Short Papers

Template-Based Shell Clustering Using a Line-Segment Representation of Data

Tsaipei Wang

***Abstract*—This paper presents the algorithms and experimental results for template-based shell clustering when the datasets are represented by line segments. Compared with point datasets, such representations have several advantages, which include better scalability and noise immunity, as well as the availability of orientation information. Using both synthetic and real-world image datasets, we have experimentally demonstrated that line-segment-based representations result in both better accuracy and better efficiency in shell clustering.**

***Index Terms*—Line-segment approximation, line-segment matching, line-segment models, possibilistic c-means (PCMs), shell clustering, template-based clustering, template matching.**

I. Introduction

Fuzzy c-means (FCM) [1] and possibilistic c-means (PCM) [2] clustering algorithms are representative examples of prototype-based clustering algorithms. In these algorithms, each cluster is represented by a prototype, which is updated during the clustering procedure to minimize an objective function. The most-common algorithms of this class are intended for detecting "compact" or "filled" clusters.

Shell-clustering algorithms, on the other hand, use prototypes that are "shells" in the feature space. The ability to efficiently detect shell-like structures of particular shapes is useful in many image processing and computer-vision applications. Examples of shell-clustering algorithms include the detection of circles [3]–[5]; general quadratic shells, such as ellipses and hyperbola [6]–[9]; rectangles [10]; and template-based shapes [11], [12].

All the existing shell-clustering algorithms are designed to cluster points in the feature space. However, for most common applications of shell clustering, where we are concerned with 2-D data, the shells are curvilinear features and can be approximated with a set of line segments, which provides a more compact representation of the data than the original set of points. Edge points in images are often grouped into line segments before processing, such as in [13] and [14]. Some researchers have studied the problem of matching line-segment-based data with a line-segment-based model or template [14]–[16]. However, this approach has never been studied in the context of shell clustering. In the following, we describe several advantages of representing the data points with line segments before clustering them into particular shapes.

1) *Scalability and efficiency*: The number of data points needed to represent a shape varies with its size, assuming a fixed spatial resolution. On the other hand, the number of line segments needed to represent the same shape remains constant. As a result, a data representation based on line segments is more efficient and practical for larger datasets, such as the edges extracted from high-resolution images.
2) *Immunity to noise/outliers*: During the process of grouping data points into line segments, an additional benefit is the identification of those points that are unlikely to belong to any line segments. Such points are presumably outliers, and by removing them in this stage, we simplifies the problem and prevents these outliers from affecting the final clustering results.
3) *Estimation of line widths*: The amount of scatter (or the line/curve width) is an important parameter for many other algorithms for line, curve, and shape detection. For example, in our previous template-based shell-clustering algorithm [12], this parameter is directly related to the range parameter used in the computation of linear densities and convergence checking for individual prototypes. Previously, this parameter has to be assigned by the user. By first grouping the data points into line segments, we can obtain an estimation of this parameter for the subsequent clustering process, thereby making the overall algorithm either globally or locally more adaptive to different data characteristics.

The motivation and main contribution of this paper is to demonstrate the feasibility and advantages of applying shell clustering to a dataset represented as line segments. To allow for meaningful comparison between the experimental results using point-based and line-segment-based data representation, we always start from a set of points, obtain a corresponding line-segment representation, and then apply shell clustering on both using identical conditions, including the number of clusters, the method of initialization, and other shared parameters.

This study is based on our previous work on possibilistic shell clustering of template-based shapes (PCT) [12], a brief review of which is given in Section II. Section III describes the modifications to the existing algorithm for the clustering of line segments. The methods used in our experiments to obtain line-segment-based representations from sets of points are covered in Section IV. We present our experimental results in Section V, followed by the conclusion in Section VI.

Manuscript received January 22, 2010; revised July 6, 2010 and October 5, 2010; accepted December 7, 2010. Date of publication January 13, 2011; date of current version June 6, 2011.

The author is with the Department of Computer Science, National Chiao Tung University, Hsinchu 300, Taiwan (e-mail: wangts@cs.nctu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TFUZZ.2011.2105880

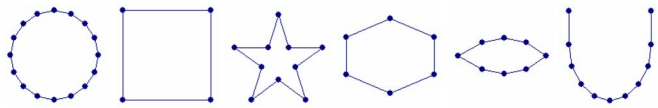

Fig. 1. Templates used in our experiments.

II. Shell Clustering of Template-Based Shapes

PCT is a prototype-based clustering method with each prototype being a transformed version of a template. A template is defined by a set of vertices and the edges that connect them. Fig. 1 displays the templates used in our experiments.

Three types of prototype transforms are described in [12], which are as follows.

Type I: a transform that involves translation, scaling, and rotation;

Type II: similar to type I, but with a different scaling factor for each dimension in the template frame of coordinates;

Type III: affine transform.

