuzzy *C* Spherical Shells (FCSS) algorithm [3] for clustering hyperspherical shells and suggested an efficient algorithm to determine the number of clusters when this is not known. We also proposed the Fuzzy C Quadric Shells (FCQS) algorithm [2] which can detect more general quadric shapes. One problem with the FCQS algorithm is that it uses the algebraic distance, which is highly nonlinear. This results in unsatisfactory performance when the data is not very "clean" [4]. Finally, none of the algorithms can handle situations in which the clusters include lines/planes and there is much noise. To summarize, the existing algorithms to detect quadric shell clusters have one or more of the following drawbacks: i) they are computationally expensive, ii) the distance measure used in the objective function can yield distorted estimates of prototype parameters if the data is not well behaved, iii) they assume that the number of clusters *C* is known, iv) their formulations do not allow the degenerate case of lines/planes, and v) they are not very robust in the presence of noise. In this paper, we address these drawbacks in more detail and propose a new algorithm to overcome these drawbacks.

2. The Fuzzy C Quadrics Algorithm

Let $x_j = [x_{j1}, x_{j2}, \ldots, x_{jn}]$ be a point in the n-dimensional feature space. We may define the algebraic (or residual) distance from x_j to a prototype β_i that resembles a second-degree curve as :

$$d_{Qij}^2 = d_Q^2(x_j,\beta_i) = (p_{i1}x_{j1}^2 + p_{i2}x_{j2}^2 + \ldots + p_{in}x_{jn}^2 + p_{i(n+1)}x_{j1}x_{j2} + p_{i(n+2)}x_{j1}x_{j3} + \ldots$$
$$+p_{is}x_{j(n-1)}x_{jn} + p_{i(s+1)}x_{j1} + p_{i(s+2)}x_{j2} + \ldots + p_{i(s+n)}x_{jn} + p_{i(s+n+1)})^2$$
$$= p_i^T q_j q_j^T p_i = p_i^T M_j p_i \,. \qquad (1)$$

The prototypes β_i are represented by the parameter vectors $p_i = [p_{i1}, p_{i2}, \ldots, p_{ir}]^T$ with $r = s+n+1 = \frac{n(n+1)}{2} + n + 1 = \frac{(n+1)(n+2)}{2}$ components, which define the equation of the curve. The M_j in (1) are given by

$$M_j = q_j q_j^T, \text{ with } q_j^T = [x_{j1}^2, x_{j2}^2, \ldots, x_{jn}^2, x_{j1}x_{j2}, \ldots, x_{j(n-1)}x_{jn}, x_{j1}, x_{j2}, \ldots, x_{jn}, 1]. \qquad (2)$$

We may now minimize the following objective function which is similar to the objective function used in Fuzzy C-Means algorithm (FCM) [6] except for the distance measure.

$$J_Q(B,U) = \sum_{i=1}^{C}\sum_{j=1}^{N}(\mu_{ij})^m d^2_{Qij} = \sum_{i=1}^{C}\sum_{j=1}^{N}(\mu_{ij})^m p_i^T M_j p_i, \quad (3)$$

where $B = (\beta_1,\ldots,\beta_C)$, C is the number of clusters, N is the total number of feature vectors and $U = [\mu_{ij}]$ is the $C\times N$ fuzzy C-partition matrix satisfying the following conditions.

$$\mu_{ij} \in [0,1] \text{ for all } i \text{ and } j, \ \sum_{i=1}^{C}\mu_{ij} = 1 \text{ for all } j, \text{ and } 0 < \sum_{j=1}^{N}\mu_{ij} < N \text{ for all } i. \quad (4)$$

Note, $J_Q(B,U)$ is homogeneous with respect to p_i. Therefore, we need to constrain the problem in order to avoid the trivial solution. Some of the possibilities are:

(i) $p_{i1} = 1$, (ii) $p_{ir} = 1$, (iii) $\|p_i\|^2 = 1$, and

$$\text{(iv) } \|p_{i1}^2 + p_{i2}^2 + \ldots + p_{in}^2 + \frac{p_{i(n+1)}^2}{2} + \frac{p_{i(n+2)}^2}{2} + \ldots + \frac{p_{is}^2}{2}\| = 1. \quad (5)$$

In [4] Dave et al have also proposed a Fuzzy C Quadrics (FCQ) algorithm using constraint (i). This constraint is more restrictive than constraint (iv) used in the FCQS algorithm proposed in [3]. Moreover, the resulting distance measure is not invariant to translations and rotations of the prototypes. Constraints (ii) and (iii) are also not suitable for the same reason. In other words, these constraints make the distance d^2_{Qij} a function of not just the relative location of point x_j to curve β_i, but also the actual location and orientation of the curve β_i in feature space, which is undesirable. However, constraint (iv) makes the distance invariant to translations and rotations [5]. Other constraints are also possible, and one of them will be discussed in Section 4. With constraint (iv) the minimization of (3) reduces to an eigenvector problem, and its implementation is straightforward. Minimization with respect to the memberships μ_{ij} is similar to the FCM case [6]. It is easy to show that the memberships are updated according to

$$\mu_{ij} = \begin{cases} \dfrac{1}{\sum_{k=1}^{C}\left(\dfrac{d_{Q_{ij}}}{d_{Q_{kj}}}\right)^{\frac{2}{m-1}}} & \text{if } I_j = \Phi \\ 0 \quad i \notin I_j & \text{if } I_j \neq \Phi \\ \sum_{i\in I_j}\mu_{ij} = 1 & \text{if } I_j \neq \Phi \end{cases} \quad (6)$$

where $I_j = \{i \mid 1 \le i \le C, d^2_{Qij} = 0\}$. The original FCQS algorithm is summarized below.

