# Fuzzy and Possibilistic Shell Clustering Algorithms and Their Application to Boundary Detection and Surface Approximation—Part II

Raghu Krishnapuram, *Member, IEEE*, Hichem Frigui, and Olfa Nasraoui

***Abstract*— Shell clustering algorithms are ideally suited for computer vision tasks such as boundary detection and surface approximation, particularly when the boundaries have jagged or scattered edges and when the range data is sparse. This is because shell clustering is insensitive to local aberrations, it can be performed directly in image space, and unlike traditional approaches it does assume dense data and does not use additional features such as curvatures and surface normals. The shell clustering algorithms introduced in Part I of this paper assume that the number of clusters is known, however, which is not the case in many boundary detection and surface approximation applications. This problem can be overcome by considering cluster validity. In this paper, we introduce a validity measure called surface density which is explicitly meant for the type of applications considered in this paper. We show through theoretical derivations that surface density is relatively invariant to size and partiality (incompleteness) of the clusters. We describe unsupervised clustering algorithms that use the surface density measure and other measures to determine the optimum number of shell clusters automatically, and illustrate the application of the proposed algorithms to boundary detection in the case of intensity images and to surface approximation in the case of range images.**

## I. Motivation

BOUNDARY detection and surface approximation are important components of many computer vision applications such as object shape recognition and object orientation estimation. There is a plethora of techniques to fit parameterized curves such as conics to segmented edge pixels [15] and to fit parameterized surfaces to segmented range data [2], [5], [12], [13], [19], [25], [26], [28]. Segmentation of edge and range data is difficult in the case of jagged edges and noisy or sparse range data, since features such as gradients and curvatures cannot be computed reliably. Jagged edges occur frequently in poor quality images and also in good quality images between textured regions (see [23] for examples). Many range finders produce only sparse data. A better approach in such situations would be to perform segmentation and boundary/surface fitting simultaneously on the data, without making use of features that assume continuity and smoothness of the edges and surfaces. Since shell clustering algorithms can perform segmentation and fitting simultaneously, they are ideally suited for boundary detection and surface approximation when the boundaries/surfaces are ill defined. They also require far less computations and memory compared to the GHT methods (see Section X of Part I of this paper). Since they look for global structures and do not use edge following or region growing, they are insensitive to local aberrations and deviations in shape. They do not use features such as gradients and curvature and hence are not sensitive to noise and sharp discontinuities at the boundaries.

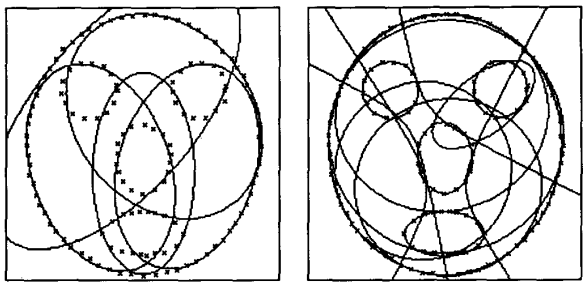
A major disadvantage of shell clustering methods is that the number of clusters has to be known in advance. Traditionally, the "optimum" number of clusters is determined by evaluating a certain global validity (performance) measure of the $C$-partition for a range of $C$ values, and then picking the value of $C$ that optimizes the validity measure in some sense [4], [10], [18]. This is a very tedious and computationally expensive process, however, since one needs to cluster the data for each value of $C$. Moreover, since many performance measures are monotonic in $C$, a significant point (such as a knee point) of the performance measure must be identified to select the optimum number of clusters, and this is not always easy. In the case of shell clustering, the algorithms frequently converge to local minima, particularly when the data is complex. When the $C$-partition corresponds to a local minimum rather than a global one, the computed performance measures will not be correct. As an example, Fig. 1(a) shows the result of the FCQS algorithm on a data set with five clusters. The prototypes found by the algorithm are superimposed on the original data set. None of the five clusters is characterized correctly. The solution corresponds to a local minimum, which is usually the result of poor initialization. Fig. 1(b) shows the result of the same algorithm with $C = 12$. This time the algorithm identifies all the clusters correctly, but it also finds many spurious clusters. The partition in Fig. 1(b) may well have a more optimum value of a given validity measure compared to that of Fig. 1(a). The situation is worse when the data set contains outliers, because in this case, the objective function may be globally optimized by a partition of the data set that is intuitively incorrect. In all these cases, the method of picking $C$ that optimizes a certain global performance measure fails.

Another approach to determining the number of clusters $C$ is to perform progressive clustering [8], [17], [19], [26]. In this approach, after convergence of the clustering algorithm with an overspecified number of clusters. Spurious clusters are eliminated, compatible clusters are merged, "good" clusters are identified, and points belonging to these clusters are

Manuscript received February 3, 1993; revised April 25, 1994. This work was supported in part by the Alexander von Humboldt Foundation, Germany.

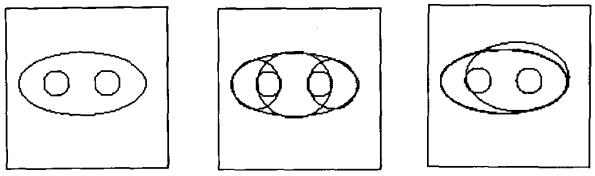
The authors are with the Department of Electrical and Computer Engineering, University of Missouri-Columbia, Columbia, MO 65211 USA.

IEEE Log Number 9406653.



(a) (b)

Fig. 1. A data set with five elliptical clusters for which the FCQS algorithm fails when the correct number of clusters is specified. (a) Result with number of clusters = 5. (b) Result with number of clusters = 12.



(a) (b) (c)

Fig. 2. (a) A synthetic image with two circles surrounded by a large ellipse (b) Result of the PCQS algorithm with number of clusters = 3. The prototypes are shown superimposed on the original image. (c) Result of the PCQS algorithm with number of clusters = 6.

temporarily removed from the data set. The clustering is performed again with the reduced number of clusters and data points. This procedure is repeated until no good clusters can be removed anymore or until no data points are left. This approach is more efficient than the previous one, since the clustering process need not be repeated for an entire range of $C$-values. Its results are also less influenced by local minima and noise, since it is based on the idea that at least one good cluster is found each time. As an example, Fig. 2(a) shows a data set with a large elliptical cluster surrounding two smaller circles. Fig. 2(b) shows the result of the Possibilistic $C$ Quadric Shells (PCQS) algorithm (see Section IX of Part I of this paper) when $C$ is chosen to be three, the optimum number of clusters. This result is obviously a local minimum. Fig. 2(c) shows the result of the PCQS algorithm with an overspecified number of clusters $C = 6$. The algorithm finds three identical elliptical clusters, two circular clusters, and a spurious cluster. Thus, by merging the three compatible clusters and eliminating the spurious cluster, one can obtain the correct final partition. For this approach to work, however, one needs a validity criterion to evaluate the goodness of a particular cluster.

In summary, two kinds of validity measures are needed, depending on which approach is used to determine the optimum number of clusters. In the first approach, we need a global performance measure that evaluates the overall partition of the data set into $C$ clusters. In the second approach, we need an individual cluster validity measure that evaluates the goodness of a particular cluster.

In this paper, we consider validity measures for quadric shell clusters, i.e., clusters that resemble thin shells whose prototypes can be represented by second-degree hypersurfaces. In Section II, we review some validity measures that have been suggested in the literature and show through examples that these validity measures cannot always distinguish between good clusters and spurious clusters. In Section III, we introduce a new validity measure called surface density, which is particularly suitable for the type of applications being considered in this paper. We show through theoretical derivations that this measure is relatively independent of the size of the clusters as well as their partiality (incompleteness). To illustrate the use of the proposed validity measure, in Sections IV and V we develop boundary detection and surface approximation algorithms that use the surface density criterion and other validity measures to determine the optimum number of clusters automatically. Several experimental results involving boundary detection and surface approximation are presented in Section VI. Section VII contains the summary conclusions.

## II. Cluster Validity

### A. Global Validity Measures

Cluster validity has been extensively discussed in the literature. Some methods measure the amount of fuzziness in each $C$-partition, and presume the least fuzzy partitions to be the most valid. This is consistent with the idea that in a fuzzy partition, feature vectors with the hardest memberships have the least uncertainty associated with their classification. Hence, the ultimate goal of these measures is to find the partition that minimizes the classification uncertainty. The underlying assumption that "good" clusters are not actually very fuzzy is only true when the data set consists of well separated clusters. Examples of such validity measures are the partition coefficient [3], the fuzzy set decomposition measure [1], the classification entropy [4], the proportion exponent [34], and the polarization degree [9]. Unfortunately, in our experience these measures generally do not perform well in the case of shell clusters. This may be attributed to the fact that, these measures are based solely on the fuzzy partition matrix, and hence they lack any direct connection to the geometric properties of the clusters themselves. Moreover, they cannot be used with possibilistic clustering methods [25], because they are based on the fact that the sum of the memberships of a data point across the classes is one.

Validity measures that relate more to the nature of the geometry of the data set also exist. Many are defined for the hard case, but can be easily generalized to the fuzzy case. Examples of such validities are the sum-of-squared-errors criterion [31], the related-minimum-variance criterion [11] and the figure of merit [16]. Gath and Geva introduced criteria based on hypervolume and density [14]. They define the fuzzy hypervolume to be

$$V = \sum_{i=1}^{C} [\det \mathbf{C}_i]^{1/2}$$

where $\mathbf{C}_i$ is the fuzzy covariance matrix of cluster $\beta_i$ given by

$$\mathbf{C}_i = \frac{1}{N_i} \sum_{j=1}^{N} (u_{ij})^m (\mathbf{x}_j - \mathbf{c}_i)(\mathbf{x}_j - \mathbf{c}_i)^{\mathrm{T}}. \qquad (1)$$

In the above equation, $u_{ij}$ is the membership of feature vector $\mathbf{x}_j$ in cluster $\beta_i$, $m \in [1, \infty)$ is the fuzzifier, $\mathbf{c}_i$ is the cluster center, and

$$N_i = \sum_{j=1}^{N} (u_{ij})^m. \tag{2}$$

The average partition density is defined as

$$D_A = \frac{1}{C} \sum_{i=1}^{C} \frac{S_i}{[\det \mathbf{C}_i]^{1/2}}$$

where $S_i$ is the "sum of central members" of cluster $\beta_i$ given by

$$S_i = \sum_j u_{ij} \quad \text{such that } (\mathbf{x}_j - \mathbf{c}_i)^{\mathrm{T}} \mathbf{C}_i^{-1} (\mathbf{x}_j - \mathbf{c}_i) < 1.$$

The partition density is defined as

$$D_P = \frac{S}{V}$$

where

$$S = \sum_{i=1}^{C} S_i.$$

The validity criteria discussed so far are meant for data sets consisting of "filled" (compact) clusters. To evaluate partitions that consist of shell-type clusters, different validity measures must be used. Davé [7] redefined the hypervolume and partition density performance measures for the case of spherical and elliptical shells. Here we generalize these definitions to more general shell types.

