Segmentation of Satellite Imagery of Natural Scenes Using Data Mining

Leen-Kiat Soh, *Member, IEEE,* and Costas Tsatsoulis, *Senior Member, IEEE*

***Abstract*—In this paper, we describe a segmentation technique that integrates traditional image processing algorithms with techniques adapted from knowledge discovery in databases (KDD) and data mining to analyze and segment unstructured satellite images of natural scenes. We have divided our segmentation task into three major steps. First, an initial segmentation is achieved using dynamic local thresholding, producing a set of regions. Then, spectral, spatial, and textural features for each region are generated from the thresholded image. Finally, given these features as attributes, an unsupervised machine learning methodology called conceptual clustering is used to cluster the regions found in the image into *N* classes—thus, determining the number of classes in the image automatically. We have applied the technique successfully to ERS-1 synthetic aperture radar (SAR), Landsat thematic mapper (TM), and NOAA advanced very high resolution radiometer (AVHRR) data of natural scenes.**

***Index Terms*—Clustering methods, image segmentation, natural scene analysis.**

I. Introduction

THE IMPROVEMENT in the resolution of satellite imagery and the increase in the number of satellites that are used for remote sensing and monitoring of the environment have led to a dramatic increase in the volume of the data available to scientists. Full exploitation of these data requires that analysis approaches be as fully automated as possible, while allowing for critical, but limited, user interaction.

The analysis of satellite imagery of natural scenes presents many unique problems, and it differs from the analysis and segmentation of urban, commercial, or agricultural areas. Natural scenes (forests, mountains, the sea, clouds, etc.) are not structured and cannot be represented easily by regular rules or grammars. In contrast to artificial structures, natural objects do not obey strict positioning rules (for example, harbor structures will always be next to water, while coniferous forests can be found in very large geographical and elevation ranges, and their positioning rules are much weaker). Finally, the appearance of natural objects can vary greatly based on the geographic area, the season, the current weather conditions, or the past weather conditions. For example, the backscatter of synthetic aperture radar (SAR) sea ice images of the polar regions is affected by factors such as snowfall and wind velocity [1]–[3]. The backscatter of forest images can be affected by chemical pollution [4], soil erosion [5], rainfall and soil moisture [6], [7], drought [8] or disruption of hydrological regimes [9], burning [10], and extensive grazing [11]. To illustrate further, researchers working on mapping and understanding mountainous ecology and geology have to deal with different spectral results of satellite imagery of mountainous regions, caused by shading [12], snowcover [13] or snowmelt [14], or volcanic activities [15], to list a few.

All of these peculiarities of natural scenes coupled with the requirement for automated or semiautomated analysis of satellite imagery, along with philosophical issues pertinent to designing a single algorithm capable of handling many situations within a same domain, have led to the need for new approaches to the segmentation and analysis of satellite imagery of natural scenes. In this paper, we present a methodology that integrates traditional image processing algorithms with techniques adapted from knowledge discovery in databases (KDD) and data mining [16] to generate a system that successfully segments natural scene images. In our methodology, we first analyze the natural scene using traditional image segmentation and analysis techniques, namely, dynamic, multilevel thresholding and generation of texture values. These preprocessing steps generate data that, together with the original image, are viewed as the contents of a database that needs to be mined. The goal of data mining is to extract coherent, describable clusters of data that represent the natural objects and classes of objects in the scene. For data mining, we use conceptual clustering, which identifies semantically coherent clusters, representing the objects in the scene.

Note that we propose a *segmentation* technique in this paper which serves as a preprocessor to classification. We believe that an image processing task on satellite imagery is incremental and that each previous step influences the execution of the next. Hence, a good segmentation is required to obtain a good classification result. Further, we argue that, once we obtain the segmentation classes of an image, it is possible to use heuristics or other domain-specific approaches to further classify, interpret, understand, register, or extract information from the segmented image. For example, domain knowledge has been used to improve detection of drainage channel networks from satellite images [17] after the first-tier image segmentation; shape analysis has been employed to identify within a segmentation class different land use classes [18]; geophysical rules have been utilized to improve initial sea

Manuscript received April 9, 1997; revised May 14, 1998. This work was supported in part by NASA Research Grant NAGW-3043, Naval Research Laboratory Contract N00014-95-C-6038, and NSF Grant CISE-CDA-9401021.

The authors are with the Department of Electrical Engineering and Computer Science, The University of Kansas, Lawrence, KS 66045 (e-mail: lksoh@eecs.ukans.edu; tsatsoul@eecs.ukans.edu).

Publisher Item Identifier S 0196-2892(99)01965-8.

ice classification, which is based on spectral segmentation [19]; ancillary data have been integrated to improve classification [20].

II. Methodology

Our underlying methodology is to first use image segmentation to provide features to describe a preliminary segmentation of the image and, given this information, perform conceptual clustering to refine and produce the final segmentation. This methodology is similar to that of data mining [16] in that the image, viewed as data, is preprocessed, transformed, mined, and evaluated to arrive at the end result, which is the segmentation. The segmented image, thus, is the knowledge that we learn from the raw image. In addition, this two-stage approach allows modularity and flexibility in its implementation. The initial segmentation allows conceptual clustering to focus its analysis only on selected information from the image, reducing the amount of computational work and preventing irrelevant information from influencing the outcome of the clustering process. On the other hand, the conceptual clustering module relieves the burden of determining the number of classes from the preliminary segmentation process. Hence, the segmentation process itself can concentrate on low-level, syntactic image manipulation, leaving higher order tasks, such as semantic interpretation, to conceptual clustering. The task of image segmentation has been defined [21], [22] as a process of detecting maximally homogeneous regions in the image; in this paper, we consider that the *pragmatic* task of image segmentation is to segment an image into different regions such that each region, while satisfying the criteria of [21], [22], can be further analyzed using domain and application knowledge, as we have indicated above. For example, in the work by Li [18], reservoirs and fish ponds were initially of the same class after segmentation, but later distinguished using shape information during classification. Of course, the closer the image regions are to the eventual classes, the easier it is to assign classifications to the segmented regions.

In general, image segmentation comes in three major branches: thresholding or clustering, edge detection, and region extraction [23]. The clustering technique in image segmentation is basically the multidimensional extension of the concept of thresholding. Instead of a single dimension, two or more characteristic features are used, and each class of regions is assumed to form a distinct cluster in the multidimensional space. Clustering is necessary in cases in which there are problems not resolvable with one feature, such as intensity, but with two or more (with the incorporation of statistics or textures). Traditionally, clustering in image segmentation is supervised or manually assisted by user-specifying the thresholds or the number of clusters [24], [25]. Several unsupervised image segmentation techniques have also been proposed, such as iterative dominance clustering [26], random field models [27], fuzzy clustering [28], and maximum likelihood [29]. Some of these techniques deal with less complex scenes and others with highly textured regions, and they are not readily extensible to complex satellite imagery. A technique similar to ours was discussed in [30], which proposed a fully automated segmentation procedure using a speckle reduction filter to iteratively smooth SAR imagery as a preprocessor and then examine each pixel to determine to which peak, if any, in the now smoothed histogram it was converging. The number of peaks determined the number of classes in the image, and pixels converging to each peak were assigned to the same class. As the author pointed out, this procedure would fail to detect small classes in some cases. In addition, the technique was shown to be successful in segmenting 100 $\times$ 100 pixel images, which are relatively small in size and do not necessarily exhibit histograms with complexities found in larger images.

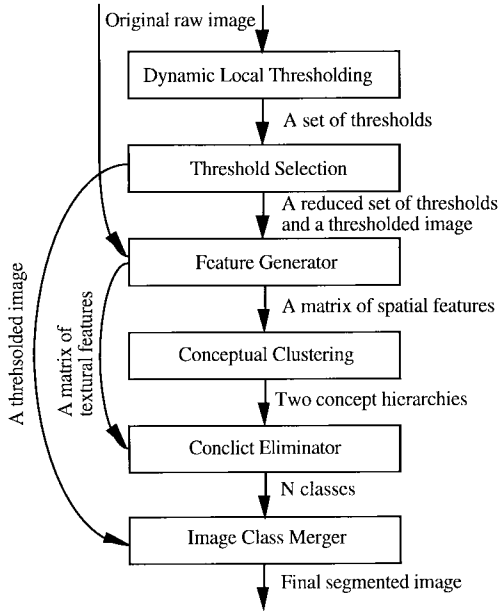


Fig. 1. Block diagram of our segmentation technique.