The update equations for the prototypes and memberships are derived from the standard objective function of PCM [2]:

$$J_{\mathrm{PCM}} = \sum_{i=1}^{C}\sum_{j=1}^{N} u_{ij}^{m} d_{ij}^{2} + \sum_{i=1}^{C} \eta_i \sum_{j=1}^{N} (1-u_{ij})^{m}. \tag{1}$$

Here, N is the number of data points, C is the number of clusters, m is the fuzzification factor, u_{ij} is the membership of a point $\boldsymbol{x}_j$ ($1 \le j \le N$) in the ith cluster, and d_{ij} is the distance between $\boldsymbol{x}_j$ and the ith cluster prototype. The parameter η_i is the "bandwidth" of a cluster that controls the dependence of u_{ij} on d_{ij}. For PCT, the distance measure d_{ij} is computed as the shortest Euclidean distance between $\boldsymbol{x}_j$ and the ith cluster prototype. This requires the identification of $\boldsymbol{p}_{ij}$, which is the match point of $\boldsymbol{x}_j$ (i.e., the closest point to $\boldsymbol{x}_j$) on the ith cluster prototype.

Our algorithms, both in [12] and in this paper, are based on PCM instead of FCM. One reason is the well-known issue that FCM is more sensitive to noise or outliers. Another problem of FCM is that when a prototype converges at an incorrect location that overlaps with part of one or more actual clusters, which happens frequently for general shapes, it can prevent other prototypes from converging to those clusters as well. This is not a problem with PCM as each prototype in PCM is independent. On the other hand, the known problems of PCM (sensitivity to initialization and overlapping prototypes) are handled with the progressive clustering procedure described below.

To overcome the many local optima in the optimization process, the alternating-optimization scheme to update prototypes and memberships is placed inside a progressive clustering procedure. The main features of our progressive clustering procedure are summarized as follows.

1) A fixed number of prototypes are maintained by replacing deleted prototypes with new ones.
2) In each iteration, prototypes that are very close to each other are merged.
3) Prototypes with high densities are moved to a separate list, and points within a small range δ_w of such good prototypes are excluded from subsequent iterations. The density works as a single-cluster validity measure [9] and is given by

$$\rho_i = \frac{1}{L_i} \sum_{d_{ij} \le \delta_w} u_{ij} \tag{2}$$

where L_i is the total length of all the edges in a prototype.
4) Spurious (i.e., low-density) prototypes are deleted.
5) For each prototype, the bandwidth parameter η_i in (1) is initialized to a large value and gradually reduced to a lower bound close to δ_w^2.
6) The main loop of progressive clustering terminates when the number of remaining data points drops below a predetermined ratio or when a maximum number of iterations are reached.

Detailed description of these features can be found in [12].

III. Shell Clustering of Template-Based Shapes With Line-Segment Data

This section describes how we extend our previous algorithm to work with datasets consisting of line segments instead of points. These line segments are called "data segments" below. The objective function (1) is modified to

$$J_{\mathrm{LPCT}} = \sum_{i=1}^{C}\sum_{j=1}^{N_L} n_j u_{ij}^{m} d_{ij}^{2} + \sum_{i=1}^{C} \eta_i \sum_{j=1}^{N_L} n_j (1-u_{ij})^{m} \tag{3}$$

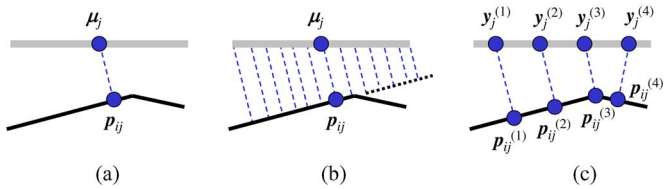

Fig. 2. Various data-segment-to-prototype-edge distance measures. (a) Midpoint method. (b) MSPD method. (c) Multipoint method with four key points. The gray solid line represents the data segment, and the black solid line represents the prototype edges.

where N_L is the number of data segments, and n_j is the number of original data points represented by the jth data segment.

The first problem that arises here is how to compute d_{ij} between a data segment and a prototype. Three different possibilities are listed as follows and are illustrated in Fig. 2.

1) *The midpoint method* [see Fig. 2(a)]: This method reduces the problem to point-based PCT, with the jth data segment represented only by its midpoint $\boldsymbol{\mu}_j$. Here, d_{ij} is simply the distance between $\boldsymbol{\mu}_j$ and $\boldsymbol{p}_{ij}$, which is the match point of $\boldsymbol{\mu}_j$ on the ith prototype. This approach is efficient and requires minimal modification to the point-based PCT implementation. The drawback is that we lose the orientation information of the data segments, which is important for determining the correct prototype transform.
2) *The mean-squared-perpendicular-distance (MSPD) method* [see Fig. 2(b)]: The idea is to compute the mean-squared distance between all the points on the data segment and the prototype edge that contains $\boldsymbol{p}_{ij}$. Let h_j be the length of the jth data segment, and let θ_{ij} be the angle between the data segment and the prototype edge containing $\boldsymbol{p}_{ij}$. Assuming that the whole data segment is matched to the same prototype edge, it is straightforward to prove that

$$d_{ij}^{2} = \frac{1}{h_j}\int_{-h_j/2}^{h_j/2} [\|\boldsymbol{\mu}_j - \boldsymbol{p}_{ij}\| + x\sin\theta_{ij}]^2\, dx$$

$$= \|\boldsymbol{\mu}_j - \boldsymbol{p}_{ij}\|^2 + \frac{h_j^2}{12}\sin^2\theta_{ij}. \tag{4}$$