Let the distance vector $\tau_{ij}$ from a feature point $\mathbf{x}_j$ to a shell prototype $\beta_i$ be defined by

$$\tau_{ij} = (\mathbf{x}_j - \mathbf{z}_j^i) \tag{3}$$

where $\mathbf{z}_j^i$ is the closest point on the shell $\beta_i$ to $\mathbf{x}_j$. In particular, for spherical clusters

$$\tau_{ij} = (\mathbf{x}_j - \mathbf{c}_i) - r_i \frac{(\mathbf{x}_j - \mathbf{c}_i)}{\|\mathbf{x}_j - \mathbf{c}_i\|}$$

where $\mathbf{c}_i$ is the center of cluster $\beta_i$. For other second-degree curves, the point $\mathbf{z}_j^i$ can be determined as explained in Appendix A of Part I of this paper. For more general types of shells, the point $\mathbf{z}_j^i$ may be difficult to compute. Therefore one can simply use the approximate closest point on the shell. For example, if the equation of the shell prototype is given by a set of functions $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_k(\mathbf{x})]^{\mathrm{T}} = \mathbf{0}$ (see Section VIII of Part I of this paper), then the approximate closest point to $x_j$ is

$$\mathbf{z}_j^i = \mathbf{x}_j - Df^{\mathrm{T}} [Df Df^{\mathrm{T}}]^{-1} \mathbf{f}(\mathbf{x}_j)$$

where $Df$ is the Jacobian of $\mathbf{f}(\mathbf{x})$ evaluated at $\mathbf{x}_j$ given by

$$Df(\mathbf{x}_j) = \begin{bmatrix} \frac{\partial f_1(\mathbf{x}_j)}{\partial x_1} & \cdots & \frac{\partial f_k(\mathbf{x}_j)}{\partial x_1} \\ \frac{\partial f_1(\mathbf{x}_j)}{\partial x_n} & \cdots & \frac{\partial fk(\mathbf{x}_j)}{\partial x_n} \end{bmatrix}.$$

If $k = 1$, then $Df(\mathbf{x}_j)$ reduces to the gradient $\nabla f(\mathbf{x}_j)$. The fuzzy shell covariance matrix can now be defined as

$$C_{Si} = \frac{1}{N_i} \sum_{j=1}^{N} (u_{ij})^m \tau_{ij} (\tau_{ij})^{\mathrm{T}}.$$

The shell hypervolume of a cluster may be defined as

$$V_{Si} = \sqrt{\det(\mathbf{C}_{Si})}. \tag{4}$$

The shell density of a cluster may be defined as

$$D_{Si} = \frac{S_i}{V_{Si}} \tag{5}$$

where $S_i$ is the sum of close members of shell $\beta_i$ given by

$$S_i = \sum_j u_{ij} \quad \text{such that } \tau_{ij}^T \mathbf{C}_{Si}^{-1} \tau_{ij} < 1.$$

The total shell hypervolume $V_S$ and shell partition density, $D_S$ may be defined as

$$V_S = \sum_{i=1}^{C} V_{Si} \quad \text{and} \quad D_S = \sum_{i=1}^{C} D_{Si}.$$

Davé also proposed another global validity index that measures the thickness of a spherical shell [8]. This measure is given by

$$T = \frac{\sum_{i=1}^{C} \left[ \sum_{j=1}^{N} (u_{ij})^m (\|\mathbf{x}_j - \mathbf{c}_i\| - r_i)^2 / N_i \right]}{\left[ \sum_{i=1}^{C} r_i \right] / C}. \tag{6}$$

Krishnapuram *et al.* proposed similar measures [22], [24]. The total fuzzy average shell thickness $T_S$ is defined as follows.

$$T_S = \sum_{i=1}^{C} \frac{1}{N_i} \sum_{j=1}^{N} (u_{ij})^m \|\tau_{ij}\|^2. \tag{7}$$

When $\|\tau_{ij}\|^2$ is difficult to compute, the approximate distance from $\mathbf{x}_j$ to a shell prototype $\beta_i$ may be used instead. The global shell thickness measures in (6) and (7) have a monotonic tendency, thus making it difficult to choose the optimum number of clusters.

The global performance measures discussed in this section may be used to evaluate an overall $C$-partition of a data set, which is to be used in determining the optimal number of clusters. As we pointed out in Section I, this method of performing the clustering for an entire range of $C$-values is very time consuming, and is not guaranteed to work due to local minima, particularly for noisy or complex data sets.

### *B. Individual Validity Measures*

As discussed in Section I, the second approach to determine the optimal number of clusters performs better than the first approach and is computationally more efficient. In this approach we need to evaluate the goodness of a particular cluster, so that we can make a decision as to whether it is a spurious cluster to be eliminated or a good cluster that should be temporarily removed from the data set. Several individual validity measures can be defined for shell clusters.

Krishnapuram *et al.* proposed an individual cluster validity measure for spherical shell-type clusters [24]. The fuzzy

TABLE I
VALUES OF SHELL THICKNESS, SHELL HYPERVOLUME, AND SHELL DENSITY FOR THE CLUSTERS IN FIG. 3

| Cluster | shell thickness | shell hypervolume | shell density |
|---|---|---|---|
| Large circle | 0.088 | 0.044 | 2592.5 |
| Small circle | 0.088 | 0.044 | 1317.3 |
| Large half circle | 0.115 | 0.057 | 1028.5 |
| Sparse circle | 0.087 | 0.044 | 827.3 |

average shell thickness was defined to be

$$T_{Si} = \frac{1}{N_i} \sum_{j=1}^{N} (u_{ij})^m \|\tau_{ij}\|^2. \tag{8}$$

In (8), the thickness $T_{Si}$ of each shell may be divided by the radius $r_i$ so that it is measured relative to its radius. Similarly, the individual shell hypervolume of a cluster defined in (4), and the individual shell partition density as defined in (5), may be used as individual validity measures.

The above cluster validity measures suffer from many drawbacks from the point of view of the applications considered in this paper. The major one being their large variability depending on the size, and the partiality (or incompleteness) of the shell cluster. They also lack a standard (theoretical) value against which one can compare the validity of a particular cluster. For example, the hypervolume and shell thickness may be very small for certain types of spurious clusters that contain only a few points. Fig. 3 shows an example to illustrate the problems with the validity measures mentioned above. The data set consists of four clusters: a large circle, a half-circle with the same radius, and a small circle with half the radius, and a sparse circle. After correctly clustering this data set using the Fuzzy $C$ Spherical Shells algorithm [24] with $C = 4$, the values of the individual cluster validities hypervolume, shell density, and shell thickness of each cluster were computed. These are listed in Table I. As can be seen, the shell density varies very much depending on the partiality, size, and sparseness of the circle. Variation due to sparseness is not very important in the type of applications we consider, since the boundaries of objects can be assumed to have approximately the same sparseness in a given image. Objects in range images also have approximately the same sparseness. Although the variation in shell density due to size can be overcome by normalizing the value by the radius, variation due to partiality will still exist. Moreover, normalization in the case of other types of curves is not so straightforward. On the other hand, hypervolume and average shell thickness are quite good in this example.

The situation is different when there are spurious clusters in the final partition. These validities can be quite misleading. As an example, Table II lists the validity measures average shell thickness, shell hypervolume, and shell density for the fuzzy partition corresponding to the result shown in Fig. 2(c). In this case, the average shell thickness, shell hypervolume and shell density for the spurious cluster are better than those for the two good circular clusters, giving the wrong indication that the spurious cluster is better than the good clusters. This is because

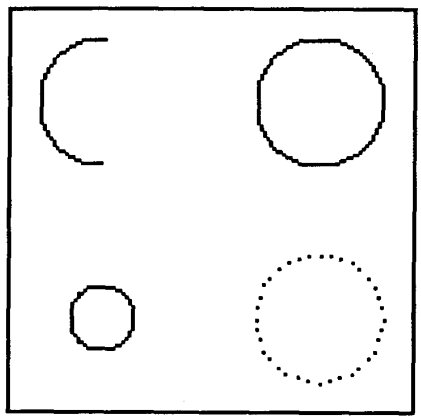


Fig. 3. A synthetic image with a large circle, a semi-circle, and a small circle.

TABLE II
VALUES OF SHELL THICKNESS, SHELL HYPERVOLUME, AND SHELL DENSITY FOR THE CLUSTERS IN FIG. 2(C)

| Cluster | shell thickness | shell hypervolume | shell density | total membership |
|---|---|---|---|---|
| Large ellipse 1 | 0.010 | 0.0039 | 56256.7 | 222 |
| Large ellipse 2 | 0.010 | 0.0042 | 52325.2 | 217 |
| Large ellipse 3 | 0.012 | 0.0059 | 37843.0 | 214 |
| Circle 1 | 0.064 | 0.0316 | 1834.1 | 58 |
| Circle 2 | 0.088 | 0.0439 | 1321.9 | 58 |
| Spurious ellipse | 0.026 | 0.0098 | 7467.1 | 71 |

the spurious cluster has a small error of fit (as indicated by the average shell thickness). Sometimes spurious clusters can be detected by the fact that they contain very few points (i.e., the total membership in such clusters is low), however, one cannot always distinguish between spurious clusters with a fair number of points (as in the present case) and partial curves. Most importantly, there is more than an order of magnitude variation in the values of these validities among the good clusters, which makes it hard to select thresholds to differentiate between good and spurious clusters. For partial shells and small shells, since the sum of central members is relatively small, the shell density is low. This may happen even for uniformly thick shells with clean edges, as in this case.

The above examples illustrate the fact that these validities are not entirely adequate for evaluating the goodness of a shell cluster for the type of applications we consider in this paper. This does not mean, however, that these validities are without utility. They can be used to provide additional information, especially when the surface density criterion (to be introduced next) has borderline values. This idea will be used in our unsupervised algorithms in Section IV.

## III. THE SURFACE DENSITY CRITERION

### A. Definitions

The validity measures defined in the previous section do not take into account the fact that shell clusters lie in subspaces of feature space. A more appropriate measure of validity for such

subspace clusters needs to relate to the density of the cluster as viewed in the subspace. We now develop an alternative validity measure specifically meant for shell clusters.

The proposed validity measure is related to the shell density measure introduced in Section II-A. Instead of measuring the number of points per unit shell volume, however, we measure the number of points per unit curve length or surface area. Using curve length or surface area takes into account the fact that shell clusters are subspace clusters. Computing the arc length or surface area of a shell cluster is quite trivial if the points in the cluster form a connected chain. We cannot assume connectivity, however, since the shell may be sparse, and since the points belonging to the shell may be scattered around the ideal prototype. Therefore, one has to resort to a method that can yield a good estimate rather than the exact value of the arc length or the surface area spanned by the cluster. As an estimate of the arc length spanned by a possibly incomplete shell cluster, one may use the circumference (surface area) of a complete circle (sphere) which is in some sense equivalent to the possibly incomplete shell cluster. We need to adopt a definition of equivalence that yields a value of circumference (surface area) of the complete circle (sphere) that is as close to the exact arc length (surface area) of the shell cluster as possible. We will henceforth refer to the radius of the equivalent circle (sphere) as the effective radius. Obviously, the definition of equivalence must involve the "span" of the shell cluster under consideration. Since the covariance matrix $\mathbf{C}$ (not the shell covariance matrix $\mathbf{C}_S$) of a cluster measures the "span" of the cluster, $\mathbf{C}$ is an obvious choice to be used in the definition of equivalence. The information contained in $\mathbf{C}$ needs to be converted to a scalar.

For a complete circle with radius $r$, the exact arc length $L$ is $2\pi r$, and the covariance matrix is given by

$$\mathbf{C} = \begin{bmatrix} \frac{r^2}{2} & 0 \\ 0 & \frac{r^2}{2} \end{bmatrix}.$$

Hence, $\mathrm{Tr}\ \mathbf{C} = r^2$. This suggests that the effective radius $r_{\text{eff}}$ should be defined in $n$-dimension as $\sqrt{\mathrm{Tr}\ \mathbf{C}}$ to yield an exact estimate of the arc length $2\pi r_{\text{eff}} = 2\pi r$. On the other hand, $\det \mathbf{C} = r^4/4$, which suggests using $r_{\text{eff}} = [n^n \det \mathbf{C}]^{1/2n}$, where $n$ is the dimensionality of the data set. This definition has the disadvantage of giving a zero effective radius for straight lines for which $\det \mathbf{C} = 0$. Therefore, this is not a good choice.