Fig. 1 shows the block diagram of our implementation. Dynamic local thresholding produces a set of thresholds, from which a set of *significant* thresholds are extracted by comparing their dominance measures over neighboring values. This significant thresholds set is reduced in size and more representative since trivial threshold values have now been weeded out. A preliminary segmentation of the image is produced using this set of thresholds. From this transformed image, higher order information is mined and serves as the key ingredient for our feature generation module. In this particular application, two sets of features that describe the spatial and textural characteristics of each significant threshold value are generated. Then conceptual clustering is performed to group the thresholds into different clusters based on their attached spatial attributes. This results in two concept hierarchies, depending on the order of presentation of the data (direct or reverse). To resolve the conflicts between these two hierarchies, textural attributes of the thresholds are compared to score the conflicting clusters. Clusters with better scores are preserved; those with worse scores are replaced. Finally, given the resolved hierarchy, we process the preliminarily segmented image by merging within cluster classes. In the following section, we discuss the dynamic local thresholding and, within its scope, the threshold selection procedure. Second, we for-

mulate the data preparation, preclustering module that is the feature generation stage. Then we present the application of conceptual clustering. Finally, we describe a conflict resolution strategy that examines the clustering result as a postclustering module.

III. Dynamic Local Thresholding

We have adapted the dynamic local thresholding technique from [19]. The following subsections describe our implementation of the technique. For more detail, refer to [19].

A. Regional Histogram Computation and Selection

The image is divided into overlapping regions such that each region shares 50% of the same pixels with each of its four neighboring regions and the intensity histogram of each region is calculated. For convenience, we denote a pixel belonging to region R_i as p_{ri}. Note that this pixel could belong to at most four different regions. Then, the variance of every histogram and only histograms with variance greater than the variance threshold V_t are selected for further processing. V_t is image dependent and set so that at least 25% of all histograms pass the variance test, providing the technique with a sufficient number of regions such that each region can be constrained adequately by other regions. This implicitly helps preserve global information from the image.

B. Gaussian Curve Approximation

The objective of this step is to provide the basis for optimal threshold extraction from each histogram. Strategies proposed in histogram-based segmentation and binarization, such as entropy [31], minimum error [32], relaxation-based [33], and so on, have shown robust results in handling bimodal images. These methodologies could be used as a direct process of locating the optimal threshold or as a precursor process of providing initial guesses for the Gaussian curve approximation. In our design, however, we assume the local intensity distribution is a binormal Gaussian mixture, paving the theoretical background for the maximum likelihood derivation when solving for the optimal threshold.

Before the approximation process can be performed, initial, good guesses must be obtained for the parameters of the two Gaussians: μ_1, μ_2, σ_1, and σ_2. To estimate the initial values for these four parameters, the histogram is divided at its mean μ_{ri} and the mean values of the histograms in the ranges $[0, \mu_{ri})$ and $[\mu_{ri}, K]$ are computed, respectively (K is the maximum intensity level, usually 255). The corresponding σ_1 and σ_2 are computed over those same ranges.

Using the stochastic estimation method in [44], the following procedure is derived to find a curve fitting binormal mixture $f(x)$, where $x = H_{ri}$ and H_{ri} is the histogram for region i. The goal of the curve fitting is to approximate $f(x)$ by a set of n Gaussians with

$$\hat{f}(x) = \sum_{k=1}^{n} c_k \Phi(x; \mu_k, \sigma_k) \tag{1}$$

where

$$\Phi(x; \mu_k, \sigma_k) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left[\frac{-(x-\mu_k)^2}{2\sigma_k^2}\right] \tag{2}$$

is the Gaussian distribution with mean μ_k and standard deviation σ_k. The set of mixture coefficients $\{c_k\}$ must satisfy the constraints

$$c_k \geq 0, \sum_{k=1}^{n} c_k = 1. \tag{3}$$

To estimate from the samples $x_1, x_2, \cdots, x_{N_{ri}}$ [or $g(p)$, $p \in R_i$, where $g(p)$ is the gray-level value of pixel p inside region R_i] requires the maximization of the regression function over the expected value of the log-likelihood function of $\hat{f}(x)$. This occurs when the partial derivatives of the function are zero. We solve for these partial derivatives iteratively. Hence, we obtain the following estimation steps. At the $(i+1)$th step, update

$$\mu_k(i+1) = \mu_k(i) + \rho(i) \frac{\partial}{\partial \mu_k} \ln \hat{f}(x)|_{x=x_i} \tag{4}$$

$$\omega_k(i+1) = \omega_k(i) + \rho(i) \left[\frac{\partial \tau_k(i)}{\partial \omega_k} \frac{\partial}{\partial \tau_k} \ln \hat{f}(x)\right]\Bigg|_{x=x_i} \tag{5}$$

$$\delta_k(i+1) = \delta_k(i) + \rho(i) \left[\sum_{l=k}^{n-1} \frac{\partial c_l(i)}{\partial \delta_k} \frac{\partial}{\partial c_l} \ln \hat{f}(x)\right]\Bigg|_{x=x_i} \tag{6}$$

where

$$\omega_k = \tau_k - \frac{1}{\tau_k} = \frac{1}{\sigma_k} - \sigma_k \qquad 0 < \tau_k < \infty \tag{7}$$

$$\tau_k = \frac{1}{\sigma_k} = \frac{1}{2}\left(\omega_k + \sqrt{\omega_k^2 + 4}\right) \tag{8}$$

$$\delta_k = \frac{2c_k - (1-d_k)}{c_k(1-d_k-c_k)} \qquad 0 \leq c_k \leq 1 - d_k \tag{9}$$

$$c_k = \frac{1}{2\delta_k}\left[(1-d_k)\delta_k - 2 + \sqrt{(1-d_k)^2\delta_k^2 + 4}\right] \tag{10}$$

where $\rho(i)$ is a sequence of positive numbers satisfying

$$\sum_{i=1}^{\infty} \rho(i) = \infty, \quad \sum_{i=1}^{\infty} \rho^2(i) < \infty, \quad \text{and} \quad \lim_{i\to\infty} \rho(i) = 0. \tag{11}$$

In our implementation, we define

$$\rho(0) = 0.5, \quad \rho(i) = 0.5\rho(i-1) \qquad i = 1, 2, 3, \cdots. \tag{12}$$

The approximation process is completed when either the partial derivatives are zero or when

$$\begin{aligned} \mu_k(i+1) &= \mu_k(i) \\ \omega_k(i+1) &= \omega_k(i) \\ \delta_k(i+1) &= \delta_k(i). \end{aligned} \tag{13}$$

Note that other approaches, such as those proposed by [34], may also be used to curve fit normal distributions.

C. Bimodality Filtering and Optimal Threshold Determination

Given the parameters of the approximation process, a bimodality measure is computed

$$\beta = \frac{\min_{i=[\mu_1,\mu_2]} \hat{f}(i)}{\min(\hat{f}(\mu_1), \hat{f}(\mu_2))}. \tag{14}$$

A region with β_{ri} greater than the bimodality threshold β_t (currently set at 0.8) will be filtered out from further processing. Note that a small value β_{ri} indicates that the valley separating the two modes is significant. For those regions allowed through by the low-pass filter, a single threshold t_{ri} is computed. This value minimizes the probability of misclassification, derived from the maximum likelihood method.

D. Significant Threshold Selection and Assignment

After the optimal threshold determination, we obtain a set of thresholds $T = \{t_{r1}, t_{r2}, \cdots, t_{rN_{rt}}\}$ and $0 < t_{ri} < 255$, where N_{rt} is the number of regions with a computed threshold value. The objective of this threshold selection module is to extract a set of *significant* thresholds T_S from T. In an image, each class of areas could have a range of intensities such that the representative (or significant) thresholds found from these areas could be far apart from each other. Occupying the interval between each pair of these thresholds are other threshold values compromising the intensity difference. By eliminating these values, we provide the clustering module with only the significant thresholds and without the residuals. Otherwise, the clustering module would have grouped each significant threshold and its residual neighboring values as an individual cluster and ignored the possibility of treating two significant threshold values as members of a same cluster.