THE FUZZY C QUADRIC SHELLS (FCQS) ALGORITHM:

Fix the number of clusters C; fix m, $1 < m < \infty$;
Set iteration counter $l = 1$;
Initialize the fuzzy C-partition $U^{(0)}$ using the FCSS algorithm;
Repeat
  Compute $p_i^{(l)}$ for each cluster β_i by minimizing (3) subject to (5);
  Update $U^{(l)}$ using (6);
  Increment l;
Until ($\| U^{(l-1)} - U^{(l)} \| < \varepsilon$);

The FCQS algorithm has the following drawbacks: i) Since the algebraic distance given by (1) is highly nonlinear, the membership assignments are not very meaningful, ii) the constraint in (5) strictly speaking does not allow us to fit linear (or planar) clusters. We now address these drawbacks in more detail and propose modifications of the algorithm to overcome these drawbacks.

3. The Modified Fuzzy C Quadric Shells Algorithm

To overcome the problem due to the nongeometric nature of d^2_{Qij}, one may use the geometric (perpendicular) distance (denoted by d^2_{Pij}) between the point x_j and the shell β_i given by

$$d^2_{Pij} = \min \|x_j - z_{ij}\|^2 \text{ such that}$$
$$(z_{ij}{}^T A_i z_{ij} + z_{ij}{}^T v_i + b_i) = 0, \quad (7)$$

where z_{ij} is a point lying on the quadric curve describing cluster β_i. Using a Lagrange multiplier λ, Equation (7) can be solved for z_{ij} as

$$z_{ij} = \frac{1}{2}(\mathbf{I} - \lambda A_i)^{-1}(\lambda v_i + 2x_j). \quad (8)$$

Substituting (8) in (7) yields a quartic (fourth-degree) equation in λ in the 2-D case, which has at most four real roots λ_k, $1 \le k \le 4$. They can be easily computed using the standard solution from mathematical tables. For each real root λ_k so computed, we calculate the corresponding $(z_{ij})_k$ using (8). Then, we may compute $d^2_{Pij} = \min_k \|x_j - (z_{ij})_k\|^2$.

One can formulate the FCQS algorithm using d^2_{Pij} as the underlying distance measure [15]. Minimizing the resulting objective function with respect to U yields an equation identical to (6) where d^2_{Qij} is replaced by d^2_{Pij}. However, minimizing the objective function with respect to the parameters p_i results in coupled nonlinear equations with no closed-form solution. To overcome this problem, we may assume that we can obtain approximately the same solution by minimizing (3) subject to (5), which will be true if all the feature points lie reasonably close to the hyperquadric shells. This assumption leads to the Modified FCQS (MFCQS) algorithm. Our experimental results show that in the 2-D case the modified FCQS algorithm gives much better results and converges much faster than the original version. In fact, our extensive simulations indicate that the performance of this algorithm is excellent, as long as the data points are all reasonably close to the curves (i. e., as long as the data is not highly scattered), which will be true in most computer vision applications. This may be attributed to the fact that the membership assignment based on the perpendicular distance is more reasonable.

The MFCQS algorithm can also be used to find linear clusters, even though the constraint in (5) forces all prototypes to be of second degree. This is because the algorithm usually fits either two coincident lines (for a single line), or an extremely elongated ellipse (for two parallel lines) or a hyperbola (for two crossing lines). It is quite simple to recognize these situations from the parameters of the prototypes, and when these situations occur, we can simply split such prototypes to a pair of lines after the algorithm converges.

It is to be noted that d^2_{Pij} has a closed-form solution only in the 2-D case. In higher dimensions, solving for d^2_{Pij} is not trivial. For example, in the three dimensional case, this results in a sixth degree equation, which needs to be solved iteratively. This makes the algorithm slow. We now propose an alternative formulation of the algorithm to overcome this problem.

4. The Fuzzy C Plano-Quadric Shells Algorithm

When the exact (geometric) distance has no closed-form solution, one of the methods suggested in the literature is to use what is known as the "approximate distance" which is the first-order approximation of the exact distance. It is easy to show [7] that the approximate distance of a point from a curve is given by

$$d^2_{Aij} = d_A{}^2(x_j, \beta_i) = \frac{d^2_{Qij}}{|\nabla d^2_{Qij}|^2} = \frac{d^2_{Qij}}{p_i{}^T D_j D_j{}^T p_i}, \quad (9)$$

where ∇d^2_{Qij} is the gradient of the distance functional

$$p_i{}^T q = [p_{i1}, p_{i2}, \ldots, p_{ir}][x_1^2, x_2^2, \ldots, x_n^2, x_1x_2, \ldots, x_{(n-1)}x_n, x_1, x_2, \ldots, x_n, 1]^T \quad (10)$$

evaluated at x_j. In (9) the matrix D_j is simply the Jacobian of q evaluated at x_j.