In the two-dimensional case, the surface density $\delta_i$ of cluster $\beta_i$ is defined as the number of points per unit estimated arc length, i.e.,

$$\delta_i = \frac{S_i}{2\pi r_{\text{eff}_i}} \tag{9}$$

where $S_i$ is the "sum of central members" defined as

$$S_i = \sum_j u_{ij} \text{ sum over } j \text{ such that } \|\tau_{ij}\| < \tau_{\max}. \tag{10}$$

In the above equation, $\|\tau_{ij}\|$ is the distance from the point to the shell. In other words, we count only those points that lie within a distance $\tau_{\max}$ from the shell. A good value for $\tau_{\max}$ is $\sqrt{\eta_i}$, where $\eta_i$ is the bandwidth used in possibilistic clustering (see Section VIII of Part I of this paper). In (9), $r_{\text{eff}}$ is the effective radius of shell cluster $\beta_i$ given by

$$r_{\text{eff}_i} = \sqrt{\mathrm{Tr}\ \mathbf{C}_i} \tag{11}$$

where $\mathbf{C}_i$ is the fuzzy covariance matrix of cluster $\beta_i$ given in (1).

From the definition in (11), it is easy to show that the equivalent circle with radius $r_{\text{eff}_i}$ has the same second moment as the shell cluster under consideration. Therefore, a geometric interpretation of the definition of equivalence is established. It is to be noted that $2\pi r_{\text{eff}_i}$ is an estimate of the exact arc length of the cluster, since the latter cannot be computed easily for shell clusters that are sparse, scattered or partial. It is obvious that if the original cluster of radius $r_i$ is partial, then $r_{\text{eff}_i}$ and $r_i$ will not be the same.

In the three-dimensional case, we define the surface density $\delta_i$ of a cluster $\beta_i$ as follows

$$\delta_i = \frac{S_i}{4\pi r_{\text{eff}_i}^2} \tag{12}$$

where $S_i$ is given by (10), and $r_{\text{eff}_i}$ is given by (11). In this case, $\delta_i$ measures the number of points per unit surface area of shell cluster $\beta_i$. The sum of the surface densities of all clusters may be used as a global performance measure to evaluate an overall $C$-shell-partition.

In the remaining part of this section, we derive the theoretical values of the surface density validity measure for some common shell types to study the discrepancies between the desired and actual values of this measure.

### *B. Surface Density for Circles*

Consider a circular arc of radius $r$ shown in Fig. 4 subtending an angle $\gamma$. The covariance matrix of this arc is given by

$$\mathbf{C} = \frac{1}{L_\gamma} \int_{-\gamma/2}^{\gamma/2} \mathbf{x}\mathbf{x}^{\mathrm{T}}\, dl - \mathbf{m}\mathbf{m}^T \tag{13}$$

where $\mathbf{x} = [r\cos\theta, r\sin\theta]^T$ is a point on the arc, $dl$ is the elemental arc length given by $dl = r\, d\theta$

$$L_\gamma = \int_{-\gamma/2}^{\gamma/2} dl = r\gamma$$

is the arc length, and $\mathbf{m}$ is the mean (centroid) of the data points given by

$$\mathbf{m} = [m_x, m_y]^T = \frac{1}{L_\gamma} \int_{-\gamma/2}^{\gamma/2} x\, dl = \left[\frac{2r\sin(\gamma/2)}{\gamma}, 0\right]^T.$$

Hence

$$\mathbf{C} = \frac{r^2}{2} \begin{bmatrix} 1 - \frac{\sin\gamma}{\gamma} - \frac{8\sin^2(\gamma/2)}{\gamma^2} & 0 \\ 0 & 1 + \frac{\sin\gamma}{\gamma} \end{bmatrix}.$$

Thus, the effective radius $r_{\text{eff}}$ is given by

$$r_{\text{eff}} = \sqrt{\mathrm{Tr}\ \mathbf{C}} = r\sqrt{1 - \frac{4\sin^2(\gamma/2)}{\gamma^2}}.$$

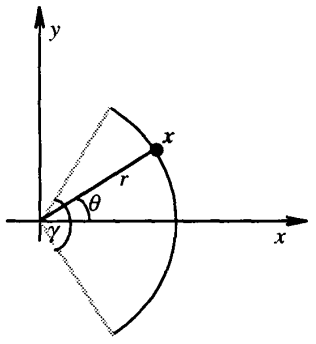



Fig. 4. Segment of a circular shell.

Therefore, the theoretical value of surface density $\delta$ (assuming the arc is perfect and continuous) is

$$\delta = \frac{L_\gamma}{2\pi r_{\text{eff}}} = \frac{\gamma}{2\pi\sqrt{1 - \frac{4\sin^2(\gamma/2)}{\gamma^2}}}. \tag{14}$$

It can be seen from (14) that the surface density criterion is independent of size (radius), and for a complete circle, it reaches its upper bound of one. This sets an absolute value for surface density against which to compare the validities of individual clusters.

### C. Surface Density for Lines

For a line of length $L$, the variance = Tr C is given by

$$\text{Tr } \mathbf{C} = \frac{L^2}{12}.$$

Hence $r_{\text{eff}} = \frac{L}{2\sqrt{3}}$, meaning that the theoretical value of surface density $\delta$ is

$$\delta = \frac{L}{2\pi r_{\text{eff}}} = \frac{\sqrt{3}}{\pi} = 0.55. \tag{15}$$

Since we deal with mixtures of lines and quadrics in this paper, we need to have a validity measure that works for both cases; however, the theoretical value for a perfect line is rather low. We now discuss this problem in more detail.

### D. Compensation for Circular Arcs and Linear Segments

From the theoretical values of surface density for circles and lines obtained in Sections III-B and III-C, we can construct Table III which shows the values of surface density for various values of $\frac{r_{\text{eff}}}{r}$. In Table III, lines are considered as circles with an infinite radius. Hence $\frac{r_{\text{eff}}}{r} = 0$ for lines. As can be seen from Table III, the values of surface density drop from the ideal value of 1.0 for partial circles and lines. In this section we will introduce compensation factors that can be used to make surface density invariant to partiality.

It can be seen from Table III that $\frac{r_{\text{eff}}}{r}$ can be used as a measure of partiality for circular clusters. It is to be noted that $r_{\text{eff}}$ and $r$ can be computed from the prototype parameters and the covariance matrix of the cluster before the computation of cluster validity. Our goal is to find a scaling factor $f_P$ to compensate for partiality effects, which when multiplied by

TABLE III
RATIOS OF EFFECTIVE RADII TO RADII AND SURFACE DENSITY FOR SOME TYPICAL CIRCULAR CLUSTERS

| Cluster type | Complete circle $\phi = 2\pi$ | Half circle $\phi = \pi$ | Quarter circle $\phi = \frac{\pi}{2}$ | Line |
|---|---|---|---|---|
| $\frac{r_{\text{eff}}}{r}$ | 1.0 | 0.77 | 0.44 | 0.00 |
| surface density | 1.0 | 0.65 | 0.57 | 0.55 |

TABLE IV
THEORETICAL VALUES OF SURFACE DENSITY AND COMPENSATION FACTORS FOR SOME TYPICAL CIRCULAR CLUSTERS AND A LINE

| Cluster type | theoretical surface density | compensation factor |
|---|---|---|
| Complete circle | 1.0 | 1.0 |
| Semi-circle | 0.65 | 1.54 |
| Quarter circle | 0.57 | 1.74 |
| Line | 0.55 | 1.81 |

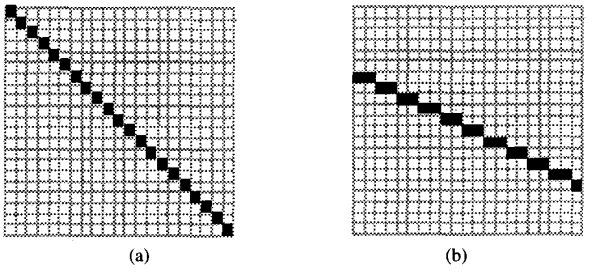
the raw surface density value as given by (14) or (15), will bring it back to the ideal value. This is easily accomplished by choosing $f_P$ to be the reciprocal of the theoretical value of the surface density for the particular partial shell cluster. Thus, for the case of linear clusters (which can be identified from their prototype parameters), (15) gives us a scaling factor $f_P$ of $\frac{\pi}{\sqrt{3}}$. For the three types of circular arcs listed in Table IV, the surface density should be scaled by the corresponding $f_P$ given in the table. For other partial circles, we can obtain the scaling factor as a function of $\frac{r_{\text{eff}}}{r}$ by linearly interpolating between the values listed in the table. This gives us the following function for the scaling factor $f_P$.

$$f_P\left(\frac{r_{\text{eff}}}{r}\right) = \begin{cases} 1.74 - 0.172\left(\frac{r_{\text{eff}}}{r} - 0.44\right) & \text{for } \frac{r_{\text{eff}}}{r} \in [0, 0.44] \\ 1.54 - 0.594\left(\frac{r_{\text{eff}}}{r} - 0.77\right) & \text{for } \frac{r_{\text{eff}}}{r} \in [0.44, 0.77] \\ 1.0 - 2.34\left(\frac{r_{\text{eff}}}{r} - 1.0\right) & \text{for } \frac{r_{\text{eff}}}{r} \in [0.77, 1.0]. \end{cases} \tag{16}$$

Thus, the value of $\frac{r_{\text{eff}}}{r}$ is first computed for every cluster, and the raw surface density is then scaled by the factor given in (16).

### E. Effects of Digitization on the Sum of Central Members

*Effect of Digitization on Line Length:* We assume that the images are always thinned before our clustering algorithms are applied. In digital pictures, lines suffer from the effects of digitization. To see the effect of digitization on the value of the sum of central members $S_i$ in the numerator of (9), consider the case of a 45 degrees line and a 150 degrees line, as shown in Figs. 5(a) and 5(b). We can see that for a perfect (8-connected) digital line, the number of pixels on the line is not equal to the length of the line. Rather, it is equal to some factor $f_D$ times the length. In other words, $L = f_D \times$ number of pixels, where $f_D$ is a factor greater than 1. In this particular



(a) (b)

Fig. 5. (a) A 45 degrees digital line (b) A 150 degrees digital line.

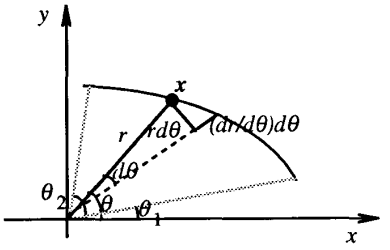



Fig. 6. Segment of an elliptic shell.

case, $f_D = \sqrt{2}$ for the 45 degrees line and $f_D = \frac{2}{\sqrt{3}}$ for the 150 degrees line. It is only for horizontal and vertical lines that the estimate is exact, i.e., $f_D = 1$. In general, the sum of central members must be scaled by a factor $f_D$ to compensate for digitization effects, where

$$f_D = \frac{1}{\max(|\cos\theta|, |\sin\theta|)}$$

and $\theta$ is the angle that the line makes with the $x$-axis. Therefore the final expression for the surface density of a 2-D linear cluster is

$$\delta_i = \frac{S_i}{2\pi r_{\text{eff}_i}} \frac{f_P}{\max(|\cos\theta_i|, |\sin\theta_i|)}. \tag{17}$$

*Effect of Digitization on Arc Length:* In digital pictures, circular arcs are also distorted due to digitization. This will cause the arc length, and hence the surface density $\delta_i$ to be underestimated. We may assume that a circular arc can be approximated by a large number of line segments. In this case, every pixel on the arc should contribute to the sum of central members not by one unit, but by a factor $f_D$ as in the case of lines. This factor depends on the orientation of the tangent to the arc at a given pixel location. Hence, the membership contribution of each point $x_j$ in cluster $\beta_i$ to the sum of central members $S_i$ should be scaled by a digitization factor given by

$$f_D = \frac{1}{\max(|\cos\theta_{ij}|, |\sin\theta_{ij}|)}$$

where $\theta_{ij}$ is the angle that the tangent to the arc at point $\mathbf{x}_j$ makes with the $x$-axis. Taking this into account, the surface density $\delta_i$ for a circular cluster $\beta_i$ may be computed as

$$\delta_i = \frac{S_i'}{2\pi r_{\text{eff}_i}} f_P \tag{18}$$

where

$$S_i' = \sum_j \frac{u_{ij}}{\max(|\cos\theta_{ij}|, |\sin\theta_{ij}|)},$$

sum over $j$ such that $\|\tau_{ij}\| < \tau_{\max}$.