To obtain T_S, a sequential histogram-tracking scheme is used to follow the histogram of thresholds

$$H_T(t) = \underset{t_i \in T}{\#_i}(t = t_i) \qquad t = 0, \cdots, 255 \tag{15}$$

where $\#_i(\bullet)$ denotes the number of occurrences of its predicate over the domain i. A *significant* threshold is a threshold value of substantial frequency and possesses a higher *coverage* than its neighbors. The scheme computes what we call the extent of t_i, $E(t_i)$, such that $E(t_i)$ is the maximum number that satisfies

$$\begin{aligned} &|H_T(t_i) - H_T(t_{i-E(t_i)})| + |H_T(t_i) - H_T(t_{i+E(t_i)})| \\ &\quad \le \alpha H_T(t_i). \end{aligned} \tag{16}$$

$E(t_i)$ is a key indicator that says how far we can walk from $H_T(t_i)$ without great changes in frequency or altitude $|H_T(t_j)|$, where $j \neq i$. $E(t_i)$ is large when $H_T(t_i)$ is in the middle of a flat surface. The attenuator α is limited to $0 \cdots 2$. If we set α high, we are looking for large structures in the histogram and vice versa. The *coverage* of t_i, $C(t_i)$, is thus defined as

$$C(t_i) = 2E(t_i) + 1. \tag{17}$$

A t_i that has high $C(t_i)$ and is not shadowed by other, more dominant threshold values will become a significant threshold. Note that our algorithm favors threshold values that situate on massive structures (plateaus, slopes, domes, valleys) rather than acute structures (gorges, peaks) in the histogram. This strategy is analogous to edge detection techniques that locate true edge pixels at the middle of edge ramps (or the zero crossings of second derivatives). We assume that sudden drops or rises in the histogram curve are not reliable and due to either noise effects in the original image or occasionally inaccurate binormal approximations (performed upon local, multimodal distributions) during the dynamic local thresholding process. Suppose T^0 is the initial threshold set and T_S the significant threshold set, initially $\emptyset$. The extraction steps of significant thresholds at kth iteration are as follows:

1) pick t_i such that $C(t_i) \ge C(t_j)$, for all $t_j \neq 0$, $t_j \in T^k$;
2) update $T^{k+1} = T^k / t_j$, where $i - E(t_i) \le j \le i + E(t_i)$;
3) update $T_S = T_S \cup t_i$;
4) if $T^{k+1} \neq \emptyset$ and $\exists (t_j \in T^{k+1} | C(t_j) \neq 0)$, then go back to step 1); otherwise exit.

This procedure guarantees that from each structure, only the most dominant threshold value will become the significant threshold.

Given the complete set of $T_S = \{T_{S1}, T_{S2}, \cdots, T_{SN_S}\}$, where N_S is the number of significant thresholds, every region with a computed threshold value is assigned a label

$$\begin{aligned} l(t_{ri}) = j \;\big|\; &|T_{Sj} - T_{ri}| \le |T_{Sk} - T_{ri}| \\ &k = 0, \cdots, N_S, T_{Sk} \in T_S. \end{aligned} \tag{18}$$

This assignment scheme uses one-dimensional (1-D) Euclidean distance as the criterion to statistically cluster nearest neighbors [35] centered at every member of T_S. As a result, we define the support of T_{Sk}

$$\gamma(T_{Sk}) = \{r_i | l(t_{ri}) = k\}, \qquad i = 0, \cdots, N_{ri}. \tag{19}$$

This support will be used in our regional interpolation implementation and textural attributes generation.

E. Regional Interpolation

Unitl now, regions with histograms failing the variance and bimodality tests have no thresholds. To fill in the missing thresholds, we interpolate T_S throughout the entire image. This regional interpolation allows the global information to be propagated, thereby imposing an implicit constraint on dramatic segmentation changes. The interpolation scheme is a concentric, weighted average of neighbors. Suppose that the region r_i resides at $\langle r_i \rangle = (m, n)$, denoting a region that is on the mth row and the nth column. The neighbors of r_i (including itself) are

$$\theta(r_i) = \{\theta(r_i, d)\} \qquad d = 0, \cdots, D \tag{20}$$

where d is the square distance and D is the maximum square distance allowed (i.e., the shortest dimension of the image minus two). Note that $\theta(r_i, 0) = r_i$. The immediate neighbors of r_i are in (21), shown at the bottom of the next page. The weighting function used is a function of d

$$w_d = \frac{D - d}{D}. \tag{22}$$

In addition, we define the utility function at region r_i

$$u_{ri} = \{u_{ri1}, u_{ri2}, \cdots, u_{riN_S}\} \tag{23}$$

such that

$$\exists u_{rim} = 1, \quad m = l(t_{ri}), \quad \forall j \neq m, \quad u_{rij} = 0. \tag{24}$$

Regardless of whether a region has been assigned a threshold, we compute the confidence that the threshold assigned to that region is correct. This confidence at each distance d is measured by

$$Q_{rij,\,d} = \sum_{m \in \theta(r_i,\,d)} w_d u_{rmj}, \qquad j = 1, \cdots, N_S. \tag{25}$$

When the accumulated confidence, $\sum_{d=1}^{d_Q} Q_{rij,d}$, exceeds the confidence threshold Q_t (set at 1.25), the interpolation terminates at $\theta(r_i, d_Q)$. This confidence measure allows an isolated threshold value to be influenced by other spatially distant threshold values while preserving the threshold value of a tightly clustered thresholds. In effect, all thresholds found in the set $\gamma(T_{Sk})$ will be interpolated such that the smoothing effect is global for thresholds of the same support but not so across the supports. The set of interpolated thresholds for a region r_i is

$$\lambda_{ri} = \{\lambda_{ri1}, \lambda_{ri2}, \cdots \lambda_{riN_S}\} \tag{26}$$

such that

$$\lambda_{rij} = \frac{\displaystyle\sum_{d=1}^{d_Q} \sum_{m \in \theta(r_i,\,d)} w_d t_m}{\displaystyle\sum_{d=1}^{d_Q} Q_{rij,\,d}}. \tag{27}$$

F. Pointwise Interpolation

To ensure continuity in the boundary points on or near the border of two neighboring regions, pointwise bilinear interpolation is performed among the center points of the regions closest to that point. The set of interpolated thresholds for a pixel p_i is

$$\lambda_{pi} = \{\lambda_{pi1}, \lambda_{pi2}, \cdots, \lambda_{piN_S}\} \tag{28}$$

such that (referring to Fig. 2)

$$\lambda_{pij} = (1-b)[(1-a)A + aB] + b[(1-a)D + aC] \tag{29}$$

such that $A, B, C, D \in \lambda_{rmj}$, where m is the four closest regions to the pixel. The points at the borders of the image are not surrounded by four regional centers. Hence, a pixel of each corner region is assigned the threshold value of the closest region center, or the region located in that corner. Pixels in the top and bottom border regions are assigned the threshold values vertically nearest to them; pixels in the left and right border regions are projected from the threshold values horizontally nearest to them.

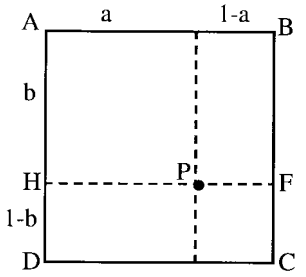


Fig. 2. Bilinear interpolation. The pixel P is assigned a value of a weighted sum of four corners A, B, C, and D.

G. N-ary Decision

Until now, every pixel in the image has a set of interpolated thresholds λ_{pi}. A pixel is assigned a classification label $c(p_i)$

$$c(p_i) = k-1, \quad \text{for } \lambda_{pij} < g(p_i) \leq \lambda_{pik},\ j = 0, \cdots, k-1. \tag{30}$$

Note that, for practical convenience, λ_{pi0} is always zero. So, the image will be segmented into $N_S + 1$ classes, with labels $0, \cdots, N_S$.

IV. Feature Generation

The objective of this module is to generate descriptors or attributes for the classes generated by the dynamic local thresholding technique. These attributes will then be attached to each significant threshold and examined by our conceptual clustering process. In our design, we utilize two sets of attributes. The first set is called the spatial attributes, derived from all $c(p_i)$. The second set is called the textural attributes, derived from a quantized version of all $g(p_i)$.

A. Spatial Attributes Generation

The spatial generator computes a *variety* curve of each significant threshold, by examining every pixel and its neighborhood—how a class label behaves spatially in relation to its eight immediate neighbors. Note that each class label is associated with a single significant threshold. To ensure one-to-one and complete matching, we create a simulated significant threshold, $T_{S(N_S+1)}$, such that class label 0 maps to T_{S1}, 1 maps to T_{S2}, $\cdots$, and N_S maps to $T_{S(N_S+1)}$. $T_{S(N_S+1)}$ is set to the maximum intensity value found in the image. Suppose a pixel p_i is located at $\langle p_i \rangle = (m, n)$ and has class label $c(p_i)$. The variety curve of T_{Sk} is defined as

$$V_{T_{Sk}} = \{V_{T_{Sk}1}, V_{T_{Sk}2}, \cdots, V_{T_{Sk}(N_S+1)}\} \tag{31}$$

such that

$$V_{T_{Sk}l} = \frac{\displaystyle\sum_{i=1}^{\#pixels} \sum_{q \in \theta(p_i,\,1)} \xi(c(p_i), T_{Sk}) * \xi(c(q), l)}{\displaystyle\sum_{i=1}^{\#pixels} \xi(c(p_i), T_{Sk})} \tag{32}$$

$$\theta(r_i, 1) = r_j | \langle r_j \rangle \in \begin{Bmatrix} (m, n+1), (m+1, n+1), (m+1, n), (m+1, n-1) \\ (m, n-1), (m-1, n-1), (m-1, n), (m-1, n+1) \end{Bmatrix} \tag{21}$$