One can easily reformulate the quadric shell clustering algorithm with d^2_{Aij} as the underlying distance measure. The objective function to be minimized in this case becomes

$$J_A(B,U) = \sum_{i=1}^{C}\sum_{j=1}^{N} (\mu_{ij})^m d^2_{Aij} = \sum_{i=1}^{C}\sum_{j=1}^{N} (\mu_{ij})^m \frac{p_i^T M_j p_i}{p_i{}^T D_j D_j{}^T p_i}. \quad (11)$$

Unfortunately, the minimization of the resulting objective function with respect to p_i in general leads to coupled nonlinear equations which can only be solved iteratively. To avoid this problem, we choose the constraints

$$p_i^T\left[\sum_{j=1}^{N}(\mu_{ij})^m D_j D_j^T\right]p_i = \sum_{j=1}^{N}(\mu_{ij})^m, \text{ or } \quad p_i^T G_i p_i = N_i, \; i = 1, \ldots, C, \tag{12}$$

where

$$G_i = \sum_{j=1}^{N}(\mu_{ij})^m D_j D_j^T \text{ and } N_i = \sum_{j=1}^{N}(\mu_{ij})^m. \tag{13}$$

The above constraint has been applied in the hard case by Taubin [8] with good results when there is only one curve to be fitted. Our contribution is to extend it to the fuzzy case and to fit C curves simultaneously. Using (12) and Lagrange multipliers, we may now minimize

$$\sum_{i=1}^{C}\sum_{j=1}^{N}(\mu_{ij})^m d_{Aij}^2 - \sum_{i=1}^{C}\lambda_i (p_i^T G_i p_i - N_i)$$

$$= \sum_{i=1}^{C}\sum_{j=1}^{N}(\mu_{ij})^m \frac{p_i^T M_j p_i}{p_i^T D_j D_j^T p_i} - \sum_{i=1}^{C}\lambda_i \left[\sum_{j=1}^{N}(\mu_{ij})^m p_i^T D_j D_j^T p_i - \sum_{j=1}^{N}(\mu_{ij})^m\right] \tag{14}$$

When most of the data points are close to the prototypes, the memberships μ_{ij} will be quite hard (i. e., they will be close to 0 or 1). This assumption is also quite good if we use possibilistic memberships [9] to be discussed in Section 5. This means that when the constraint in (12) is satisfied, we may say that $p_i^T D_j D_j^T p_i \approx 1$. In fact, it is easy to show that the condition $p_i^T D_j D_j^T p_i \approx 1$ is exactly true for the case of lines/planes and certain quadrics such as circles and cylinders. Hence, we will obtain approximately the same solution if we minimize

$$\sum_{i=1}^{C}\sum_{j=1}^{N}(\mu_{ij})^m p_i^T M_j p_i - \sum_{i=1}^{C}\lambda_i\left[\sum_{j=1}^{N}(\mu_{ij})^m p_i^T D_j D_j^T p_i - \sum_{j=1}^{N}(\mu_{ij})^m\right].$$

If we assume that the prototypes are independent of each other, then this is equivalent to independently minimizing

$$\sum_{j=1}^{N}(\mu_{ij})^m p_i^T M_j p_i - \lambda_i\left[p_i^T\sum_{j=1}^{N}(\mu_{ij})^m D_j D_j^T p_i - \sum_{j=1}^{N}(\mu_{ij})^m\right]$$

$$= p_i^T F_i p_i - \lambda_i (p_i^T G_i p_i - N_i) \tag{15}$$

where

$$F_i = \sum_{j=1}^{N}(\mu_{ij})^m M_j . \tag{16}$$

The solution of (16) is given by the generalized eigenvector problem

$$F_i p_i = \lambda_i G_i p_i , \tag{17}$$

which can be converted to the standard eigenvector problem if the matrix G_i is not rank-deficient. Unfortunately this is not the case. In fact, the last row of D_j is always [0, . . . ,0]. Equation (17) can still be solved using other techniques that use the modified Cholesky decomposition [8], and the solution is computationally quite inexpensive when the feature space is 2-D or 3-D. Another advantage of this constraint is that it can also fit lines and planes in addition to quadrics. Minimization of (11) with respect to the memberships μ_{ij} leads to

$$\mu_{ij} = \begin{cases} \dfrac{1}{\sum_{k=1}^{C}\left(\dfrac{d_{A_{ij}}}{d_{A_{kj}}}\right)^{\frac{2}{m-1}}} & \text{if } I_j = \Phi \\ 0 \quad i \notin I_j \;\text{ if } I_j \neq \Phi & \\ \sum_{i\in I_j}\mu_{ij} = 1 & \text{if } I_j \neq \Phi \end{cases} \tag{18}$$

In the 2-D case, d_{Aij}^2 in the above equation may also be substituted by d_{Pij}^2. We notice that in practice this gives more rapid convergence. The resulting clustering algorithm, which we call the Fuzzy C Plano-Quadric Shells algorithm, is summarized below.