Equation (18) reduces to (17) for the case of lines, since all $\theta_{ij}$ are equal. Examples of compensated surface density values are given in Section III-J.

### F. Surface Density for Ellipses

Consider an elliptical arc of major diameter $2a$, and minor diameter $2b$ that extends from $\theta = \theta_1$ to $\theta = \theta_2$, as shown in Fig. 6. The covariance matrix $\mathbf{C}$ of this elliptical segment is given by (13) where

$$dl = \sqrt{(rd\theta)^2 + (dr)^2} = \sqrt{r^2 + (dr/d\theta)^2}\, d\theta$$

$r$ being the radial distance of the point $\mathbf{x} = [r\cos\theta, r\sin\theta]^{\mathrm{T}}$ from the center. It is easily verified that

$$r = \frac{ab}{\sqrt{b^2\cos^2\theta + a^2\sin^2\theta}}.$$

For the case of ellipses, the arc length $L_{\theta_{12}}$ is given by

$$L_{\theta_{12}} = \int_{\theta_1}^{\theta_2} dl = \int_{\theta_1}^{\theta_2} \sqrt{r^2 + (dr/d\theta)^2}\, d\theta \tag{19}$$

and the mean $\mathbf{m} = [m_x, m_y]^T$ is given by

$$m_x = \frac{1}{L_{\theta_{12}}} \int_{\theta_1}^{\theta_2} r\cos\theta \sqrt{r^2 + (dr/d\theta)^2}\, d\theta \tag{20}$$

and

$$m_y = \frac{1}{L_{\theta_{12}}} \int_{\theta_1}^{\theta_2} r\sin\theta \sqrt{r^2 + (dr/d\theta)^2}\, d\theta. \tag{21}$$

Hence

$$\text{Tr } \mathbf{C} = \frac{1}{L_{\theta_{12}}} \int_{\theta_1}^{\theta_2} r^2 \sqrt{r^2 + (dr/d\theta)^2}\, d\theta - (m_x^2 + m_y^2). \tag{22}$$

The integrals in (19)–(22) can be numerically computed to obtain $\text{Tr } \mathbf{C} = (r_{\text{eff}})^2$ and the surface density $\delta$ can then be computed as $L_{\theta_{12}}/2\pi r_{\text{eff}}$. Values of the effective radius $r_{\text{eff}}$ and surface density $\delta_i$ obtained by performing numerical integrations for some values of $a, b, \theta_1$, and $\theta_2$ are listed in Table V. It can be seen that surface density does not vary very much for large variations in the partiality of the ellipse as well as its elongatedness. As in the case of circles, it is possible to compensate for the small variations; however, this will not be discussed in this paper. In practice, the uncompensated densities can be used with no serious problems.

### G. Surface Density for Planes

For a plane with dimensions $L \times M$

$$r_{\text{eff}^2} = \text{Tr } \mathbf{C} = \frac{L^2 + M^2}{12}$$

TABLE V
THEORETICAL VALUES OF EFFECTIVE RADIUS AND SURFACE DENSITY FOR SOME TYPICAL ELLIPTICAL CLUSTERS

| Ellipse type | major axis to minor axis ratio | effective radius | surface density |
|---|---|---|---|
| Complete | 2 | 1.50 | 1.03 |
| " | 5 | 3.04 | 1.08 |
| " | 10 | 5.90 | 1.09 |
| Half (symmetric with respect to major axis) | 2 | 0.98 | 0.79 |
| " | 5 | 1.69 | 0.99 |
| " | 10 | 3.03 | 1.07 |
| Half (symmetric with respect to minor axis) | 2 | 1.33 | 0.58 |
| " | 5 | 3.01 | 0.56 |
| " | 10 | 5.85 | 0.55 |
| Quarter (symmetric with respect to major axis) | 2 | 0.58 | 0.66 |
| " | 5 | 0.94 | 0.89 |
| " | 10 | 1.58 | 0.98 |
| Quarter (one quadrant) | 2 | 0.68 | 0.56 |
| " | 5 | 1.51 | 0.55 |
| " | 10 | 2.93 | 0.55 |

and the surface area is $LM$. Hence, the surface density

$$\delta = \frac{3LM}{\pi(L^2 + M^2)}. \tag{23}$$

For the special case of a square plane ($L = M$), we get $\delta = \frac{3}{2\pi} = 0.48$.

### *H. Compensation for Planes*

Since the theoretical value of surface density for planes is $3/2\pi$, we should compensate it by a factor $f_P = 2\pi/3$. In addition, we have to account for the effect of foreshortening which decreases the visible area of the plane and hence the count of the number of central members for the plane. We can expect a discrepancy whenever the foreshortened surface area (as seen in the image) differs from the actual surface area of the plane. This happens whenever the planar surface is not parallel to the image plane. Thus, the compensated surface density becomes

$$\delta_i = \frac{S_i}{4\pi r_{\text{eff}_i^2}} \frac{f_P}{|\cos\theta_i|}$$

where $\theta_i$ is the angle between the normal to the plane and the viewing direction ($z$-axis). For a planar cluster with a surface normal $\mathbf{n}_i = [n_{i1}, n_{i2}, n_{i3}]^{\mathrm{T}}$

$$\cos\theta_i = \frac{n_{i3}}{\sqrt{n_{i1}^2 + n_{i2}^2 + n_{i3}^2}}.$$

TABLE VI
THEORETICAL VALUES OF SURFACE DENSITY FOR SOME TYPICAL SPHERICAL SHELL CLUSTERS. HERE $\theta$ IS THE AZIMUTH ANGLE AND $\phi$ IS THE ELEVATION ANGLE OF THE SPHERICAL PATCH

| Shell-type | unnormalized surface density | normalized surface density |
|---|---|---|
| Complete sphere $0 \le \phi \le \pi, 0 \le \theta \le 2\pi$ | 1.00 | - |
| Hemisphere $0 \le \phi \le \pi/2, 0 \le \theta \le 2\pi$ | 0.67 | 1.00 |
| Partial sphere $0 \le \phi \le \pi/4, 0 \le \theta \le 2\pi$ | 0.54 | 0.81 |
| Quarter sphere $0 \le \phi \le \pi/2, 0 \le \theta \le \pi$ | 0.50 | 0.75 |

### *I. Surface Density for Spheres and Ellipsoids*

Expressions for the surface density of spherical and ellipsoidal patches can also be derived. Appendix A gives the details. In practice, the most we can see of a sphere in a range image is half of its surface area. Therefore, we must normalize the surface density for spherical patches by dividing it by the value $f_N = 2/3$ which is obtained for hemispheres. Table VI lists unnormalized and normalized surface density values for specific cases. As seen in Table VI, partiality does have an effect on the values of surface density for spherical shells. One can devise a scheme similar to that of the one for circles to compensate for this variation, although this is not required in most practical applications. We do need to account for the effect of foreshortening in 3-D, however, which causes the estimation of the sum of central members to be less than the actual value. We may approximate a spherical surface at each pixel location by a planar surface, and compensate for foreshortening by scaling the contribution of the pixel by a factor similar to the one used for planes. This leads to

$$\delta_i = \frac{S_i'}{4\pi r_{\text{eff}^2}} f_N \tag{24}$$

where

$$S_i' = \sum_j \frac{u_{ij}}{|\cos\phi_{ij}|} \quad \text{sum over } j \text{ such that } \|\tau_{ij}\| < \tau_{\max}$$

where $\phi_{ij}$ is the angle between the surface normal at $\mathbf{x}_j = [x_{j1}, x_{j2}, x_{j3}]^{\mathrm{T}}$ and the viewing direction ($z$-axis) given by

$$\cos\phi_{ij} = \frac{x_{j3} - c_{i3}}{\|\mathbf{x}_j - \mathbf{c}_i\|}.$$

Some theoretical values for surface density for ellipsoidal patches found by numerical integration (using the expressions derived in Appendix A) are listed in Table VII. It can be seen that the surface density does not vary significantly for large variations in the ratios of the diameters of the ellipsoid as well as the partiality of the ellipsoid. Compensation for ellipsoidal shells will not be discussed in this paper.

TABLE VII
VALUES OF SURFACE DENSITY FOR SOME TYPICAL ELLIPSOIDAL SHELL CLUSTERS. HERE $\theta$ IS THE AZIMUTH ANGLE AND $\phi$ IS THE ELEVATION ANGLE OF THE ELLIPSOIDAL PATCH

| Shell-type | $\frac{b}{a}$ | $\frac{c}{a}$ | surface density |
|---|---|---|---|
| Half ellipsoid $0 \le \phi \le \pi/2, 0 \le \theta \le 2\pi$ | 1 | 2 | 0.83 |
| " | 2 | 5 | 0.86 |
| Partial ellipsoid $0 \le \phi \le \pi/4, 0 \le \theta \le 2\pi$ | 1 | 2 | 0.72 |
| " | 2 | 5 | 0.84 |
| Partial ellipsoid $0 \le \phi \le \pi/2, 0 \le \theta \le \pi$ | 1 | 2 | 0.57 |
| " | 2 | 5 | 0.58 |

TABLE VIII
VALUES OF SURFACE DENSITY FOR THE CLUSTERS IN FIG. 3

| Cluster | uncompensated surface density | compensated surface density |
|---|---|---|
| Large circle | 0.91 | 1.01 |
| Small circle | 0.93 | 1.04 |
| Large half circle | 0.60 | 1.00 |
| Sparse circle | 0.29 | 0.32 |

### J. Examples of Surface Density

The surface density values for the circular clusters in Fig. 3 and the elliptical and circular clusters in Fig. 2(c) are listed in Tables VIII and IX. It can be seen in Table VIII that the value of the compensated surface density as given by (18) is approximately the same for the small and large circles. Also, it is almost identical for the complete circle and for the half-circle; however, surface density does vary with the sparseness of the cluster. Since the sparse cluster has only about a third of the points of the complete big circle, its surface density is also roughly 1/3. As mentioned in Section II-B, however, in most computer vision applications, this does not pose a serious problem. It can also be seen from Table IX that in the case of Fig. 2(c), surface density is high and close to 1.0 for all the good clusters, and low for the spurious cluster. Thus, it would be easy to devise an unsupervised algorithm to find the optimum number of clusters using this validity measure. This algorithm would initially use a large number of clusters and then progressively merge "good" compatible clusters and eliminate "bad" (spurious) clusters. Here, we would like to note that using the possibilistic versions (see Part I of this paper) of our shell clustering algorithms yields high validities for all three identical clusters, whereas fuzzy membership values would have caused an underestimation of the sum of central members of each cluster because the membership values would have been about 1/3. Thus, in the fuzzy case, all three identical clusters could be considered spurious.

TABLE IX
VALUES OF SURFACE DENSITY FOR THE CLUSTERS IN FIG. 2(C)

| Cluster | uncompensated surface density |
|---|---|
| Large ellipse 1 | 1.14 |
| Large ellipse 2 | 1.10 |
| Large ellipse 3 | 1.11 |
| Circle 1 | 0.96 |
| Circle 2 | 0.93 |
| Spurious ellipse | 0.38 |

In the next section, we will describe unsupervised algorithms for boundary detection and surface approximation using the validity measures discussed in this section. In developing these algorithms, we used the possibilistic versions (PCQS and PCPQS) rather than the fuzzy versions (FCQS and FCPQS) of the shell clustering algorithms introduced in Part I of this paper. The reason for this choice was to make the algorithms more robust.