TABLE I
EXAMPLE OF OUR SPATIAL MATRIX, GENERATED BY THE SPATIAL MODULE OF THE FEATURE GENERATOR. THE FIRST ROW AND COLUMN CARRY THE LABELS OF THE THRESHOLDS AND THEIR VALUES. TO INTERPRET, FOR EXAMPLE, THE PROBABILITY THAT T_{S1} IS A NEIGHBOR OF T_{S2} IS 0.7924

	T_{S1}=18	T_{S2}= 23	T_{S3}= 34	...	$T_{S(N_S+1)}$=172
T_{S1}= 18	0.8617	0.1035	0.0044	...	0.0000
T_{S2}= 23	0.7924	0.1199	0.0343	...	0.0000
T_{S3}= 34	0.5871	0.3319	0.0550	...	0.0001
...	...	...	...	...	...
$T_{S(N_S+1)}$= 172	0.0000	0.0002	0.0002	...	0.9035

where

$$\xi(a, b) = \begin{cases} 1, & \text{if } a = b \\ 0, & \text{if } a \neq b. \end{cases}$$

By observing the variety curve, we can tell how frequent two threshold values are in spatial proximity with each other, hinting that such two values should be merged as one. Table I shows an example of a spatial matrix. As we can see, the first two threshold values are highly related to each other in terms of spatial proximity. In addition, we can see that T_{S1} occurs at more compact regions (its high autovariety point $V_{T_{S1}T_{S1}}$), and that T_{S2} actually resides around the fringes of T_{S1}. This suggests a high confidence that the two thresholds should be grouped together, with T_{S1} as the core of the region and T_{S2} the pixels of the transition between different regions.

B. Textural Attributes Generation

Since gray-level co-occurrence matrices have been strongly theorized, widely used as a transformation of textural information from intensity values, and experimentally shown to be a superior method in texture discrimination [36], [37], we use them here to characterize these textures and serve as discriminants. The definition of gray-level co-occurrence matrices is as follows [38]. Suppose an image to be analyzed is rectangular and has N_x pixels in the horizontal direction and N_y pixels in the vertical direction. Suppose that the gray tone appearing in each pixel is quantized to N_g levels. Let $L_x = \{1, 2, \cdots, N_x\}$ be the horizontal spatial domain, $L_y = \{1, 2, \cdots, N_y\}$ be the vertical spatial domain, and $G = \{1, 2, \cdots, N_g\}$ be the set of N_g quantized intensity. The set $L_y \times L_x$ is the set of pixels of the image ordered by their row–column designations. The image I can be represented as a function that assigns some intensity in G to each pixel or pair of coordinates in $L_y \times L_x$; I: $L_y \times L_x \rightarrow G$. The texture-context information is specified by the matrix of relative frequencies P_{ij} with two neighboring pixels separated by distance δ occur on the image, one with intensity i and the other with intensity j. Such matrices of intensity co-occurrence frequencies are a function of the angular relationship and distance between the neighboring pixels. To implement the matrices, we have defined several parameters. First, we use a quantized angular interval of 45°, corresponding to a pixel's right neighbor (0°), upper right neighbor (45°), top neighbor (90°), and upper left neighbor (135°), each a member of the set Θ. Second, we use a range of distance values $\delta \in \Delta = \{0, \cdots, 32 \text{with step } 4\}$. The use of different ranges is to obtain both local and global textural information. The use of the step function is to speed up the computation process. Third, we define the size of each textural region to be 64 × 64, with the dimension twice the maximum size of δ. Finally, we use straight, uniform quantization scheme to quantize the image into $N_g = 64$ intensity levels.

In addition, we use eight second-order statistical, textural features in our study: energy, contrast, correlation, homogeneity, entropy, auto-correlation, dissimilarity, and maximum probability [38]. Thus, a region will have a textural vector

$$f_{ri} = \{f_{ri}^1, f_{ri}^2, \cdots, f_{ri}^{N_f}\} \tag{33}$$

where $N_f = 8$. Each f_{ri}^w represents one of the textural features. We use a multidisplacement co-occurrence matrix, called the mean displacement and mean orientation (MDMO) matrix, where feature measures are averaged over all orientation and distance values. Formally

$$f_{ri}^w = \frac{1}{\chi} \sum_{\delta \in \Delta} \sum_{\theta \in \Theta} f_{ri}^w(\delta, \theta) \tag{34}$$

where $\chi = |\Delta||\Theta|$ is the normalizing factor. $|\bullet|$ is the cardinality or the number of elements in the set. The MDMO implementation assumes that every matrix of a specific δ and θ is partially and accumulatively representative for the sample. Thus, averaging the features over a range of such values, although inevitably smoothing some spatial features, it does incorporate both local and global textural information constructively. To generate textural attributes, the raw, original image is quantized and divided into 64 × 64 regions. For each region, co-occurrence matrices of a combination of δ and θ are computed, yielding a total of $\chi = |\Delta||\Theta|$. From these matrices, $f_{ri} = \{f_{ri}^1, f_{ri}^2, \cdots, f_{ri}^{N_f}\}$ are generated. To associate each f_{ri} with a significant threshold T_{Sk}, we do the following. Using (18) and (19), all textural features from the regions with the same label $l(t_{ri})$ are averaged. The set of labels is a one-to-one mapping to the set of T_S, such that

$$F_{T_{Sk}} = \{F_{T_{Sk}}^1, F_{T_{Sk}}^2, \cdots, F_{T_{Sk}}^{N_f}\} \tag{35}$$

where

$$F_{T_{Sk}}^w = \frac{1}{|\gamma(T_{Sk})|} \sum_{i \in \gamma(T_{Sk})} f_{ri}^w. \tag{36}$$

Similarly to the process of spatial attributes generation, a simulated significant threshold is created and $F_{T_{S(N_S+1)}}$ is assigned to it. Table II shows an example matrix resulting from our textural attributes generation.

V. CONCEPTUAL CLUSTERING

To classify the set of significant thresholds into groups, we use an incremental concept formation strategy [39]. According to [39], much of human learning can be viewed as a succession of events from which we induce a hierarchy of concepts that summarize and organize our experience. In supervised exemplar learning, the sets of examples and counter-examples (or

TABLE II
EXAMPLE OF OUR TEXTURAL MATRIX. EACH SIGNIFICANT THRESHOLD HAS A VECTOR OF EIGHT FEATURE MEASURES. $F^1_{T_{Sk}}$ IS ENERGY, $F^2_{T_{Sk}}$ IS CONTRAST, ETC.

	$F^1_{T_{Sk}}$	$F^2_{T_{Sk}}$	$F^3_{T_{Sk}}$	$F^4_{T_{Sk}}$	$F^5_{T_{Sk}}$	$F^6_{T_{Sk}}$	$F^7_{T_{Sk}}$	$F^8_{T_{Sk}}$
T_{S1}	0.02311	412.058	-1.60890	0.17759	1.82018	894.052	15.4747	0.06670
T_{S2}	0.02789	327.036	-2.18634	0.19152	1.76583	763.403	13.5586	0.08134
T_{S3}	0.03274	349.111	-2.54839	0.20636	1.71759	988.996	13.9753	0.09618
...	...	...	...	...	...	...	...	...
$T_{S(N_S+1)}$	0.13931	211.771	-10.0216	0.39053	1.19845	2495.15	9.65390	0.32651

positive and negative samples) are used to characterize what a class is and what a class is not, respectively. In conceptual clustering, the learning process is unsupervised. The identity of each instance is not known *a priori* to the program. Thus, in order to cluster the instances into different groups or concepts, conceptual clustering derives concepts from the behaviors of the events, or characteristics of the objects. With this knowledge, the process clusters these instances by optimizing a measure of goodness criterion on the clusters of concepts. In the incremental concept formation strategy, this learning practice is incremental—every instance encountered triggers an evaluation of the current hierarchy of concepts. Specifically, this task is defined as follows. Given a sequential presentation of instances and their associated attributes, find the following:

1) clusterings that group those instances in concepts;
2) summary of each concept;
3) hierarchical organization for these concepts.

Note that incremental conceptual clustering has been used as a descriptive and predictive tool (e.g., COBWEB [40] and LABRYINTH [41]). In this paper, we use conceptual clustering as a classification tool. Each image is processed and described (with the spatial and textural attributes), presented to the clustering program, and a concept hierarchy is built. The hierarchy is refined, and it becomes the end result of our segmentation task.

The ability of conceptual clustering to perform unsupervised classification is particularly attractive. SAR images of natural scenes come in a great variety of compositions. With this self-organizing ability of conceptual clustering, we are able to highly automate the image segmentation process.