THE FUZZY C PLANO-QUADRIC SHELLS (FCPQS) ALGORITHM:

Fix the number of clusters C; fix m, $1 < m < \infty$;
Set iteration counter $l = 1$;
Initialize the fuzzy C-partition $U^{(0)}$;
Repeat
  Compute the matrices F_i and G_i using (13) and (16)
  Compute $p_i^{(l)}$ for each cluster β_i solving (17)
  Update $U^{(l)}$ using (18);
  Increment l;
Until ($\| U^{(l-1)} - U^{(l)} \| < \varepsilon$);

5. Robust Shell Clustering

The algorithms discussed above will be sensitive to outlier points even when the objective function based on the approximate distance is minimized. To overcome this problem, we have converted the algorithm to a possibilistic algorithm [9]. This is very easily achieved by updating the memberships according to

$$\mu_{ij} = \frac{1}{1 + \left(\frac{d_{ij}^2}{\eta_i} \right)^{\frac{1}{m-1}}} \tag{23}$$

instead of (18). In (23), one attractive choice for η_i in practice is the average fuzzy intra-cluster distance given by

$$\eta_i = \frac{\sum_{j=1}^{N} \mu_{ij}^m d_{ij}^2}{\sum_{j=1}^{N} (\mu_{ij})^m}. \tag{24}$$

Our experimental results show that the resulting algorithm, which we call the Possibilistic C Plano-Quadric Shells (PCPQS) algorithm, is quite robust in the presence of poorly defined boundaries (i. e., when the edge points are somewhat scattered around the ideal boundary curve in the 2-D case and when the range values are not very accurate in the 3-D case). It is also very immune to impulse noise and outliers, as can be seen in the examples presented in Section 7. A possibilistic version of the Modified FCQS algorithm (denoted by MPCQS) was also implemented.

6. Determination of Number of Clusters

The number of clusters C is not known *a priori* in some pattern recognition applications and most computer vision applications. When the number of clusters is unknown, one method to determine this number is to perform clustering for a range of C values, and pick the C value for which a suitable validity measure is minimized (or maximized) [10,12]. However this method is rather tedious, especially when the number of clusters is large. Also, in our experiments, we found that the C value obtained this way may not be optimum. This is because when C is large, the clustering algorithm sometimes converges to a local minimum of the objective function, and this may result in a bad value for the validity of the clustering, even though the value of C is correct. Moreover, when C is greater than the optimum number, the algorithm may split a single shell cluster into more than one cluster, and yet achieve a good value for the overall validity. To overcome these problems, we propose an alternative Unsupervised C Shell Clustering algorithm which is computationally more efficient, since it does not perform the clustering for an entire range of C values.

Our proposed method progressively clusters the data starting with an overspecified number C_{max} of clusters. Initially, the FCPQS algorithm is run with $C=C_{max}$. After the algorithm converges, spurious clusters (with low validity) are eliminated; compatible clusters are merged; and points assigned to clusters with good validity are temporarily removed from the data set to reduce computations. The FCPQS algorithm is invoked again with the remaining feature points. The above procedure is repeated until no more elimination, merging, or removing occurs, or until C=1. This algorithm is summarized below.

THE UNSUPERVISED POSSIBILISTIC C PLANO-QUADRIC SHELLS (UPCPQS) ALGORITHM:

Set $C = Cmax$; fix m , $1 < m < \infty$;
CRemoved := 0; *MergeFlag* := *EliminmateFlag* := *RemoveFlag* := TRUE;
While $C > 1$ **and** (*MergeFlag* = TRUE **or** *EliminmateFlag* = TRUE **or**
RemoveFlag = TRUE) **do**
MergeFlag := *EliminmateFlag* := *RemoveFlag* := FALSE;
Perform the PCPQS algorithm with the number of clusters = C;
Eliminate spurious clusters using validity, decrement C accordingly,
and set *EliminmateFlag* = TRUE if any elimination has occurred;
Merge compatible prototypes among the C prototypes , update C,
and set *MergeFlag* = TRUE if merging has occurred;
Remove good clusters using validity, update C, and set *RemoveFlag* = TRUE if
any good clusters are removed;
Save the remaining clusters' prototypes ;
End While
Replace all the removed feature points back into the data set.
Append the list of remaining clusters' prototypes from the last iteration in the while loop
to the list of removed clusters' prototypes;
Do
Perform the PCPQS algorithm with the new C;
Merge compatible prototypes in the prototype list and update C ;
Eliminate tiny clusters and decrement C accordingly;
Until No more merging or elimination takes place;