## IV. THE UNSUPERVISED BOUNDARY DETECTION ALGORITHM

In this section, we describe an unsupervised clustering algorithm which can be used to detect an unknown number of second-degree boundary curves (including the degenerate case of lines) and estimate their parameters. Our unsupervised clustering algorithm works by progressively clustering the data starting with an overspecified number $C_{\max}$ of clusters. Initially, the PCQS algorithm is run with $C = C_{\max}$, followed by the Line Detection algorithm (see Part I of this paper). At this stage, spurious clusters are eliminated, compatible clusters are merged, good clusters are identified, and points with high memberships in good clusters are temporarily removed from the data set to reduce the complexity of the remaining data set. This also reduces the computational burden. The PCQS algorithm is invoked again with the remaining feature points. This procedure is repeated until no more elimination, merging, or removing occurs, or until $C = 1$. A more detailed discussion of the steps is described next.

### A. Elimination of Spurious Clusters

Cluster $\beta_i$ is considered spurious if the sum of the memberships of all feature points in that cluster is very low or if the sum is low and the surface density is also low, i.e.,

$$\sum_{j=1}^{N} u_{ij} < N_{VL}, \text{ or } \sum_{j=1}^{N} u_{ij} < N_L \quad \text{and } \delta_i < \delta_L.$$

If the values of $N_{VL}, N_L$, and $\delta_L$ are too large, many clusters will be eliminated and there will not be enough clusters left to obtain the correct final partition. If they are too small, the unsupervised algorithm may take a long time to converge. In

our experience, a good choice for $N_{VL}$ is about 2% of the total number of data points and for $N_L$ it is about 4% of the total number of points. A good value for $\delta_L$ in our application was about 0.15, although any value below 0.4 seems to work quite well.

### *B. Merging Compatible Clusters*

One way to determine if two clusters are compatible is to estimate the error of fit and the validity for the merged cluster. To do this, all points having a membership greater than an $\alpha$-cut in either one of the two clusters are collected. Let this data set be denoted by $X_{ij}$). A value of about 0.25 works best for $\alpha$ in practice. Then the PCQS algorithm is run with $C = 1$ on this data set. If the fit and surface density for the resulting cluster are good, the two clusters are merged. In other words, the condition for merging is

$$T_i < T_L \quad \text{and} \quad \delta_i > \delta_H$$

where $T_i$ and $\delta_i$ are the average shell thickness and the surface density of the cluster formed by $X_{ij}$. Suitable values for $T_L$ and $\delta_H$ for this application are about 2.0 and 0.7.

The merging procedure described above is only valid for clusters with second-degree prototypes. Two linear clusters $\beta_i$ and $\beta_j$ are considered compatible if they satisfy the conditions used in the Compatible Cluster Merging (CCM) algorithm [23] (see next section). Combinations of linear and second-degree clusters are not considered in the merging step.

For the special case of data sets with linear and circular clusters, the merging process can be simplified. Note that hyperspheres are described by $\mathbf{p}_i^{\mathrm{T}}\mathbf{q} = [p_{i1}, p_{i2}, \cdots, p_{i(n+1)}, p_{i(n+2)}]\,[(x_1^2 + x_2^2 + x_3^2 + \cdots + x_n^2), x_1, \cdots, x_n, 1]^{\mathrm{T}} = 0$. Two circular shell clusters $\beta_i$ and $\beta_j$ are considered compatible if they satisfy the following conditions

$$\|\mathbf{c}_i - \mathbf{c}_j\| < \varepsilon_1 \quad \text{and} \quad |r_i - r_j| < \varepsilon_2$$

where $\mathbf{c}_i$ and $\mathbf{c}_j$ are the centers, and $r_i$ and $r_j$ are the radii of the two clusters under consideration. These values can be computed from the parameter vectors $\mathbf{p}_i$ and $\mathbf{p}_j$. Suitable values for $\varepsilon_1$ and $\varepsilon_2$ are about 1.0. A cluster is determined to be linear/planar if its prototype $\mathbf{p}_i$ satisfies the following condition

$$\frac{|p_{i1}|}{\sum_{r=2}^{n+1} |p_{ir}|} \leq \varepsilon_L$$

where $\varepsilon_L$ is a suitable threshold. Typically $\varepsilon_L = 0.0001$. In this case, the Possibilistic $C$ Plano-Spherical Shells algorithm (Section VI, Part I) is used instead of the PCQS algorithm.

### *C. Identification of Good Clusters and Removal of Feature Points*

Cluster $\beta_i$ is characterized as good if

$$\delta_i > \delta_{VH} \quad \text{or} \quad \delta_i > \delta_H \quad \text{and} \quad V_i < V_L,$$

where $\delta_{VH}$ is a very high threshold for surface density, $\delta_H$ is the same value that was used for merging, and $V_L$ is a low value for the fuzzy hypervolume. It is to be noted that the second condition is designed to handle cases in which the surface density has borderline values. Suitable values for $\delta_{VH}$ and $V_L$ are about 0.85 and 0.5 in our application. Points are temporarily removed from the data set if their membership in one of the good clusters is greater than $u_H = 0.5$.

In addition to the above steps, we need to identify noise points and temporarily remove them from the data set . Noise points are identified as those which have low memberships in all clusters, i.e., we remove feature point $\mathbf{x}_k$ if

$$u_{ik} < u_L \quad \text{for } 1 \leq i \leq C.$$

Noise points have to be removed at the end of each run the PCQS algorithm, because as the number of clusters decreases and points assigned to good clusters are removed, the number of noise points relative to the good points becomes too high, making it difficult to detect the few good clusters that are left. A good choice for $u_L$ is about 0.1. Note that the condition for noise point removal can be used only when possibilistic memberships are used, and not when fuzzy memberships are used. In the case of fuzzy memberships the above condition can be true even for good points if they are shared among many clusters. A similar comment applies to the removal of good points.

The threshold values mentioned in this section are meant only as a guideline, and the actual choice depends on the application. However, in our experience, the exact choice of thresholds is not crucial for a given application. A range of values produce the same final result, although the order in which particular clusters are eliminated, merged, or removed in the intermediate stages may vary.

---

*THE UNSUPERVISED BOUNDARY DETECTION ALGORITHM*

*Set* $C = C_{\max}$; *Fix* $m$, $m \in [1, \infty)$;
*REPEAT*
  *Perform clustering using the PCQS algorithm*;
  *Run the Line Detection Algorithm*;
  *Eliminate spurious clusters and update* $C$,
  *Merge compatible prototypes and update* $C$;
  *Detect good clusters, save their prototypes in a list, remove points with high memberships in them and update* $C$;
  *Remove noise points*;
*UNTIL* (*No elimination or merger or removal takes place*);
*Replace all the removed feature points back into the data set*;
*Append remaining clusters' prototypes from the last iterationin the above repeat loop to the list of removed clusters' prototypes and update* $C$;
*REPEAT*
  *Perform the PCQS algorithm and the Line Detection Algorithm using the prototype list as initialization*;
  *Merge compatible prototypes and update* $C$ *accordingly*;
  *Eliminate tiny clusters and update* $C$ *accordingly*;
*UNTIL* (*No more merging or elimination takes place*);

---

The unsupervised boundary detection algorithm is summarized below. The first repeat loop typically requires three passes and the second one or two passes. The passes become progressively faster because the number of clusters is reduced and the points belonging to the good clusters are removed in each pass. The second repeat loop is used to fine-tune the results. In this loop, all clusters with a sum of memberships less than $N_L$ are considered tiny.

## V. The Unsupervised Quadric Surface Fitting Algorithm

The unsupervised algorithm presented in the previous section does not perform as well in 3-D. One reason is that the solution space is more complex, making convergence to local minima more probable. Another problem that arises when the algorithm in Section IV is directly applied to 3-D is that most quadric surfaces such as cones, cylinders, and planes have an infinite extent, i.e., they are not bounded surfaces. Therefore, if a cluster having the prototype of one of these surfaces is identified as good and the points assigned to it are removed, many other points which belong to other clusters may be removed because they lie on the extension of this good cluster. This may cause them to become partial and disconnected, and hence they become more difficult to detect. Another difference in the 3-D case is that, since computing the perpendicular distance is expensive, the PCPQS algorithm needs to be used instead of the PCQS algorithm. For these reasons, we developed an alternative unsupervised algorithm for the 3-D case. This algorithm is a generalization of the CCM algorithm [21]. The CCM algorithm is first briefly summarized in Section V-A, then its generalization is discussed in Section V-B.

### A. The CCM Algorithm

The CCM algorithm produces a linear or planar approximation of a data set. Initially, the Gustafson–Kessel (G-K) algorithm (see [21] for details) is run with the number of clusters $C$ set to $C_{\max}$ which is higher than the maximum number of clusters that may be expected for the particular problem. After the algorithm converges, the resulting clusters are first bunched into compatible groups such that the compatibility conditions (to be discussed below) are satisfied for every pair of clusters in each group. The clusters in each group are merged, the centers and covariance matrices for the new clusters are calculated, and the G-K algorithm is reinitiated with the new values. This process is repeated until no more mergers can take place.

The compatibility conditions are based on the eigenvalues and eigenvectors of the covariance matrices of the clusters. Let the centers of two clusters $\beta_i$ and $\beta_j$ be $\mathbf{c}_i$ and $\mathbf{c}_j$; the eigenvalues of the covariance matrices of the two clusters be $\{\lambda_{i1}, \cdots, \lambda_{in}\}$ and $\{\lambda_{j1}, \cdots, \lambda_{jn}\}$; and the eigenvectors be $\{\phi_{i1}, \cdots, \phi_{in}\}$ and $\{\phi_{j1}, \cdots, \phi_{jn}\}$. It is assumed that the eigenvalues and vectors are arranged in descending order. The compatibility conditions are given below

$$|\phi_{in} \cdot \phi_{jn}| \geq C_1, C_1 \text{ close to } 1 \tag{25}$$

$$\left| \frac{\phi_{in} + \phi_{jn}}{2} \cdot \hat{\mathbf{c}}_{ij} \right| \leq C_2, C_2 \text{ close to zero}, \tag{26}$$

and

$$\|\mathbf{c}_i - \mathbf{c}_j\| \leq C_3(\sqrt{\lambda_{i1}} + \sqrt{\lambda_{j1}}). \tag{27}$$

In (26), $\hat{\mathbf{c}}_{ij}$ is the unit vector in the direction of $\mathbf{c}_i - \mathbf{c}_j$. In our experience, condition (27) does not work well in certain cases. Therefore, we modified this condition to

$$\|\mathbf{c}_i - \mathbf{c}_j\| \leq C_3\left( \sqrt{\lambda_{il}(\phi_{i1} \cdot \hat{\mathbf{c}}_{ij}) + \cdots + \lambda_{in}(\phi_{in} \cdot \hat{\mathbf{c}}_{ij})} + \sqrt{\lambda_{j1}(\phi_{jl} \cdot \hat{\mathbf{c}}_{ij}) + \cdots + \lambda_{jn}(\phi_{jn} \cdot \hat{\mathbf{c}}_{ij})} \right) \tag{28}$$

In other words, rather than just using the largest eigenvalues of the two clusters, we use a weighted combination of all eigenvalues where the weights are equal to the projections of $\hat{\mathbf{c}}_{ij}$, in the directions of the respective eigenvectors. It is easy to see that in the 2-D case, when conditions (25) and (26) are satisfied, condition (28) reduces to (27). Good choices for the constants $C_1, C_2$ and $C_3$ are 0.95, 0.05 and $\sqrt{3}$, respectively, [23].