A. COBWEB/3

In our work, we use COBWEB/3 [42] that uses a probabilistic concept formation algorithm: an instance belongs *fuzzily* to a concept with a degree of certainty. Unlike deterministic approaches, such as decision trees, this fuzziness allows smoother separations among clusters and easier associations among concepts within each cluster. There are two important conditional probabilities involved. First, the predictiveness of a value v for category c is the conditional probability that an instance i will be a member of c, given that i has value v, or $P(c|a = v)$. Second, the predictability of a value v for category c is the conditional probability that an instance i will have value v, given that i is a member of c, or $P(a = v|c)$. Fig. 3 shows one example of a COBWEB/3 concept hierarchy. Each concept (or node) includes the probabilities associated

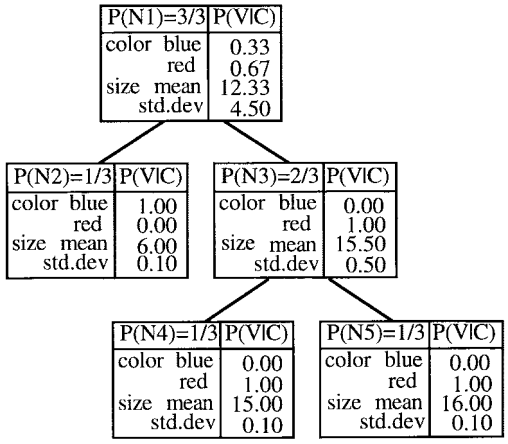


Fig. 3. Concept hierarchy generated by COBWEB/3. The attribute *color* is nominal, and the attribute *size* is assumed to be normal. N_1 is the root node. N_2, N_4, and N_5 are terminal nodes or instances.

with itself and its attribute values. For example, the root node (N_1) has an associated probability of one. Its nominal attribute *color* has a probability 0.33 of being *blue* and 0.67 of being *red*. Its normal attribute *size* has a mean of 12.33 with a standard deviation of 4.50. Nodes N_2, N_4, and N_5 are terminal nodes, representing one single instance each. Concept N_3 consists of two instances, with the summary: 1) its children have a *color* of value red with a certainty of one and 2) its children have a *size* distribution with mean of 15.50 and standard deviation of 0.50. Note that the standard deviation value for single instances is initialized to the value of $acuity = 0.1$, which is a user-adjustable parameter to avoid computation error.

The main algorithm and operational algorithms of COBWEB/3 are as shown in Figs. 4 and 5, respectively. Given an instance and the current hierarchy, it first checks whether the hierarchy consists of only the root node. If so, the new instance is incorporated into the hierarchy as a new terminal node. Otherwise, the algorithm exhausts all alternatives of incorporating the new instance into the hierarchy by placing the new instance as a stand-alone node and by merging the instance with every child of hierarchy, and assigning an evaluation score to each situation. In addition, the algorithm merges the two highest-scoring nodes and scores the merger; splits the highest-scoring nodes into two and scores the split. The configuration yielding the best score will be picked, and the concept hierarchy will be updated accordingly.

COBWEB/3 uses an evaluation function called category utility to score its configuration of concept hierarchy, favoring partitions that maximize the potential for inferring information and attempts to maximize intraclass similarity and interclass differences. It also provides a principled tradeoff between predictiveness and predictability. For any set of instances, any attribute-value pair $A_i = V_{ij}$ and any class C_k, we can compute the predictability $P(A_i = V_{ij}|C_k)$ and the predictiveness $P(C_k|A_i = V_{ij})$. These two probabilities can

```
Input:          The current node N of the concept hierarchy.
                An unclassified (attribute-value) instance I.
Results:        A concept hierarchy that classifies the instance.
Top-Level Call: Cobweb(Top-node,I).
Variables:      C, P, Q, and R are nodes in the hierarchy.
                U, V, W, and X are clustering scores

Cobweb(N,I)
  If N is a terminal node Then
    Create-new-terminals(N,I).
    Incorporate(N,I).
  Else
    Incorporate(N,I).
    For each child C of node N
      Compute the score for placing I in C.
    Let P be the node with the highest score W.
    Let R be the node with the second highest score.
    Let X be the score for placing I in a new node Q.
    Let Y be the score for merging P and R into one node.
    Let Z be the score for splitting P into its children.
    If W is the best score Then
      Cobweb(P,I)      ; place I in category P.
    Else If X is the best score Then
      Initialize Q's probabilities using I's values; place I by itself.
    Else If Y is the best score Then
      Let O be Merge(P,R,N).
      Cobweb(O,I).
    Else If Z is the best score Then
      Split(P,N).
      Cobweb(N,I).
```

Fig. 4. Main COBWEB algorithm.

```
Variables:      N, O, P, and R are nodes in the hierarchy.
                I is an unclassified instance.
                A is a normal attribute.
                V is a value of an attribute.

Incorporate(N,I)
  Update the probability of category N.
  For each attribute A in instance I
    For each value V or A
      Update the probability of V given category N.

Create-new-terminals(N,I)
  Create a new child M of node N.
  Initialize M's probabilities to those for N.
  Create a new child O of node N.
  Initialize O's probabilities using I's values.

Merge(P,R,N)
  Make O a new child of N.
  Set O's probabilities to be P and R's average.
  Remove P and R as children of node N.
  Add P and R as children of node O.
  Return O.

Split(P,N)
  Remove the child P of node N.,
  Promote the children of P to be children of N.
```

Fig. 5. Algorithms of auxiliary COBWEB operations.

then be combined into an overall measure of clustering quality

$$\sum_k \sum_i \sum_j P(A_i = V_{ij}) P(C_k | A_i = V_{ij}) P(A_i = V_{ij} | C_k). \tag{37}$$

This measure represents a tradeoff between predictability and predictiveness over the ranges of all classes (k), attributes (i), and values (j), where $P(A_i = V_{ij})$ serves as a probability-based weighting function. Using Bayes' rule, (37) can be expressed as

$$\sum_k P(C_k) \sum_i \sum_j P(A_i = V_{ij} | C_k)^2. \tag{38}$$

In [40], the authors showed that the subexpression $\sum_i \sum_j P(A_i = V_{ij}|C_k)^2$ is the expected number of attribute values of possible correct guesses given the knowledge of class C_k. This number assumes a probability matching strategy, in which we guess an attribute value with a probability equal to its probability of occurring, as found in psychological experiments. Derived from (38), [43] defined category utility as the increase in the expected number of attribute values that can be correctly guessed, given a set of n categories, over the expected number of correct guesses without such knowledge, which is $\sum_i \sum_j P(A_i = V_{ij})^2$. Subtracting this expression from (38) yields the complete formula for category utility in (39), shown at the bottom of the page, where K is the number of categories. When the terms need to be generalized for real-valued attributes, the revised evaluation function becomes

$$\Gamma_{\text{normal}} = \frac{\sum_k^K P(C_k) \sum_i^I 1/\sigma_{ik} - \sum_i^I 1/\sigma_{ip}}{K} \tag{40}$$

where I is the number of attributes, σ_{ik} is the standard deviation for a given attribute in a given class, and σ_{ip} is the standard deviation for a given attribute in the parent node. COBWEB/3 uses both nominal and continuous evaluation functions. Applying COBWEB/3 to our imagery, each instance i has the intensity attribute (T_{Si}) and the $N_S + 1$ spatial attributes ($V_{T_{Si}}$). Note that the textural attributes are not used until the conflict resolution step.

B. Modifications

We have imposed a constraint on two operations in COBWEB/3: placement of an instance into an existing cluster [algorithm Cobweb(P, I)] and the merging process [algorithm Merge(P, R, N)]. This constraint is necessary such that only thresholds in successive order are allowed to be grouped together to avoid merging a high-intensity threshold value with a low-intensity threshold value. When the set of significant thresholds T_S is generated, its members are completely ordered. Thus, a new instance i with T_{Si} is allowed to be placed into an existing cluster only if that cluster has already incorporated an instance with a value $T_{S(i-1)}$ or $T_{S(i+1)}$; similarly for merging two existing clusters. To accomplish this task, we labeled each instance with an integer. We then

$$\Gamma_{\text{nominal}} = \frac{\sum_{k=1}^K P(C_k) \sum_i \sum_j P(A_i = V_{ij}|C_k)^2 - \sum_i \sum_j P(A_i = V_{ij})^2}{K} \tag{39}$$

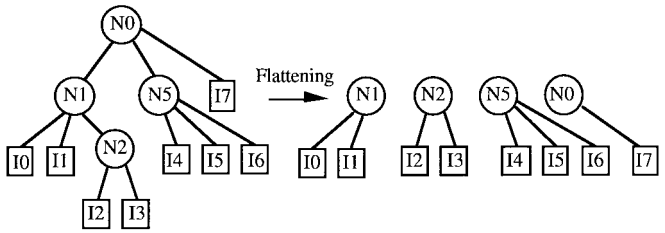


Fig. 6. Process of flattening a hierarchy tree: circles are cluster nodes; boxes are instances.