One way to determine if two clusters are compatible (i. e., whether they can be merged), is to estimate the best fit for all the points having a membership greater than an α-cut in the two clusters. If the validity for the resulting cluster is good, then the two clusters are considered mergeable. The above algorithm also requires a validity measure to discriminate between good and bad clusters. Several cluster validity criteria have been presented in the literature. For example, performance measures based on the memberships in the partition matrix U have been proposed by some researchers [6,10]. Unfortunately, these are not very effective for shell clusters, since they do not reflect the actual geometric structure of the data set. One possible validity measure we may define is the shell thickness measure, which is simply the sum of the squared errors of the fit for the ith cluster given by

$$T_i = \sum_{j=1}^{N} (\mu_{ij})^m d_{Pij}^2 . \tag{19}$$

However, it is difficult to estimate a "good" value for this validity measure in noisy conditions. Validity measures may also be defined using hypervolume and density [11,12]. To do this, the distance vector from a feature point to a shell prototype is first defined as $\delta_{ij} = (x_j - z_i)$, where z_i is the closest point on the curve (or surface) to the feature point x_j in the approximate distance sense. The fuzzy spherical shell covariance matrix is defined by

$$\Sigma_i = \frac{\sum_{j=1}^{N} (\mu_{ij})^m \delta_{ij} \delta_{ij}^T}{\sum_{j=1}^{N} (\mu_{ij})^m} . \tag{20}$$

Using (15) the fuzzy shell hypervolume and the shell density may be defined as

$$V_i = \sqrt{\det(F_i)}, \quad \text{and} \quad D_i = \frac{S_i}{V_i}, \tag{21}$$

where S_i is the sum of close members of shell β_i given by

$$S_i = \Sigma_{,}\mu_{ij} \quad \text{such that} \quad \delta_i^T \Sigma_i^{-1} \delta_i < 1. \tag{22}$$

However, the above measures are not very reliable because their values can vary widely for good clusters, depending on the sizes of the clusters and noise. They can also be "good" for spurious clusters. Therefore, we have developed a new validity measure for shell clusters based on the idea of curve (surface) density, which is a measure of the number of feature points per unit length (surface area) of the shell cluster. We have also developed methods to estimate the effective curve length (surface area) of the shell clusters when the curves (surfaces) are partial. A more detailed discussion of this validity measure will be the subject of a future paper.

7. Experimental Results

Although the algorithms presented in the previous sections are applicable to feature spaces of any dimension, in this paper we present only results of two-dimensional data sets. In all the examples shown in this paper, the UPCPQS algorithm was applied with the fuzzifier $m = 2$ and $C_{max} = 25$. To obtain a good initialization of the fuzzy C-partition $U^{(0)}$, we run the Gustafson-Kessel algorithm with $m = 1.5$ for a few iterations (which gives an excellent linear approximation of the data) followed by the Fuzzy C Spherical Shells algorithm [2]. This was observed to give excellent results. The data sets consists of object edges obtained by applying an edge operator to real images. Uniformly distributed noise with an interval of 30 was added to the images to make them noisy. The edge images were then thinned [14] to reduce the number of pixels to be processed. The resulting input images typically had about 2000 points. The PCPQS algorithm still sometimes fits second-degree curves for linear clusters, especially when the data is scattered. Therefore, the algorithm was modified to identify such situations and split such clusters into lines after convergence, as explained in Section 3. In practice, there seems to be very little difference between the PCPQS and MPCQS algorithms in the 2-D case.

Figure 1(a) shows the original noisy image of a box with holes. The edge-detected and thinned image is shown in Figure 1(b). As can be seen, there are many noise points, and the pixel boundaries are not always well-defined. Figure 1(c) shows the result of the UPCPQS algorithm. The final prototypes are shown superimposed on the edge image. The prototypes are virtually unaffected by noise and poor boundaries. Figure 1(d) shows the "cleaned" edge map. This is obtained by plotting the boundaries generated by the prototypes only in locations where there is at least one pixel with a high membership value in a 3×3 neighborhood. Figures 2 and 3 show similar results for images with collections of objects of various sizes and shapes.

8. Summary

In this paper, we propose a new approach to boundary and surface approximation in computer vision. Current techniques to describe boundaries and surfaces in terms of parametrized or algebraic forms have the following disadvantages: i) Many techniques apply in cases when the boundaries/surfaces belonging to different objects have already been segmented, ii) they look for local structures and use edge following or region growing and hence would be sensitive to local aberrations and deviations in shapes , iii) they are computationally intensive and the memory requirement are high, iv) they require features (such as curvature and surface normals) to be calculated and hence are sensitive to noise and the computed features are inaccurate at boundaries of surfaces, v) most of the feature-based techniques assume dense data and hence are not suitable if the data is sparse or if there are gaps in the data, and vi) some methods are not invariant to rigid transformations. The approach we propose overcomes these drawbacks. If the clustering is performed in the feature space, it can have the disadvantages of high dimensionality, and loss of pixel adjacency information. However, since the proposed methods apply clustering techniques directly to image data, they do not suffer from these disadvantages. Another disadvantage of clustering methods is that the number of clusters has to be known in advance. The proposed approach overcomes this problem by using new cluster validity measures and compatible cluster merging.