### B. The Quadric Compatible Cluster Merging (QCCM) Algorithm

The quadric compatible cluster merging (QCCM) algorithm starts with the initial planar approximation achieved by the CCM algorithm. Then the points belonging to each pair of close clusters (i.e. the points satisfying condition (28)) are used as the input data set to the PCPQS algorithm with $C = 1$. If the fit of the resulting quadric cluster is good, then the two clusters are merged. The goodness of the fit can be verified using the shell thickness measure in (7). Since the exact distance from a feature point to a surface is difficult to compute in the 3-D case, we use the approximate distance while computing this measure. The QCCM algorithm is summarized below.

*THE QUADRIC COMPATIBLE CLUSTER MERGING ALGORITHM*

*merge = TRUE;*
*WHILE (merge = TRUE) DO*
  *merge = FALSE;*
  *FOR each pair of clusters $\beta_i$ and $\beta_j$ DO*
    *IF clusters $\beta_i$ and $\beta_j$ satisfy condition (28) THEN*
      *Let $X_{ij}$ be the set of all feature points having a membership greater than an $\alpha$-cut in cluster $\beta_i$ or cluster $\beta_j$;*
      *Run the* PCPQS *algorithm on $X_{ij}$ with $C = 1$;*
      *Estimate error of fit $T_i$ using (8);*
      *IF ($T_i < T_L$) THEN*
        *$u_{ik} = max(u_{ik}, u_{jk})$ for all $k$;*
        *Eliminate cluster $\beta_j$ and replace the parameters of cluster $\beta_i$ with the parameters of the combined cluster;*
        *$C = C - 1$; merge = TRUE;*
      *END IF;*
  *END IF;*
*END FOR;*
*END WHILE.*

A good value for $T_L$ in our applications was about 0.5. When finding the fit, we use the reweight procedure (see Section VII of Part I of this paper) for better accuracy. In other words, we estimate the fit with and without the reweight and accept the better fit. When the PCPQS algorithm is used inside the QCCM algorithm with $C = 1$, the parameter vector of cluster $\beta_i$ is the solution of a generalized eigenvector problem. As mentioned in Section VI of Part I of this paper, we use a solution that corresponds to an "acceptable" surface type. However, this time we do not include planes and pairs of planes in the list of acceptable types since we assume that we have already obtained the best possible planar approximation by using the CCM algorithm. Hence, no more two planes can be merged into a single plane.

### C. The Unsupervised Quadric Surface Fitting Algorithm

The Unsupervised Quadric Surface Fitting algorithm is summarized below.

*THE UNSUPERVISED SURFACE FITTING ALGORITHM*

$C = C_{\max}$;
*Sample the data set if required*:
*Perform the CCM algorithm*;
*Perform the QCCM algorithm*;
*Run the PCPQS algorithm with all data ponts, using the prototypes obtained at the end of the previous step as initialization, using a distance measure that is a combination of Euclidean and approximate distances.*

The Unsupervised Quadric Surface Fitting algorithm consists of the following steps. The range image is first sampled so that the number of points to be processed is reduced. Initially, the number of clusters $C$ is set to $C_{\max}$ which is two or three times the maximum number of clusters that may be expected for the particular problem. Then a planar approximation of the data is obtained by running the CCM algorithm described in Section V-A. This results in a smaller value for the number of clusters $C$. Then the QCCM algorithm is invoked, and all possible neighboring planes satisfying condition (28) are merged to produce quadric surface fits. This gives us the final number of clusters $C$. The results obtained at this point are fine-tuned further by running the PCPQS algorithm with the final value of $C$, using the prototype parameters obtained at the end of the QCCM algorithm for initialization. In this phase, all feature points are used, and a modified distance, which is a combination of the Euclidean distance and the approximate distance is used. The modified distance is required because in the 3-D case many quadric surfaces are not bounded surfaces (see Section VI of Part I of this paper).

## VI. Experimental Results

In this section we illustrate the effectiveness of the proposed unsupervised algorithms by examples involving several real and synthetic images. Due to space limitation, only a limited number of examples are presented.

### A. Examples of Boundary Detection

Parts (a) of Figs. 7–9 show three 200 × 200 images of objects whose boundaries can be described by linear and second-degree curves. Uniformly distributed noise with an interval of 30 was added to the image intensity values. The object edges were then obtained by applying the Sobel operator and thresholding. The edge images were then thinned [20]. The thinning procedure is important because it makes all edges one-pixel thick. This ensures that the surface density values are always around the theoretically predicted range. The thinning process also reduces the number of pixels to be processed. Parts (b) of Figs. 7–9 show the thinned images which are used as inputs to the Unsupervised Boundary Detection algorithm. It can be seen that the boundaries are not always clean and there are many noise points. The resulting input images typically had about 2000 points. For all the images, the same values for the various thresholds were used. It was also observed that the unsupervised boundary detection algorithm was not very sensitive to the choice of these values.

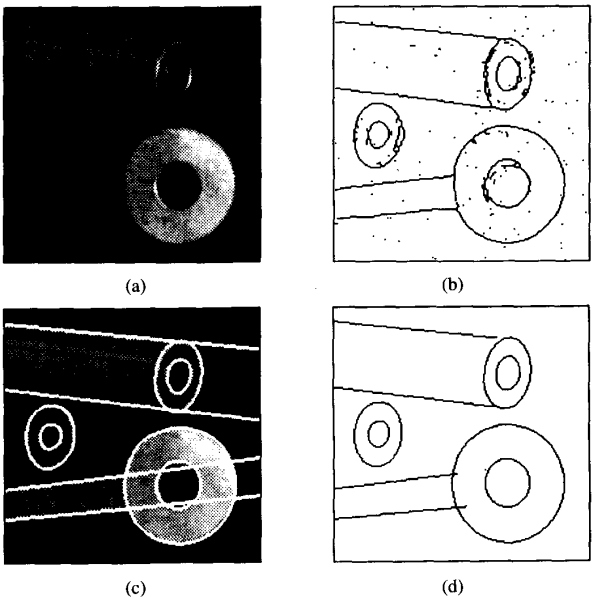
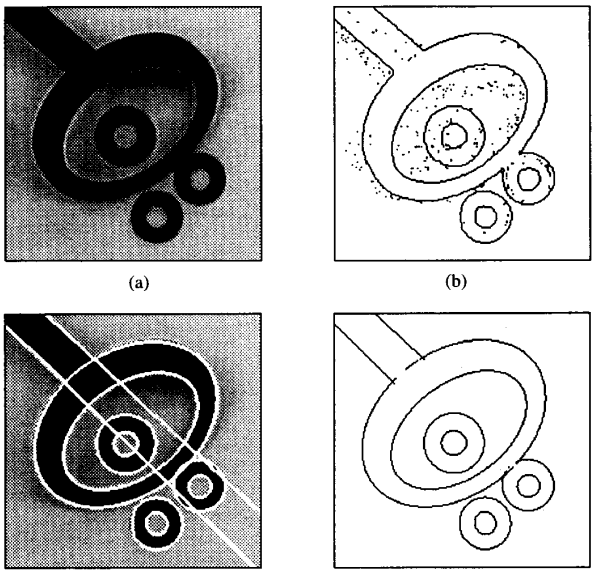



Fig. 7. (a) Original noisy image containing mechanical parts. (b) Edge image with jagged and noisy edges. (c) Prototypes found by the unsupervised boundary detection algorithm superimposed on the original image. (d) Cleaned edge image.

The Unsupervised Boundary Detection algorithm was always applied with the initial number of clusters $C_{\max}$ =25. In the first stage of the PCQS algorithm, the initial values of the bandwidths $\eta_i$ were estimated to be the shell thickness values computed after the FCQS algorithm converged. In the second stage, we fixed $\eta_i$ to 2.0 and ran the algorithm for five more iterations. While computing surface densities the value of $\tau_{\max}$ used was $\sqrt{\eta_i} = \sqrt{2}$.

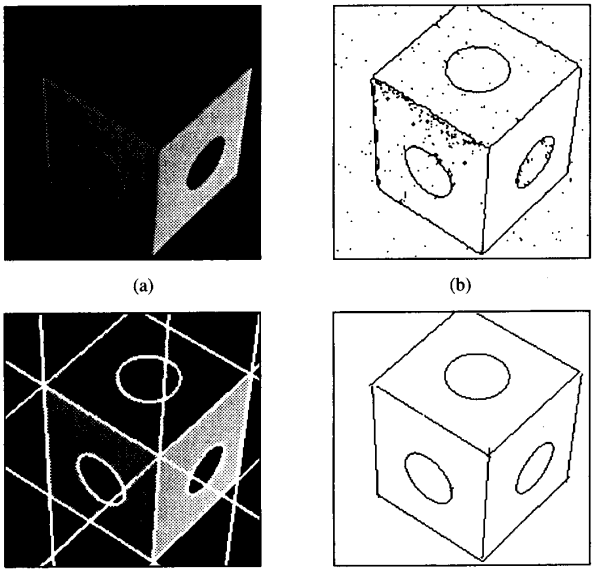
Parts (c) of Figs. 7–9 show the final prototypes superimposed on the images. The prototypes are shown three-pixels thick for emphasis. As can be seen, the results are good in all cases. The "cleaned edge images" in part (d) of Figs. 7–9 were obtained by plotting the prototypes only in those regions where there were at least two edge pixels within a $3 \times 3$ neighborhood.

Fig. 10 shows an example of the special case of linear and circular boundaries. The image of a color filter is shown in



(a) (b) (c) (d)

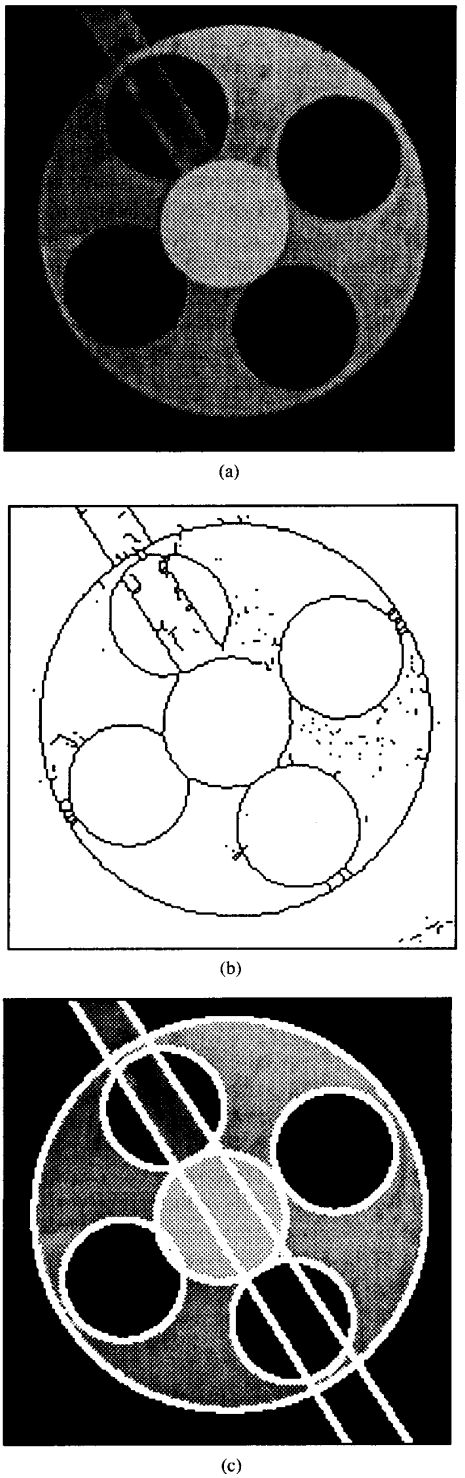
Fig. 8. (a) Original noisy image containing a tube and four rings. (b) Edge image with jagged and noisy edges. (c) Prototypes found by the unsupervised boundary detection algorithm superimposed on the original image. (d) Cleaned edge image.