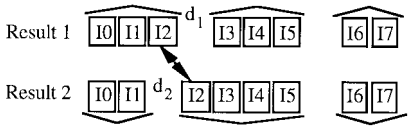


Fig. 7. Conflict that occurs between two clustering results. Result 1 is the flattened hierarchy of directly ordered instances; Result 2 is the reversely ordered. The instance I_2 is the cause of the conflict. To determine which cluster I_2 belongs to, we compute the intercluster differences (d_1 and d_2). The partition yielding a higher difference is favored.

modified the program by inserting a condition in the category utility evaluation process: if the labels are not numerically successive, then assign the lowest score to the merger.

C. Direct and Reverse Orders

COBWEB/3 examines its instances sequentially and learns the concepts incrementally. Thus, the order of the input instances is important in affecting the final structure of the concept hierarchy tree. Although COBWEB/3 provides merging and splitting operations to repartition previous hierarchy upon receiving new instances, it cannot fully eliminate the effects of early commitment of an instance to a cluster, especially if the number of instances is small. To provide a more consistent clustering result, we arrange the data in two orders: direct and reverse. The results are then merged using a conflict elimination mechanism.

VI. Conflict Elimination

This module is necessary to combine the clustering results and resolve any discrepancy. Due to the sequential treatment of instances, COBWEB/3 does not produce the same concept hierarchy, given the same set but differently ordered of instances. Our conflict eliminator first takes the two hierarchies and *flattens* them. By flattening, the higher levels of the hierarchy are discarded and only clusters are preserved. It can be viewed as a projection strategy that eliminates the *hierarchy* axis. Fig. 6 shows the process of flattening a hierarchy tree.

Here, the node N_0 is at level 0, N_1 and N_5 at level 1, and N_2 at level 2. First, the node N_2 is promoted to a higher level and separated from being a child node of N_1. Next, the terminal node I_7 is demoted such that it becomes a cousin to all other terminal nodes. As a result, the hierarchy of four levels is flattened to two. To implement this flattening scheme, we use a prefix tree traversal algorithm: the parent is visited and the subtrees are visited in order of their parents, i.e., from left to right. Denote a node as N_i and its set of children as $\kappa(N_i)$. The flattening scheme starts by initializing a basket $B_j = \emptyset$ and $j = 0$. For every node N_i encountered:

1) if $\kappa(N_i) \neq \emptyset$ and $B_j \neq \emptyset$, increment j by one and continue,;
2) if $\kappa(N_i) = \emptyset$, update $B_j = B_j \cup N_i$;
3) after all nodes have been visited, collect all baskets, and they are now flattened clusters.

Note that the hierarchies are flattened such that they are comparable as necessary during the conflict resolution phase. Once the two hierarchies are flattened, the eliminator inspects the clusters to find any conflict in their grouping of members. Fig. 7 shows an example of a conflict. The instance I_2 is associated with two different clusters generated by COBWEB/3 given the directly and reversely ordered data sets.

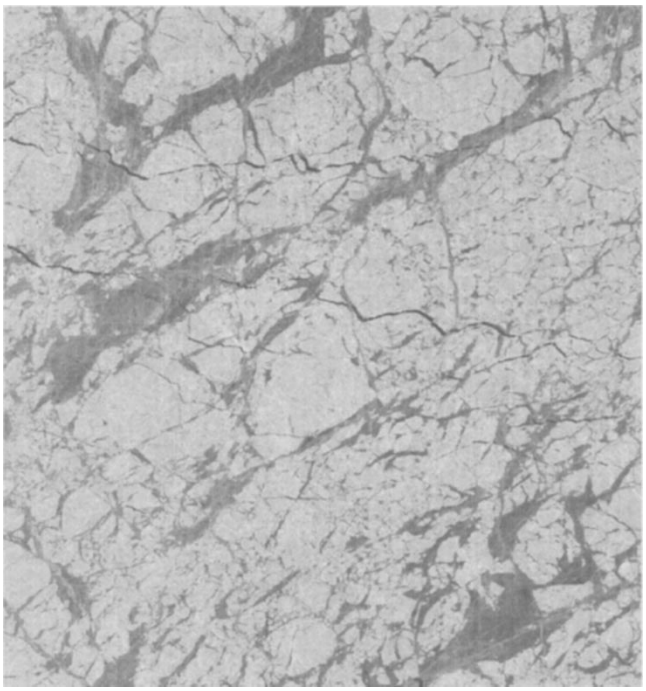


Fig. 8. Original ERS-1 SAR sea ice image (March 27, 1992, 73.46 N, 156.19 E). © ESA.

To resolve this inconsistency, the eliminator computes the average textural vector for all clusters involved in the conflict: for this example, I_0-I_1-I_2 and I_3-I_4-I_5 for Result 1, and I_0-I_1 and I_2-I_3-I_4-I_5 for Result 2. The eliminator computes a weighted average of all instances within a cluster based on the number of occurrences of the instances in the image. Suppose clusters C_1 and C_2 have average textural vectors of $\overline{v}_{C_1}$ and $\overline{v}_{C_2}$, respectively, where

$$\overline{v}_k = \{\overline{F}^1_{C_k}, \overline{F}^2_{C_k}, \cdots, \overline{F}^8_{C_k}\} \tag{41}$$

is a vector of eight textural features, such that, with reference to the support definition of (19)

$$\overline{F}^w_{C_k} = \frac{\sum_{m \in Cluster\,k} |\gamma(T_{Sm})| * F^w_{T_{Sm}}}{\sum_{m \in Cluster\,k} |\gamma(T_{Sm})|}. \tag{42}$$

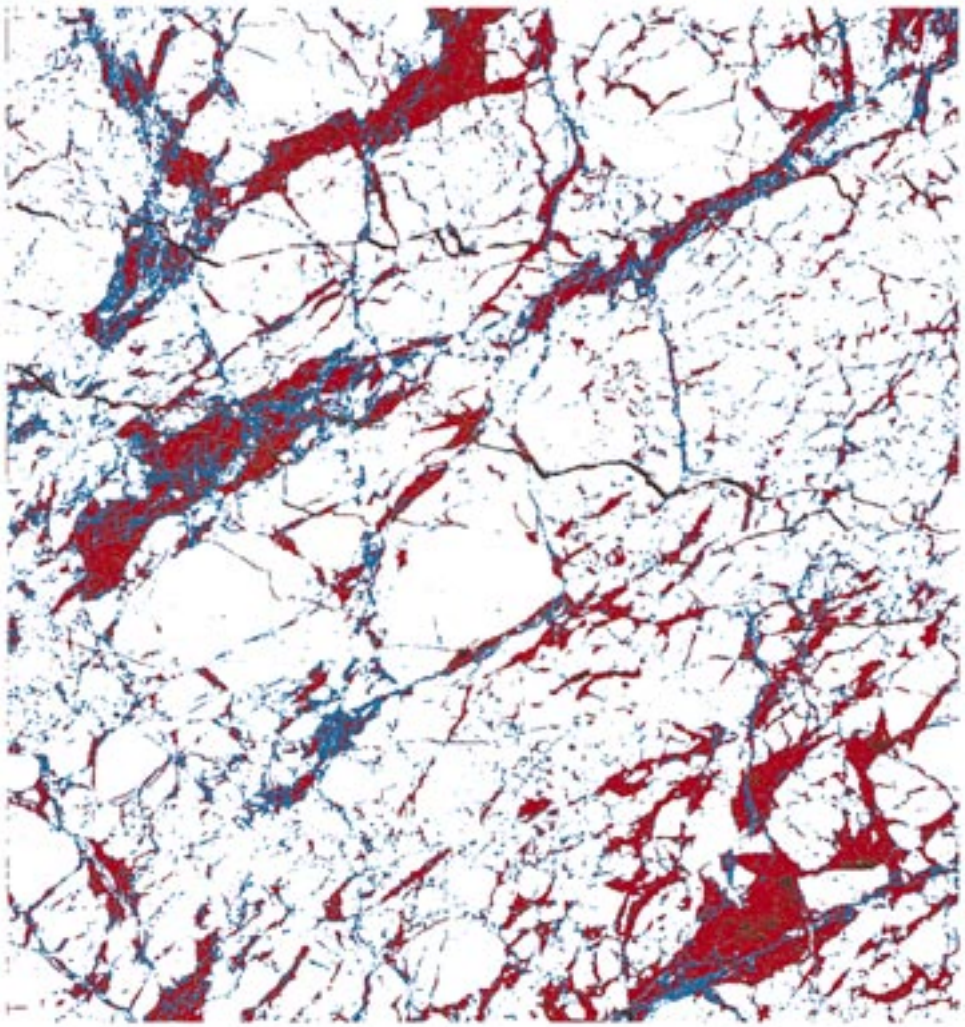


Fig. 9. Final segmentation of the image in Fig. 8. There are four classes: black, red, blue, and white.