This MSPD distance measure does contain the orientation information through θ_{ij}, and it is actually possible to optimize the prototype rotation angle based on this distance for type I transforms [15]. However, it is difficult to use this angle to optimize our types II and III or other more complicated prototype transforms. In addition, the match points of a data segment often fall on multiple prototype edges. This adds a lot of computational complexity since a data segment needs to be subdivided into subsegments, with each matched to a different prototype edge, so that (4) can be computed for each subsegment. An approximate solution, as used in [15], is to match the whole data segment to a single (infinitely extended) prototype edge [see the dotted line in Fig. 2(b)].
3) *The multipoint method* [see Fig. 2(c)]: This is a compromise between the midpoint and MSPD methods. We place a small number of evenly spaced key points on a data segment and compute their mean-squared distances to a prototype, thereby leading to a distance measure that is an approximation of MSPD

$$d_{ij}^{2} = \frac{1}{n_p(j)} \sum_{k=1}^{n_p(j)} \|\boldsymbol{y}_j^{(k)} - \boldsymbol{p}_{ij}^{(k)}\|^2. \tag{5}$$

Here, $n_p(j)$ is the number of key points, $\boldsymbol{y}_j^{(k)}$ is the kth key point on the jth data segment, and $\boldsymbol{p}_{ij}^{(k)}$ is the match point of $\boldsymbol{y}_j^{(k)}$ on the ith prototype. By imposing an upper bound on $n_p(j)$, we can limit the computational complexity to stay linear with respect to the number of data segments. In our experiments, we have used an upper bound of four key points per data segment, which is a value selected empirically that provides good compromise between accuracy and efficiency for our datasets.

The multipoint method is what we have implemented in our system. Compared with the midpoint method, here, we are able to make use of the orientation information, although only implicitly. To better understand this, let us consider the case when there is an angle between a data segment and a prototype edge, while the midpoint of the data segment falls on the prototype edge. The distance according to the midpoint method is zero. However, with the multipoint method, the torque created by the "attraction" from the key points can rotate (or, more generally, modify the orientation of) the prototype edge toward the data segment. The use of orientation information without explicitly using angles also makes the application to, say, affine prototype transforms straightforward, which is an advantage over the MSPD method. As the key points of a data segment can have match points on different prototype edges, the multipoint method provides an approximate solution to the problem of data-segment subdivision of the MSPD method.

Another place where we utilize the orientation information is during the progressive clustering procedure, where the density of a prototype is computed using

$$\rho_i = \frac{1}{L_i} \sum_{\|\boldsymbol{y}_j^{(k)} - \boldsymbol{p}_{ij}^{(k)}\| \le \delta_w} \left[\frac{n_j}{n_p(j)} \right] u_{ij} \cos \theta_{ij}^{(k)} \tag{6}$$

instead of (2). The factor $\cos \theta_{ij}^{(k)}$ helps to prevent the case when a key point happens to be located within δ_w of a prototype edge while θ_{ij} is large, thereby incorrectly causing all the data points associated with the key point to completely contribute to the density of the prototype.

IV. Representing Data With Line Segments

So far, our algorithm imposes no constraint on how the line segments are obtained. However, to be able to compare the proposed algorithm with point-based PCT, we need to start from the same (point) data for both. This requires the additional step of extracting line segments from point data. The two sections below briefly describe the two methods used for this step in our experiments. Neither of them requires a prespecified number of line segments. Please note that these methods are independent of the actual shell clustering, and users can choose among many available algorithms for line-segment extraction.

A. Line-Segment Representation of Generic Point Data

For our synthetic datasets, we obtain the line-segment representation using robust competitive agglomeration (RCA) [17]. Its objective function is given by

$$J_{\mathrm{RCA}} = \sum_{i=1}^{C} \sum_{j=1}^{N} u_{ij}^2 \gamma_i(d_{ij}^2) - \alpha \sum_{i=1}^{C} \left[\sum_{j=1}^{N} w_{ij} u_{ij} \right]^2. \tag{7}$$

The $\gamma_i(d_{ij}^2)$ factor is a cluster-dependent loss function that discriminates against points far away from the ith cluster, and w_{ij}, which is the derivative of γ_i, gives the "typicality" of $\boldsymbol{x}_j$ in the ith cluster. The second term in the cost function favors larger clusters and has the effect of shrinking the smaller clusters, thereby gradually reducing the total number of clusters. As a result, RCA is started with an overspecified number of clusters. The parameter α controls the pace of agglomeration.