(a) (b) (c) (d)

Fig. 9. (a) Original noisy image containing a box with holes. (b) Edge image with jagged and noisy edges. (c) Prototypes found by the unsupervised boundary detection algorithm superimposed on the original image. (d) Cleaned edge image.

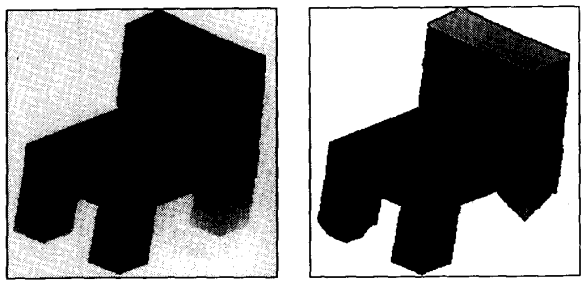
Fig. 10(a). The thinned edge image is shown in Fig. 10(b) where it can be seen that the edges are quite noisy. Fig. 10(c) shows the correct prototypes found by an unsupervised version of the Possibilistic $C$ Plano-Spherical Shells algorithm.



(a) (b) (c)

Fig. 10. (a) Original image of a color filter. (b) Thinned edges of the image in (a). (c) Prototypes obtained from the unsupervised boundary detection algorithm superimposed on the original image.

The CPU time required to run the 2-D unsupervised algorithm on a Sun Sparc 1 workstation was between 30 to 60



(a) (b)

Fig. 11. (a) Original range image of a chair. (b) Surface approximation obtained by the unsupervised surface fitting algorithm.

min. (No attempt was made to optimize the code.) This is reasonable, considering the number of pixels to be processed and complexity of the problem. Since shell clustering algorithms are inherently parallel in nature, the CPU requirements can be considerably reduced in a parallel implementation.

### *B. Examples of Surface Fitting*

The examples used in this section consist of some real and some synthetic range images. A sampling rate of three in the $x$ and $y$ directions was used to reduce computations. This also makes the data sparse, illustrating the fact that the algorithms work for sparse data. The number of feature points after sampling ranged from 2000 to 4000. In all the examples shown in this section, in the CCM algorithm the G-K algorithm was applied with $m = 1.5$ and the initial number of clusters $C_{\max}$ was 15.

Fig. 11(a) shows a real 252× 263 range image of a chair obtained from the University of Southern California. The result of the surface fitting algorithm is displayed in Fig. 11(b) which consists of the correct planar surfaces. Each surface is displayed with a different gray value. Since this image contains only planes, the QCCM algorithm does not merge any planes (because pairs of planes are considered "unacceptable"), and the result of the unsupervised surface fitting algorithm is identical to the result of the CCM algorithm. This example illustrates the fact that the unsupervised surface fitting algorithm can be used without any problems even when there are no quadric surfaces in the image.

Fig. 12(a) shows a 200 × 200 synthetic range image consisting of two planes, a right circular cone, and an ellipsoid. Fig. 12(b) displays the planar approximation for the above mentioned image obtained by using the CCM algorithm. The final results of the unsupervised surface fitting algorithm consisting of the correctly identified surfaces are shown in Fig. 12(c). Fig. 13 shows similar results on a 200 × 200 synthetic range image of a lamp consisting of a cone, a cylinder, and a plane.

The CPU time requirements to run the 3-D unsupervised algorithm on a Sun Sparc 1 workstation were very similar to the 2-D case. This is not surprising because the PCPQS algorithm is run with a known number of clusters (determined in the previous stages) with excellent initialization. Much of the time is in fact spent on obtaining the initial planar approximation. As in the 2-D case, the CPU time can be reduced considerably in a parallel implementation.

## VII. Summary and Conclusion

Although many techniques have been proposed in the literature for the tasks of boundary description and surface approximation, in the case of jagged or scattered edges and noisy or sparse range data, these tasks still remain difficult. This is because region growing techniques cannot be applied when the data is sparse, and features such as gradients and curvatures cannot be computed reliably. In this paper, we have proposed a solution to this problem by combining the ability of shell clustering to deal with scattered and sparse data with the ability of the possibilistic approach to achieve noise immunity. A major disadvantage of clustering methods is that the number of clusters needs to be determined. The proposed approach overcomes this problem by using progressive clustering. The use of cluster elimination and merging avoids the tedious and unreliable alternative of performing clustering for a range of $C$ values.

In order for progressive clustering to work effectively, one needs a set of reliable validity measures. The surface density criterion introduced in this paper to assess the "goodness" of an individual linear or shell cluster is a good supplement to the existing validity measures such as shell thickness, shell density and shell volume. It is especially suitable for the type of computer vision applications considered in this paper. Shell thickness and shell volume cannot always distinguish between good clusters and spurious clusters, and shell density varies very much depending on size and partiality. These validity measures also have about an order of magnitude variation even for good clusters. Therefore, in our experience, by themselves they are not sufficient for the proposed applications. We have shown through derivations that surface density is relatively invariant to size and partiality, and its range can be predicted. This makes it an attractive candidate for use in progressive clustering algorithms. When used in combination with the existing measures, surface density is effective in unsupervised algorithms. We also believe that this validity measure will be quite good for even more complex shapes, as long as the shapes are convex. Some improvements are still possible. For example, our method of estimating the effective radius by using the covariance matrix, though simple to implement, may not be the best way. Moreover, the surface density measure is still not invariant to sparseness. The unsupervised algorithms proposed in this paper to describe boundaries and surfaces in terms of parameterized algebraic forms are particularly suitable for situations in which the data is noisy, scattered or sparse.

The proposed unsupervised boundary detection and surface approximation algorithms can be used in a variety of applications, including object recognition, pose estimation and character recognition. These algorithms can be generalized easily to deal with more complex shells such as those represented by implicit algebraic curves (surfaces) of higher order [20], [30] and superquadrics [29], [33]. We are currently exploring this issue.

## APPENDIX A

### *A. Derivation of Surface Density for Spherical and Ellipsoidal Surface Patches*

Consider a spherical shell patch as shown in Fig. 14. Let the radius of the sphere be $r$. In terms of the parameters $\phi$ and $\theta$, a point $\mathbf{x} = [x, y, z]^T$ on the shell satisfies the following equations

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r\sin\phi\cos\theta \\ r\sin\phi\sin\theta \\ r\cos\phi \end{bmatrix}. \tag{A1}$$

Let us suppose that the patch extends from $\phi_1$ to $\phi_2$ in $\phi$, and from $\theta_1$ to $\theta_2$ in $\theta$. In this case, the covariance matrix is given by

$$\mathbf{C} = \frac{\int_{\theta_1}^{\theta_2}\int_{\phi_1}^{\phi_2} \mathbf{x}\mathbf{x}^{\mathrm{T}}\, dA}{\int_{\theta_1}^{\theta_2}\int_{\phi_1}^{\phi_2} dA} - \mathbf{m}\mathbf{m}^{\mathrm{T}} \tag{A2}$$

where $dA = r^2 \sin\phi\ d\phi d\theta$, is the elemental area of the patch, and

$$\mathbf{m} = \frac{1}{A}\int_{\theta_1}^{\theta_2}\int_{\phi_1}^{\phi_2} \mathbf{x}\, dA \tag{A3}$$

is the centroid of the patch. The denominator of the first term in (A2) represents the surface area A of the shell patch, i.e.,

$$A = \int_{\theta_1}^{\theta_2}\int_{\phi_1}^{\phi_2} dA. \tag{A4}$$

In the case of spherical patches, it is easily verified that

$$A = r^2(\theta_2 - \theta_1)(\cos\phi_1 - \cos\phi_2). \tag{A5}$$

Hence, we may write

$$\mathrm{Tr}\ \mathbf{C} = \frac{1}{A}\int_{\theta_1}^{\theta_2}\int_{\phi_1}^{\phi_2} r^2\, dA - \mathbf{m}^T\mathbf{m} = r^2 - \mathbf{m}^T\mathbf{m}. \tag{A6}$$

For spherical shells, $\mathbf{m} = [m_x, m_y, m_z]^T$ is given by

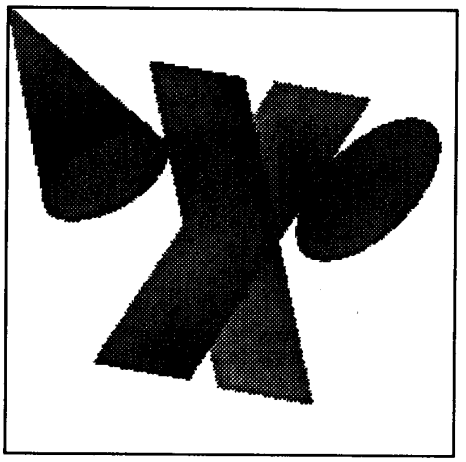
$$\mathbf{m}_x = \frac{r}{2(\theta_2 - \theta_1)}\left(\frac{\sin\theta_2 - \sin\theta_1}{\cos\phi_1 - \cos\phi_2}\right) \times \left((\phi_2 - \phi_1) - \frac{\sin 2\phi_2 - \sin 2\phi_1}{2}\right), \tag{A7}$$

$$\mathbf{m}_y = \frac{r}{2(\theta_2 - \theta_1)}\left(\frac{\cos\theta_1 - \cos\theta_2}{\cos\phi_1 - \cos\phi_2}\right) \times \left((\phi_2 - \phi_1) - \frac{\sin 2\phi_2 - \sin 2\phi_1}{2}\right) \tag{A8}$$

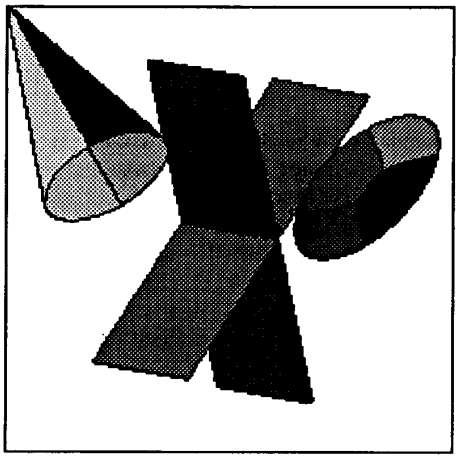
and

$$\mathbf{m}_z = \frac{r}{4}\left(\frac{\cos 2\phi_1 - \cos 2\phi_2}{\cos\phi_1 - \cos\phi_2}\right) \tag{A9}$$

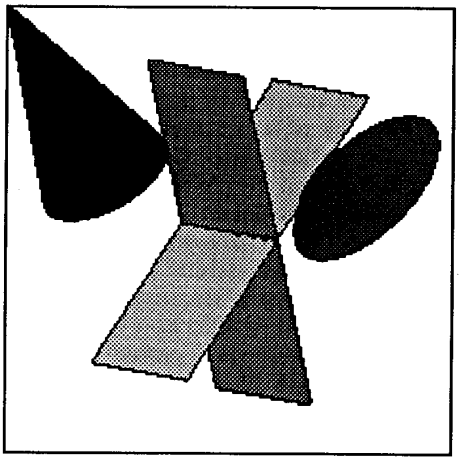
The theoretical values of surface density for various types of spherical shells can be found by using (12), (A5)–(A9).