The intercluster difference is

$$D(C_1, C_2) = \sqrt{\sum_{i=1}^{N_f} \left(\frac{\overline{F}^i_{C_1} - \overline{F}^i_{C_2}}{\max(\overline{F}^i_{C_1}, \overline{F}^i_{C_2})} \right)^2}. \quad (43)$$

Thus, a hierarchy that has a larger value of $D(C_1, C_2)$ will be preferred to maximize the difference between clusters. Other conflict situations may involve more clusters in which more computation is required, but the basic mechanism is the same. Therefore, our technique is able to resolve the conflicts and generate a consistent final clustering. Note that textural features are not used together with the spatial matrix in the first tier conceptual clustering because they would be overdiscriminatory and COBWEB/3 usually would fail to establish multi-instance clusters. Various thresholds used in our design are tabulated in (44), shown at the bottom of the next page.

VII. Results

We have successfully applied the technique to several types of satellite imagery—ERS-1 SAR, Landsat thematic mapper (TM), and NOAA advanced very high resolution radiometer (AVHRR). For all examples, we used the same parameter settings for the algorithm in (44), shown at the bottom of the page.

Fig. 8 shows an original SAR sea ice image that consists of packed ice with very dark, cutting linear structures (leads) and grayish regions (new ice or open water). Moreover, there are brighter, silky structures (possibly deformed first year ice)

Parameter	Meaning	Value
V_t	Variance threshold for the first filtering of regions (see Sect. III-A)	25%
β_t	Bimodality threshold for filtering nonbimodal regions (see Sect. III-C)	0.8
α	Attenuator for selection of the significant threshold [see (16)]	0.75
Q_t	Confidence threshold for regional interpolation (see Sect. III-E)	1.25
	Accuity value for COBWEB/3 algorithm	0.1

(44)

(a)

(b)

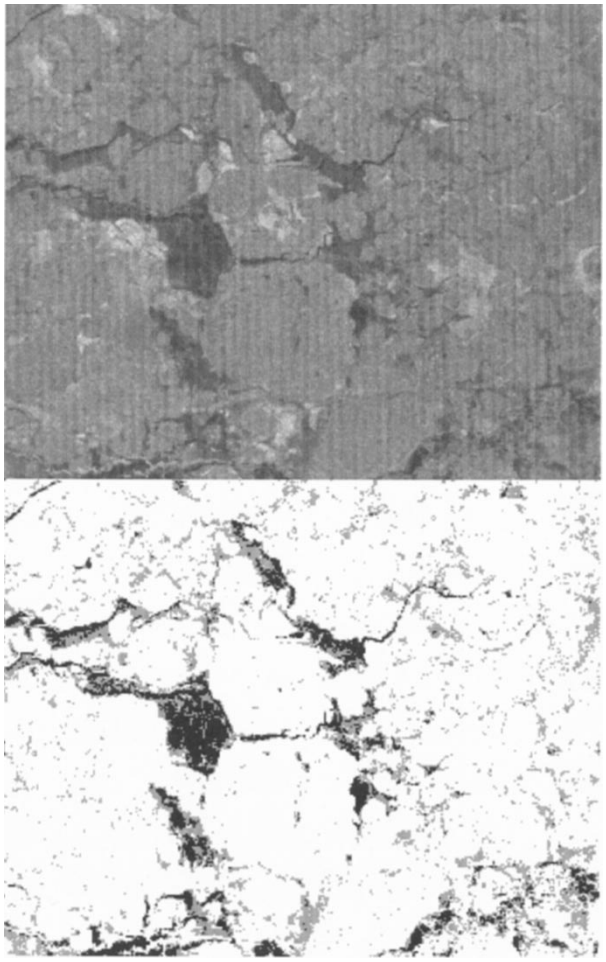
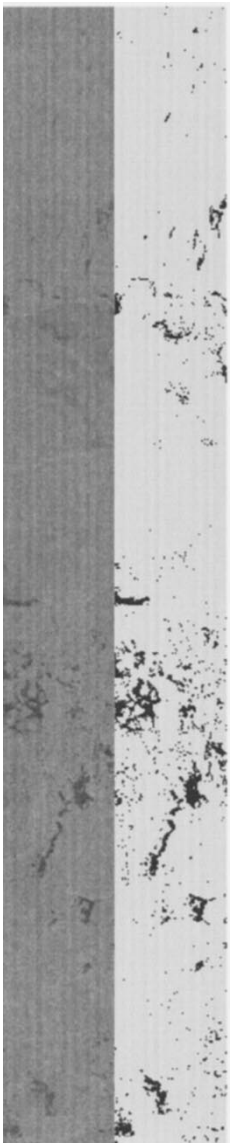


Fig. 10. (a) Portion of an original ERS-1 SAR sea ice image, taken October 11, 1995. (© ESA 1995). Classified by sea ice geophysicist to have three classes. (b) The segmentation result, with three classes: black, gray, and white.

straining within the grayish regions. So, there are essentially four classes in the image. After the preclustering module, we obtained 19 significant thresholds. The final, resolved clustering, as shown in Fig. 9, consists of four classes: I_0-I_1-I_2-I_3-I_4 (black), I_5-I_6-I_7-I_8-I_9 (red), I_{10}-I_{11}-I_{12} (blue), and I_{13}-I_{14}-I_{15}-I_{16}-I_{17}-I_{18} (white), which corresponds to the human interpretation of the image. Note that the SAR image is taken by ERS-1 satellite, on C-band, with a resolution of 100 m/pixel.

Before we explore the application of our technique to other remotely sensed imagery, we present a quantitative discussion of our technique in sea ice analysis by comparing the results to human classification provided by sea ice experts of the National Ice Center, Washington, DC. Fig. 10(a) shows a region of an original SAR sea ice image that has been classified by a sea ice expert to have more than 90% of ice coverage, which includes about 80% of old ice, about 10% of young ice, and less than 10% of new ice. Fig. 10(b) shows our segmentation result that has found three classes. The white class covers 81.45% of the image, the gray 13.33%, and the black 5.23%. Fig. 11(a) shows a strip of a SAR sea ice image that has been classified to have more than about 90% of old



(a) (b)

Fig. 11. (a) Portion of an original ERS-1 SAR sea ice image, taken October 5, 1995. (© ESA 1995). Classified by sea ice geophysicist to have two classes. (b) The segmentation result, with two classes: black and gray.

ice and about 10% of new ice, essentially having only two ice classes. Fig. 11(b) shows our segmentation result that has correctly identified only two classes. The gray class covers 93.89% of the image, the black 6.11%.

Based on discussions with the National Ice Center, the accuracy of manual sea ice classification is approximately $\pm 5\%$. In the examples of Figs. 10 and 11, our classification is well within this margin. This evaluation provides only a coverage metric for the segmentation accuracy; spatial accuracy cannot be determined as easily since the classified sea ice images are expressed only in terms of ice class concentration for an area. A visual evaluation of the spatial accuracy of the segmentation indicated that the algorithm had correctly identified spatial features, but a quantitative accuracy measurement was not possible.

The top of Fig. 12 shows Clinton Lake, a location of the northwest Douglas County in Kansas. It is the TM Band 4,

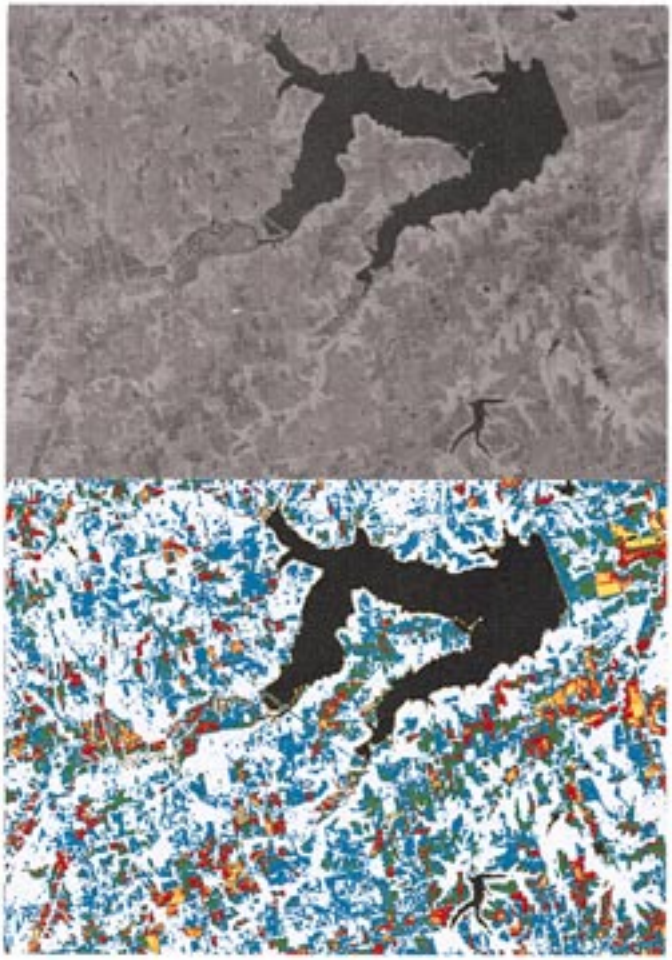


Fig. 12. (Top) Original Landsat TM image. (Bottom) The result of our segmentation: seven classes.