In this section, each cluster is a line segment, which is different from the other sections where a cluster is an instance of the shape being detected. The prototype parameters of the ith cluster consist of the robust mean $\boldsymbol{\mu}_i$ and the robust fuzzy-covariance matrix $\boldsymbol{M}_i$ [17]

$$\boldsymbol{\mu}_i = \frac{\left[\sum_{j=1}^{N} u_{ij}^2 w_{ij} \boldsymbol{x}_j \right]}{\left[\sum_{j=1}^{N} u_{ij}^2 w_{ij} \right]} \tag{8}$$

$$\boldsymbol{M}_i = \frac{\left[\sum_{j=1}^{N} u_{ij}^2 w_{ij} (\boldsymbol{x}_j - \boldsymbol{\mu}_i)(\boldsymbol{x}_j - \boldsymbol{\mu}_i)^T \right]}{\left[\sum_{j=1}^{N} u_{ij}^2 w_{ij} \right]}. \tag{9}$$

Let a_{i1} and a_{i2} ($a_{i1} \le a_{i2}$) be the eigenvalues of $\boldsymbol{M}_i$, and let $\boldsymbol{e}_{i1}$ and $\boldsymbol{e}_{i2}$ be their respective eigenvectors. The corresponding line segment is centered at $\boldsymbol{\mu}_i$ and along the direction given by $\boldsymbol{e}_{i2}$. To compute the length and width of the line segment, we utilize the fact that the ratio between the half-width and the standard deviation of a 1-D uniform distribution is $\sqrt{3}$. By assuming uniform point distribution in the line segment, its length (i.e., full width) can be estimated as $2\sqrt{3}$ times the standard deviation along the major (minor) axis $\sqrt{a_{i2}}$ ($\sqrt{a_{i1}}$).

As of the distance measure d_{ij}, we use the shortest Euclidean distance between a point and a line segment

$$d_{ij}^2 = \begin{cases} [(\boldsymbol{x}_j - \boldsymbol{\mu}_i) \bullet \boldsymbol{e}_{i1}]^2, & \text{if } |(\boldsymbol{x}_j - \boldsymbol{\mu}_i) \bullet \boldsymbol{e}_{i2}| \le \sqrt{3a_{i2}} \\ [(\boldsymbol{x}_j - \boldsymbol{\mu}_i) \bullet \boldsymbol{e}_{i1}]^2 + [|(\boldsymbol{x}_j - \boldsymbol{\mu}_i) \bullet \boldsymbol{e}_{i2}| - \sqrt{3a_{i2}}]^2, & \text{otherwise.} \end{cases} \tag{10}$$

We do not use the distance measure of adaptive fuzzy clustering (AFC) [18], which is also used in [17], due to its tendency of "bridging the gaps" between actually separate line segments.

For noisy datasets, even a robust clustering algorithm, like RCA, may still produce line segments that consist of mostly noise points. As the noise line segments generally have lower linear densities compared with the real ones, we empirically select one fourth of the weighted mean linear density of all the line segments as a threshold and discard those line segments whose linear densities are below this threshold.

To prevent different actual line segments from being merged together, we specifically use a small α in (3) for weaker agglomeration. The side effect is that some actual line segments may be divided into multiple pieces. As a result, we add an additional step of compatible cluster merging (CCM) [9] to merge these pieces back together.

B. Line-Segment Representation of Image Data

When trying to approximate the edge points in an image with line segments, it is a common practice to assume that the edge points are connected and the edges have a maximum width. Based on these assumptions, we employ the following procedure to obtain a set of line segments from the edge points.

1) We first extract the connected components from the edge points. We use eight-connectivity as it is less likely to oversegment thin lines that are about 1-pixel wide. Those components containing fewer points than a threshold (which is currently 10) are deleted to reduce the effect of spurious edge points.
2) Each connected component is approximated with a line segment according to (8) and (9). If its full width ($2\sqrt{3a_1}$) is above a given threshold (which is currently 2 pixels), the component is split using two-cluster FCM with the distance measure (10). The two resulting clusters are first defuzzified and then further divided into eight-connected components. This process is repeated until

TABLE I
SUMMARY OF DATASETS

Synthetic Data Sets					Image Data Sets				
Dataset Name	Type	N	C^*	C_0	Dataset Name	Type	N	C^*	C_0
Circles	I	677	3	2	Ring of beads	I	748	12	10
U-shape	I	612	4	3	Leaves	II	740	5	8
Ellipses	II	587	3	5	Paper cards	III	403	3	4
Rectangles	II	768	3	6					
Stars	III	805	5	10					
Parallelogram	III	575	3	5					

Type: type of prototype transformation; N: number of nonnoise points; C^*: number of actual clusters; C_0: number of prototypes used during clustering.

the line segment approximating each connected component has a width within the threshold.