(a)



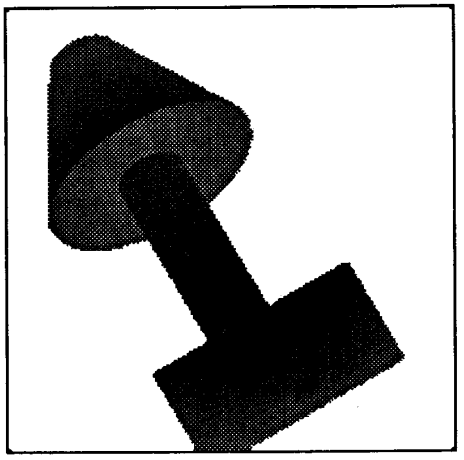
(b)



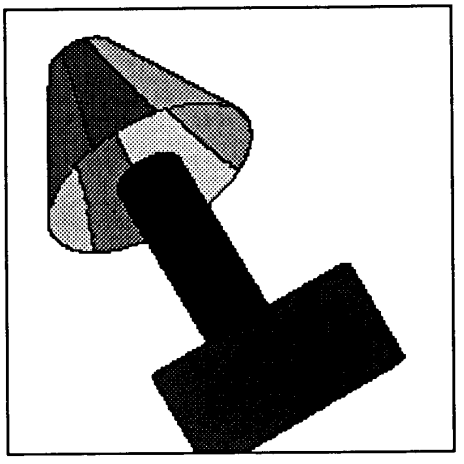
(c)

Fig. 12. (a) Original range image of a cone, two crossing planes and an ellipsoid. (b) Surface approximation obtained by the CCM algorithm. (c) Surface approximation obtained by the unsupervised surface fitting algorithm.

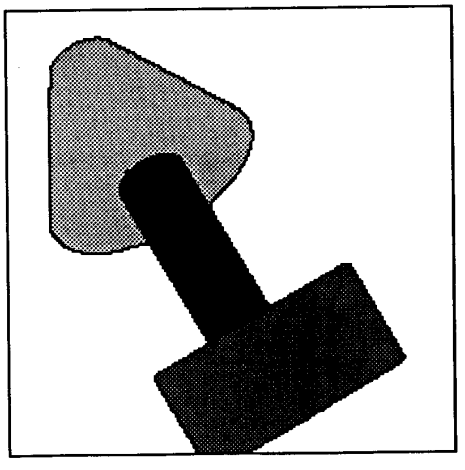
Now consider an ellipsoidal shell patch with axes lengths $= 2a, 2b$ and $2c$ in the $x, y$ and $z$ directions, respectively, as



(a)



(b)



(c)

Fig. 13. (a) Original range image of a lamp. (b) Surface approximation obtained by the CCM algorithm. (c) Surface approximation obtained by the unsupervised surface fitting algorithm.

shown in Fig. 15. In terms of the parameters $\phi$ and $\theta$, a point $\mathbf{x} = [x, y, z]^{\mathrm{T}}$ on the shell satisfies (A1), where $r$ is now the

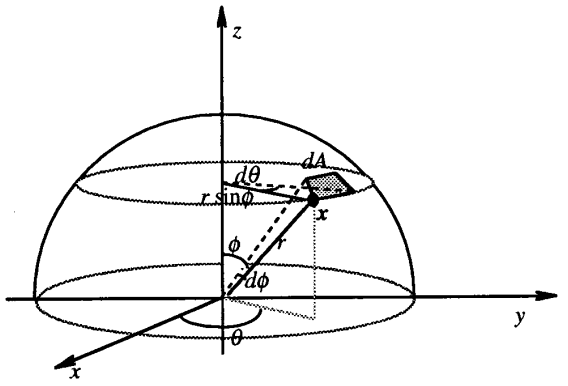



Fig. 14. Segment of a spherical shell.

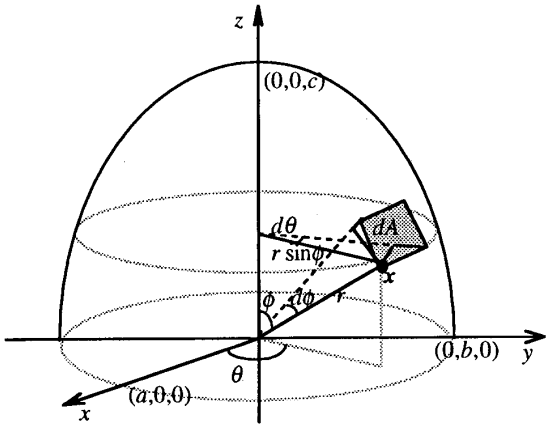



Fig. 15. Segment of an ellipsoidal shell.

radial length from the center of the ellipsoid to point $x$ given by

$$r = \sqrt{x^2 + y^2 + z^2} = \left( \frac{\sin^2\phi\cos^2\theta}{a^2} + \frac{\sin^2\phi\sin^2\theta}{b^2} + \frac{\phi\cos^2\phi}{c^2} \right)^{-\frac{1}{2}}. \quad \text{(A10)}$$

Let us suppose that the patch extends from $\phi_1$ to $\phi_2$ in $\phi$, and from $\theta_1$ to $\theta_2$ in $\theta$. In this case, the covariance matrix is still given by (A2), but $dA$ is now the elemental area of the ellipsoidal patch given by

$$dA = \sqrt{r^2 + \left(\frac{\partial r}{\partial \phi}\right)^2} \sqrt{r^2 + \left(\frac{\partial r}{\partial \theta}\right)^2} \sin\phi \, d\phi \, d\theta \quad \text{(A11)}$$

and $\mathbf{m} = [m_x, m_y, m_z]^{\mathrm{T}}$ is the centroid of the ellipsoidal patch given by (A3). Hence

$$\mathrm{Tr}\ \mathbf{C} = \frac{1}{A} \int_{\theta_1}^{\theta_2} \int_{\phi_1}^{\phi_2} r^2 dA - (m_x^2 + m_y^2 + m_z^2). \quad \text{(A12)}$$

Expressions for $A$, $m_x, m_y$ and $m_z$, can be found by using (A1), (A3), (A4), (A10) and (A11). The surface density can be then computed using (12) and (A12).

## ACKNOWLEDGMENT

The authors are grateful to the anonymous reviewers for their valuable comments, which improved the presentation and contents of this paper considerably.

## REFERENCES

[1] E. Backer and A. K. Jain, "A clustering performance measure based on fuzzy set decomposition," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 3, no. 1, pp. 66–74, 1981.

[2] P. J. Besl and R. C. Jain, "Segmentation through variable-order surface fitting," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 10, no. 2, pp. 167–192, Mar. 1988.

[3] J. C. Bezdek, "Cluster validity with fuzzy sets," *J. Cybernetics*, vol. 3, pp. 58–73, 1974.

[4] ——, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum, 1981.

[5] R. M. Bolle and B. Vemuri, "On three-dimensional surface reconstruction methods," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 13, no. 1, pp. 1–13, Jan. 1991.

[6] D. S. Chen, "A data driven intermediate level feature extraction algorithm," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, no. 7, Jul. 1989.

[7] R. N. Davé, "New measures for evaluating fuzzy partitions induced through c-shells clustering," in *Proc. SPIE Conf. Intell. Robot Computer Vision X: SPIE*, vol. 1607, Boston, Nov. 1991, pp. 406–414.

[8] R. N. Davé and K. J. Patel, "Progressive fuzzy clustering algorithms for characteristic shape recognition," in *Proc. N. Am. Fuzzy Inf. Process. Soc. Workshop*, Toronto, 1990, pp. 121–124.

[9] D. Dimitrescu, "Hierarchical pattern classification," *Fuzzy Sets and Syst.*, vol. 28, pp. 145–162, 1988.

[10] R. Dubes and A. K. Jain, "Validity studies in clustering methodologies," *Pattern Recognition*, vol. 11, no. 4, pp. 235–254, 1976.

[11] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[12] T. J. Fan, G. Medioni, and R. Nevatia, "Recognizing 3-D objects using surface description," *IEEE Trans. PAMI*, vol. 11, no. 11, pp. 1140–1157, 1989.

[13] O. D. Faugeras and M. Hebert, "The representation, recognition, and positioning of 3D shapes from range data," in *Techniques for 3D Machine Perception*,A. Rosenfeld, Ed. Amsterdam, The Netherlands: Elsevier, 1986, pp. 113–148.

[14] I. Gath and A. B. Geva, "Unsupervised optimal fuzzy clustering," *IEEE Trans. PAMI*, vol. 11, no. 7, pp. 773–781, July 1989.

[15] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*, vol. I, chapter 11. Reading, MA: Addison-Wesley, 1992.

[16] R. Hoffman and A. K. Jain, "Segmentation and classification of range images," *IEEE Trans. PAMI*, vol. PAMI-9, no. 5, pp. 608–620, 1987.

[17] T. L. Huntsberger, C. L. Jacobs, and R. L. Cannon, "Iterative fuzzy image segmentation," *Pattern Recognition*, vol. 18, no. 2, pp. 131–138, 1985.

[18] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, NJ: Prentice-Hall.

[19] J.-M. Jolion, P. Meer, and S. Bataouche, "Robust clustering with appplications in computer vision," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 13, no. 8, pp. 791–801, Aug. 1991.

[20] D. Karen, D. Cooper, and J. Subrahmonia, "Describing complicated objects by implicit polynomials," *IEEE Trans. PAMI*, vol. 16, no. 1, pp. 38–53, Jan. 1994.

[21] D. J. Kriegman and J. Ponce, "On recognizing and positioning curved 3D objects from range image contours," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, no. 12, pp. 1127–1137, Dec. 1990.

[22] R. Krishnapuram and L. Chen, "Implementation of parallel thinning algorithms using iterative neural networks," vol. 4, no. 1, pp. 142–147, Jan. 1993.

[23] 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, pp. 385–400, 1992.

[24] R. Krishnapuram, H. Frigui, and O. Nasraoui, "Quadratic shell clustering algorithms and the detection of second-degree curves," *Pattern Recognition Lett.*, vol. 14, no. 7, pp. 545–552, Jul. 1993.

[25] R. Krishnapuram and J. M. Keller, "A possibilistic approach to clustering," *IEEE Trans. Fuzzy Syst.*, vol. 1, no. 2, pp. 98–110, May 1993.

[26] R. Krishnapuram, O. Nasraoui, and H. Frigui, "The fuzzy $C$ spherical shells algorithms: A new approach," *IEEE Trans. Neural Networks*, vol. 3, no. 5, pp. 663–671, Sept. 1992.

[27] D. G. Lowe, "Fitting parametrized three-dimensional models to images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 13, no. 5, pp. 441–450, May 1991.

[28] B. Sabata, F. Arman, and J. K. Aggarwal, "Segmentation of 3-D range images using pyramidal data structures," in *Proc. Third Int. Conf. Comput. Vision*, Osaka, Dec. 1990, pp. 662–666.

[29] F. Solina and R. Bajcsy, "Recovery of parametric models from range images: The case of superquadrics with global deformations," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, no. 2, Feb. 1990, pp. 131–147.

[30] G. Taubin, F. Cukierman, S. Sullivan, J. Ponce, and D. J. Kriegman, "Parametrized functions of polynomials for bounded algebraic curve and surface fitting," *IEEE Trans. PAMI*, vol. 16, no. 3, pp. 287–303, Mar. 1994.

[31] R. L. Thorndike, "Who belongs in the family," *Psychometrika*, vol. 18, pp. 267–276, 1953.

[32] M. A. Wani and B. G. Batchelor, "Edge-region-based segmentation of range images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 16, no. 4, pp. 314–319, Apr. 1994.

[33] P. Whaite and F. P. Ferrie, "From uncertainty to visual exploration," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 13, no. 10, pp. 1038–1049, Oct. 1990.

[34] M. P. Windham, "Cluster validity for fuzzy clustering algorithms," *Fuzzy Sets and Syst.*, vol. 5, pp. 177–185, 1981.

[35] N. Yokoya and M. D. Levine, "Range image segmentation based on differential geometry: A hybride approach," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, pp. 643–649, June 1989.

**Raghu Krishnapuram** For a photograph and biography see this issue, page 43.

**Hichem Frigui** For a photograph and biography see this issue, page 43.

**Olfa Nasraoui** For a photograph and biography see this issue, page 43.