Linear and Quadric shapes are not sufficiently general for all computer vision applications. We propose to extend our algorithm to more general shells such as those represented by algebraic curves, or superquadrics. Currently there are no algorithms that simultaneously fit an unknown number of general curves (or surfaces) to noisy and/or scattered data. This includes boundaries and surfaces that are locally very noisy, and boundaries and surfaces that are sparsely sampled. Methods based on feature computation and region growing do not work in these cases.

9. References

1. R. N. Dave, "Fuzzy-shell clustering and applications to circle detection in digital images", *International Journal of General Systems*, vol. 16, 1990, pp. 343-355.
2. R. Krishnapuram, H. Frigui, and O. Nasraoui, "New fuzzy shell clustering algorithms for boundary detection and pattern recognition", *Proceedings of the SPIE Conference on Intelligent Robots and Computer Vision X: Algorithms and Techniques*, Boston, Nov. 1991, pp.458-465.
3. R. Krishnapuram, O. Nasraoui, and H. Frigui, "The fuzzy C spherical shells algorithm: A new approach", *IEEE Transactions on Neural Networks*, vol. 3, no. 5, September 1992.
4. R. N. Dave and K. Bhaswan, "Adaptive fuzzy C-shells clustering and detection of ellipses", to appear in the *IEEE Transactions on Neural Networks*, vol. 3, no. 5, 1992.

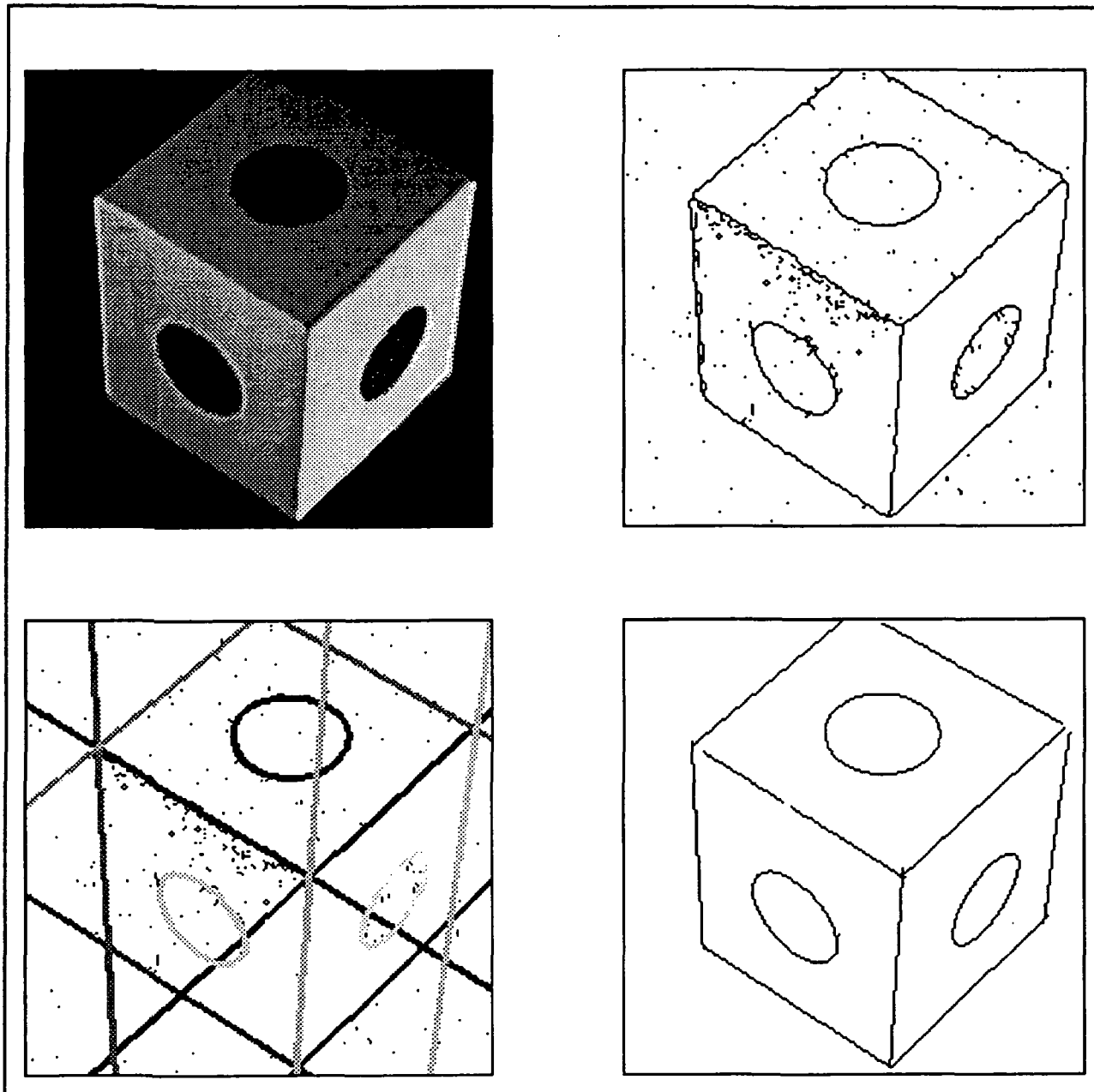

Figure 1: Results of the UPCPQS algorithm for the "box" image. (a) original image, (b) edge pixels obtained after thinning, (c) detected clusters with prototypes superimposed on the edge image in (b), and (d) the "cleaned" edge image.