3) Finally, we use CCM to merge oversegmented line segments.

V. EXPERIMENTAL RESULTS

Table I summarizes the datasets used in our experiments. There are six synthetic datasets and three image datasets; the image data are also used in [12]. For each synthetic dataset, we also have a noisy version with 800 additional uniformly distributed noise points. The number 800 is selected to exceed the number of nonnoise (in-lier) points in most cases.

Fig. 3 contains the example clustering results of the synthetic datasets. The three main columns, from left to right, contain results using types I, II, and III prototype transforms, respectively. For each dataset, the left plot shows the original data points with the data segments, and the right plot shows the final prototypes overlaid on the data segments. The results indicate that the algorithm is capable of detecting the desired shapes from the data segments.

When comparing the results for noiseless and noisy data, we can see that most of the noise points are not covered by any data segment. This, in turn, leads to similar clustering results with or without noise. These are examples of the advantage stated earlier regarding improved noise/outlier immunity of line-segment-based representations. This observation is quantitatively verified later in this section.

Fig. 4 contains example clustering results for the image datasets. The left column is the original image (160×120) overlaid with the final prototypes; the other two columns have the same meanings as the corresponding columns in Fig. 3. The steps of edge-point extraction include smoothing, gradient computation, thresholding, and thinning, although for brevity, we do not explain the details here. Our algorithm can be applied to gray-scale or color images in the same way, with the only difference being how the edge points are determined. As there are many possibilities for this purpose, each user can select any method that works for the particular application.

We evaluate the accuracy of the clustering results by comparing the final prototypes with the ground truth. For synthetic datasets, the ground truth consists of the actual locations of all the shell clusters (which are called the "target clusters" below) used to generate the data. The ground truth of the image datasets are marked manually. We use the "grade of detection" g_d for a target cluster for evaluation purposes, which is defined in [12] as

$$g_d(P_T) = \begin{cases} 1, & d \le \delta_w \\ (3 - d/\delta_w)/2, & \delta_w < d \le 3\delta_w \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

Here, P_T is the target cluster prototype, and d is its maximum deviation from the closest prototype generated by the clustering algorithm. Such a "soft" definition of detection provides a gradual degradation of accuracy for data with some scatter.

The performance comparison is based on two measures: the terminal g_d (which is denoted as g_d^*; the larger, the better), and the amount of computation (which is denoted as k_c; the smaller, the better) required to reach a given g_d, thus taking into account both accuracy and efficiency. Following [12], we define k_c as the number of d_{ij} computations (including matching-point determination) per data point as this is the most computationally expensive step of the whole algorithm.

Table II lists the performance comparison, including g_d^* and k_c at $g_d = 0.5$ (noiseless data) or 0.25 (noisy data). We use k_c at $g_d = 0.25$ for noisy data to allow the comparison between the results using point-based and line-segment-based data representations, as g_d never reaches 0.5 when using the point-based method. The reported values are all averaged over all the target clusters and 20 test runs. For the noiseless datasets, the line-segment-based method consistently produces g_d^* values that are comparable with or slightly better than the point-based method. The differences are much more significant for the noisy datasets, which are consistent with our expectation that line-segment-based representations are more immune to noise/outliers. The comparison of k_c values also clearly indicates much better efficiency for the line-segment-based representation. Please note that many factors, including characteristics of the shapes themselves and the number of instances, can affect the performance, and we naturally will expect more complicated datasets (those with more instances or more complicated shapes) to require more computation to process.

Now that our algorithm uses line segments to represent both the template and data, it can be applied to problems that involve the matching or registration between two sets of line segments, which has been a research topic in computer vision for a long time. Here, we compare our algorithm with a well-known technique for this purpose: the local-search-based method in [15]. We choose this technique because it bears some similarity to our approach in that both use random starts that search locally for the solution by iteratively minimizing a cost function in a steepest-descent-like manner. The difference is that our local search is in the space of transform parameters, and the local search in [15] is in the space of possible correspondences between the two sets of line segments. We use only type I transforms for our method here due to the limitations of the method in [15].

The progressive clustering procedure in Section IV is modified to facilitate this experiment. Specifically, we maintain only a single prototype at any time, and the progressive clustering procedure continues until we have obtained a prespecified number of detections. We use the total number of iterations in the progressive clustering procedure, which is denoted n_{it} here, as the unit for the comparison of the amount of computation. This is because each iteration has the same form of asymptotic complexity of $O(N_d N_m)$ in both methods, with N_d and N_m being the number of data segments and model segments (prototype edges), respectively. For our method, this involves the distance computation between all the pairings of data segments and prototype edges. For the method in [15], this involves the computation of "fit error" for each set of correspondences in the Hamming-distance-one neighborhood of the current set of correspondences.

The experimental results, which are averaged over 20 runs, are listed in Table III. Only the three datasets in Table II intended for type I transforms are included. Our method consistently performs better (higher g_d and smaller n_{it}) than the method in [15]. It is also worth noting that the added noise hardly affect n_{it} for our method but significantly increase n_{it} for the method in [15]. The large difference in g_d^* for the "ring of beads" dataset, where the actual shapes are more localized, seems to indicate that our method is preferred in such scenarios.

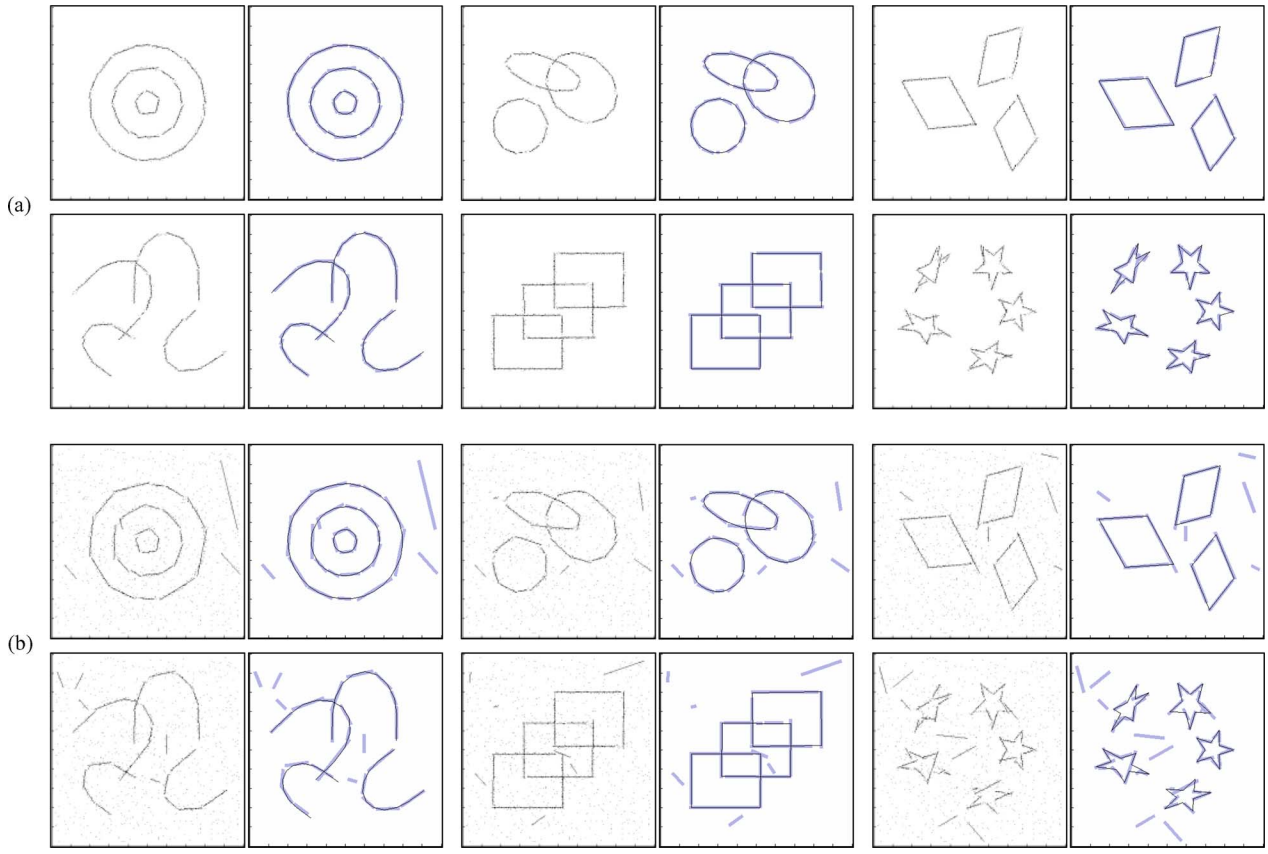

Fig. 3. Example clustering results of synthetic datasets. (a) Noiseless datasets. (b) Noisy datasets. For each dataset, the left plot is the original point data overlapped with the data segments, and the right plot displays both the data segments (gray thick line segments) and the final prototypes.