5. F. L. Bookstein, "Fitting conic sections to scattered data", Computer Vision, Graphics, and Image Processing, vol. 9, pp. 56-71, 1979.
6. J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York, 1981.
7. R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*, vol. I, Addison Wesley, Reading, MA, 1992, Chapter 11.
8. G. Taubin, "Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 11, Nov. 1991, pp. 1115-1138.
9. R. Krishnapuram and J. M. Keller, "A possibilistic approach to clustering", submitted to *IEEE Transactions on Fuzzy Systems*.

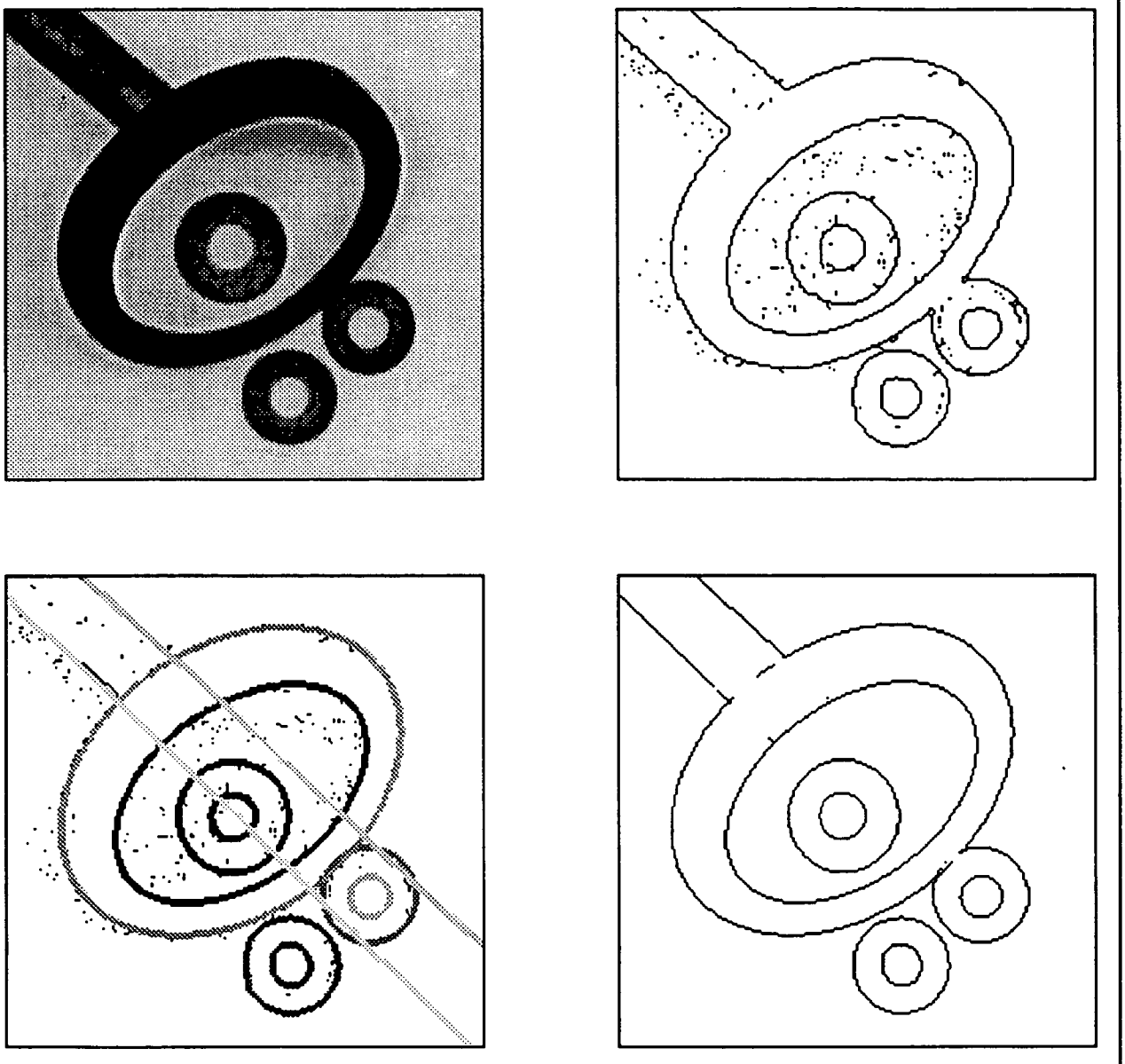

Figure 2: Results of the UPCPQS algorithm for the "magnifying glass" image. (a) original image, (b) edge pixels obtained after thinning, (c) detected clusters with prototypes superimposed on the edge image in (b), and (d) the "cleaned" edge image.

10. E. Backer and A. K. Jain, "A Clustering Performance Measure Based on Fuzzy Set Decomposition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 3, pp. 66, 1981.
11. I. Gath and A. B. Geva, Unsupervised optimal fuzzy clustering, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, July 1989, 773-781.
12. R. N. Dave, "New measures for evaluating fuzzy partitions induced through C-shells clustering", *Proceedings of the SPIE Conference on Intelligent Robots and Computer Vision X: Algorithms and Techniques*, Boston, Nov. 1991, pp. 406-414.
13. R. Krishnapuram and C.-P. Freg, "Fitting an Unknown Number of Lines and Planes to Image Data through Compatible Cluster Merging", *Pattern Recognition*, vol. 25, no. 4, 1992.

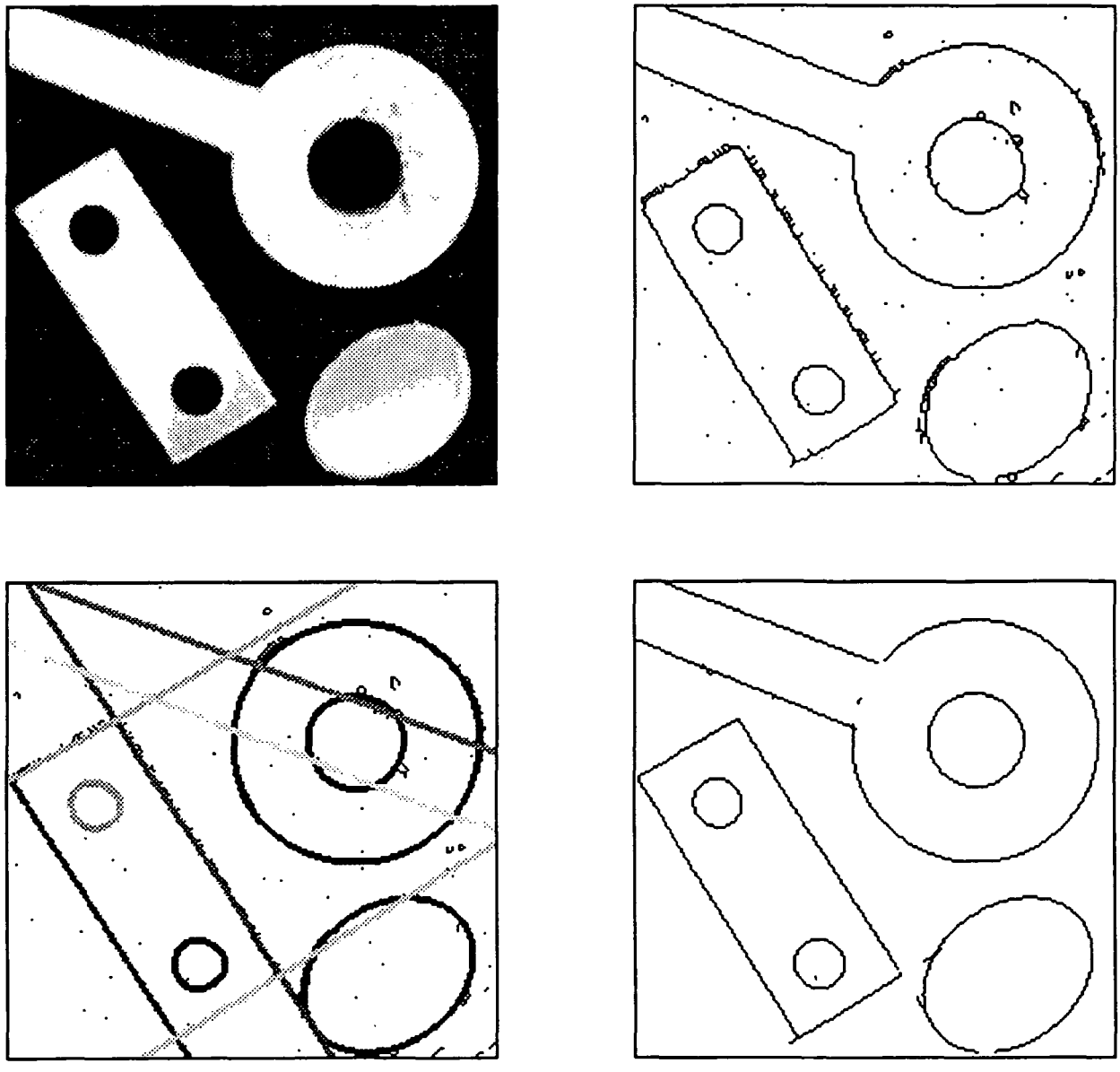

Figure 3: Results of the UPCPQS algorithm for the "parts" image. (a) original image, (b) edge pixels obtained after thinning, (c) detected clusters with prototypes superimposed on the edge image in (b), and (d) the "cleaned" edge image.

14. R. Krishnapuram and L. Chen, "Implementation of Parallel Thinning Algorithms Using Iterative Neural Networks", to appear in the *IEEE Transactions on Neural Networks*, 1992.
15. R. Krishnapuram, H. Frigui, and O. Nasraoui, "Quadric shell clustering algorithms and their applications", submitted to *Pattern Recognition Letters*.