TABLE II
PERFORMANCE COMPARISON BETWEEN CLUSTERING RESULTS USING POINT-BASED AND LINE-SEGMENT-BASED DATA REPRESENTATION

Dataset Name	Noiseless Sets				Noisy Sets			
	g_d^*		k_c at g_d=0.5		g_d^*		k_c at g_d=0.25	
	Point	Line	Point	Line	Point	Line	Point	Line
Circles	0.90	1.00	111	19	0.25	1.00	216	6
U-shape	0.75	0.80	241	75	0.40	0.60	227	14
Ellipses	0.98	1.00	121	51	0.33	0.74	150	9
Rectangles	0.64	1.00	209	19	0.44	0.95	134	6
Stars	0.98	1.00	319	182	0.30	0.95	1875	48
Parallelogram	1.00	0.98	162	29	0.17	0.88	-	7
Ring of beads	0.94	1.00	649	146				
Leaves	0.92	0.99	661	213				
Paper cards	0.83	0.96	336	74				

TABLE III
PERFORMANCE COMPARISON BETWEEN OUR ALGORITHM AND THE LOCAL-SEARCH METHOD IN [15]

Dataset Name	Noiseless Sets				Noisy Sets			
	g_d^*		n_{it}		g_d^*		n_{it}	
	(A)	(B)	(A)	(B)	(A)	(B)	(A)	(B)
Circles	1.00	0.82	77	132	0.99	0.95	81	282
U-shape	0.87	0.72	520	610	0.56	0.52	506	770
Ring of beads	0.92	0.56	698	775				

(A) Our method; (B) the method given in [15].

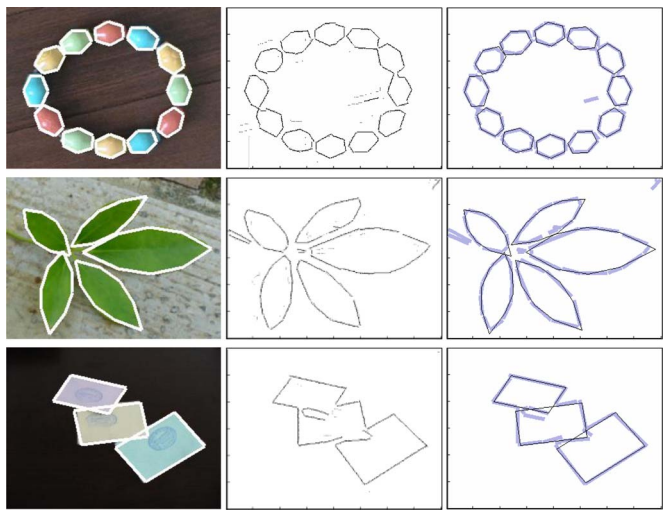

Fig. 4. Example of clustering results of image datasets. For each dataset, the columns, from left to right, are the original image overlaid with the final prototypes, edge points overlapped with the data segments, and the data segments (gray thick line segments) overlapped with the final prototypes, respectively.

VI. CONCLUSION

In summary, we have implemented the algorithms to apply shell clustering to datasets represented as line segments. Such representations are more compact and scalable than point datasets because they

already provide a level of information condensation by grouping points into line segments. The algorithms, which are given in this paper, are based on our previous work on template-based shell clustering. We have presented clustering results for both synthetic and real-world image datasets with shell clusters of several different template-based shapes to demonstrate the effectiveness of this new approach. The quantitative performance comparison provides clear evidence of improvements, in both efficiency and accuracy, by representing the point data with line segments. In addition, we achieve much better noise immunity with line-segment-based data representation by employing a robust method such as RCA to extract the line segments, thereby effectively excluding most noise points from the shell-clustering procedure. Favorable comparison with the method in [15] also supports our algorithm as a general-purpose technique to match line segments.

Note that, while the algorithm to generate a line-segment representation from a set of points is not by itself the focus of this paper, the quality of such a representation does affect the performance of our algorithm. Therefore, it is important to select an algorithm and its parameters appropriately for the particular application. We have used an RCA-based method for generic point datasets because it does not require the selection of many parameters and is not very sensitive to initialization. Other methods might work better in different scenarios. For example, gradient-direction information and/or the color/brightness/texture of nearby pixels may help to produce a more accurate line-segment representation from the edge points in an image.

We have also identified a number of research topics that we are interested to pursue. A possibility is the extension to datasets of more than two dimensions. For example, the role of line segments in 2-D is replaced by planar patches in 3-D. An interesting direction is to integrate CCM and template-based shapes using planar patches, which is an extension of the work done in [9] and [17] for quadratic surfaces. We are also interested in applying our algorithm to such applications as line-segment-based image registration, including performance comparisons with other existing algorithms for this purpose. In addition, it should be interesting to investigate the replacement of PCM with several related, but more recent, algorithms, such as [19]–[21], as our "base" clustering algorithm, as it remains unknown how these algorithms perform in shell clustering.

REFERENCES

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

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

[3] R. N. Dave, "Fuzzy shell-clustering and application to circle detection in digital images," *Int. J. Gen. Syst.*, vol. 16, pp. 343–355, 1990.

[4] Y. H. Man and I. Gath, "Detection and separation of ring-shaped clusters using fuzzy clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 8, pp. 855–861, Aug. 1994.

[5] R. Krishnapurum, O. Nasraoui, and H. Frigui, "The fuzzy c spherical shells algorithm: A new approach," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 663–671, Sep. 1992.

[6] R. N. Dave and K. Bhaswan, "Adaptive fuzzy C-shells clustering and detection of ellipses," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 643–662, Sep. 1992.

[7] H. Frigui and R. Krishnapurum, "A comparison of fuzzy shell clustering methods for the detection of ellipses," *IEEE Trans. Fuzzy Syst.*, vol. 4, no. 2, pp. 193–199, May 1996.

[8] R. Krishnapurum, H. Frigui, and O. Nasraoui, "Fuzzy and possibilistic shell clustering algorithms and their application to boundary detection and surface approximation—Part I," *IEEE. Trans. Fuzzy Syst.*, vol. 3, no. 1, pp. 29–43, Feb. 1995.

[9] R. Krishnapurum, H. Frigui, and O. Nasraoui, "Fuzzy and possibilistic shell clustering algorithms and their application to boundary detection and surface approximation—Part II," *IEEE. Trans. Fuzzy Syst.*, vol. 3, no. 1, pp. 44–60, Feb. 1995.

[10] F. Hoeppner, "Fuzzy shell clustering algorithms in image processing: Fuzzy c-rectangular and 2-rectangular shells," *IEEE Trans. Fuzzy Syst.*, vol. 5, no. 4, pp. 599–613, Nov. 1997.

[11] X.-B. Gao, W.-X. Xie, J.-Z. Liu, and J. Li, "Template based fuzzy c-shells clustering algorithm and its fast implementation," in *Proc. IEEE Int. Conf. Signal Process.*, 1996, pp. 1269–1272.

[12] T. Wang, "Possibilistic shell clustering of template-based shapes," *IEEE Trans. Fuzzy Syst.*, vol. 17, no. 4, pp. 777–793, Aug. 2009.

[13] M. Barni and R. Gualtieri, "A new possibilistic clustering algorithm for line detection in real world imagery," *Pattern Recognit.*, vol. 32, pp. 1897–1909, 1999.

[14] X. Ren, "Learning and matching line aspects for articulated objects," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2007, pp. 1–8.

[15] J. R. Beveridge and E. M. Riseman, "How easy is matching 2d line models using local search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 6, pp. 564–579, Jun. 1997.

[16] L.-K. Sharka, A. A. Kurekinb, and B. J. Matuszewski, "Development and evaluation of fast branch-and-bound algorithm for feature matching based on line segments," *Pattern Recognit.*, vol. 40, pp. 1432–1450, 2007.

[17] H. Frigui and R. Krishnapuram, "A robust competitive clustering algorithm with applications in computer vision," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 5, pp. 450–465, May 1999.

[18] R. N. Dave, "Use of the adaptive fuzzy clustering algorithm to detect lines in digital images," in *Proc. SPIE*, 1989, vol. 1192, pp. 600–611.

[19] M. S. Yang and K. L. Wu, "Unsupervised possibilistic clustering," *Pattern Recognit.*, vol. 39, pp. 5–21, 2006.

[20] N. R. Pal, K. Pal, J. M. Keller, and J. C. Bezdek, "A possibilistic fuzzy *c*-means," *IEEE Trans. Fuzzy Syst.*, vol. 13, no. 4, pp. 517–530, Aug. 2005.

[21] J.-S. Zhang and Y.-W. Leung, "Improved possibilistic *c*-means clustering algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 2, pp. 209–217, Apr. 2004.

A Cluster-Validity Index Combining an Overlap Measure and a Separation Measure Based on Fuzzy-Aggregation Operators

Hoel Le Capitaine and Carl Frélicot

***Abstract*—Since a clustering algorithm can produce as many partitions as desired, one needs to assess their quality in order to select the partition that most represents the structure in the data, if there is any. This is the rationale for the cluster-validity (CV) problem and indices. This paper presents a CV index that helps to find the optimal number of clusters of data from partitions generated by a fuzzy-clustering algorithm, such as the fuzzy *c*-means (FCM) or its derivatives. Given a fuzzy partition, this new index uses a measure of multiple cluster overlap and a separation measure for each data point, both based on an aggregation operation of membership degrees. Experimental results on artificial and benchmark datasets are given to demonstrate the performance of the proposed index, as compared with traditional and recent indices.**

***Index Terms*—Aggregation operators (AOs), cluster validity (CV), fuzzy-cluster analysis, triangular norms (t-norms).**

Manuscript received February 2, 2010; revised June 28, 2010 and October 11, 2010; accepted December 31, 2010. Date of publication January 17, 2011; date of current version June 6, 2011.

The authors are with the Mathematics, Image, and Applications Laboratory, University of La Rochelle, 17000 La Rochelle, France (e-mail: hoel.le_capitaine@univ-lr.fr; carl.frelicot@univ-lr.fr).

Digital Object Identifier 10.1109/TFUZZ.2011.2106216