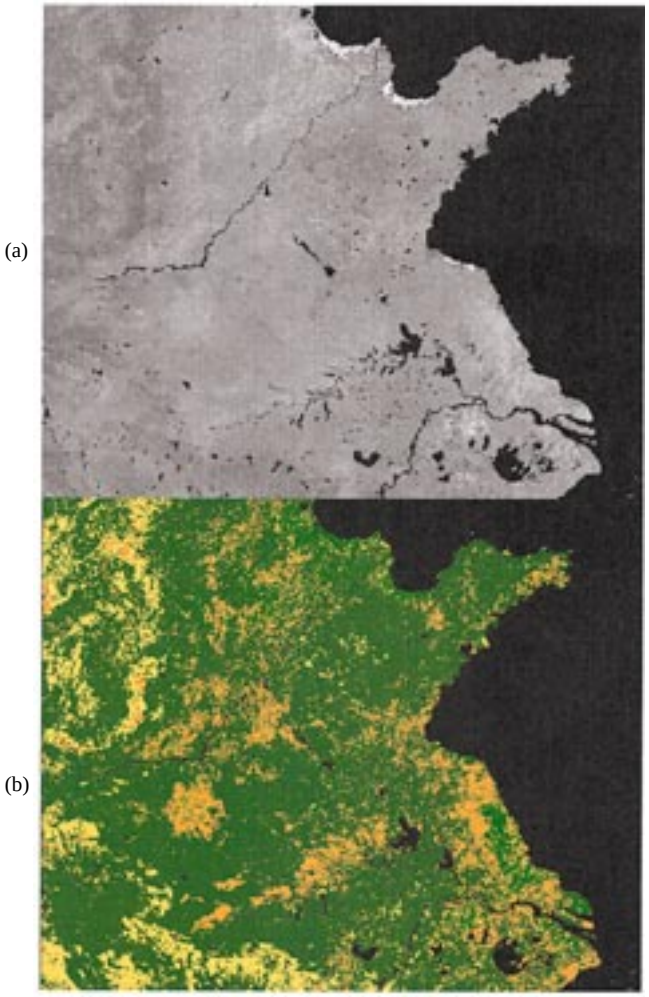


Fig. 13. (a) Original AVHRR image. (b) The result of our segmentation: six classes.

0.7–0.90-μm (near-infrared) image, with a resolution of 30 m. There are woodlands, grasslands, and other vegetation/crop land cover types. The bottom of Fig. 12 shows the segmentation results. The black class is water, identifying the body of water of Clinton Lake. The white class consists of woodland areas. The blue class pixels are grasslands. The green, red, orange, and yellow classes are different types of crop land covers or artificial grass fields, as can be inferred from their geometric shapes.

Fig. 13(a) shows the Yellow River plain, Shandong Peninsula, and the delta of Yangtze River at the south in China. It is the infrared band (0.725–1.10 μm) of AVHRR, with a resolution of 1500 m/pixel. The image was a composite of a ten-day series, taken during September 1–10, 1992. In the image, the dark regions are bodies of water (sea, rivers, and lakes). To the west of the region lies the mountain range of Taihang. To the south of the region lies the mountain range of Dabie. Fig. 13(b) shows the segmentation results. The class labels are as follows:

1) black—water;
2) bright green—saline meadow;
3) orange—temperate coniferous forest and grassland;
4) dark green—warm temperate crops (rice) and deciduous coniferous forest;
5) yellow—scrub (mountains);
6) red—possibly broad-leaved deciduous forest.

Fig. 14(a) shows the Gobi desert to the west and mountain ranges to the east. Note also the crescent-shaped region to the lower-middle area of the image caused by the nearby Yellow River. Indeed, under close examination, we can see the curving Yellow River portion. The image is processed as the ratio of band 2 (0.725–1.10 μm) over band 1 (0.58–0.68 μm) values as the vegetation index. Fig. 14(b) shows the segmentation results. The class labels are as follows:

1) black—water;
2) yellow—desert;
3) dark green—steppe grassland;
4) light blue and orange—a mixture of meadow steppe and mountain scrub.

VIII. Conclusion

We have developed and implemented an image segmentation methodology that utilizes data mining techniques. We have used data preprocessing and transformation strategies to extract effective abstractions of the data. These strategies

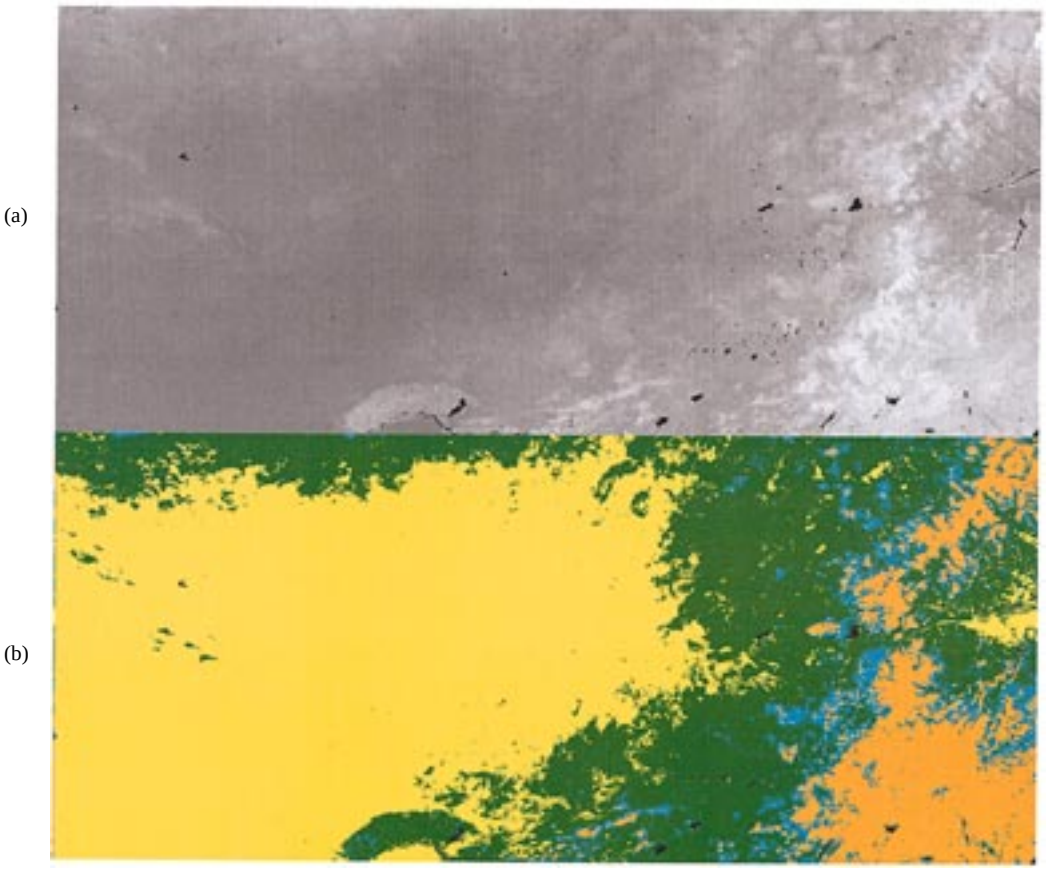


Fig. 14. (a) Original AVHRR image. (b) The result of our segmentation: five classes.

reduce the amount of data that we have to mine and implicitly reduce the noise effects since we are now looking at a higher level representation. To automatically determine the number of classes in the image, we have used conceptual clustering, a machine learning technique that has been used in the context of unsupervised discovery. The user specifies a number of parameters for the specific domain of application and target sensor data (e.g., the biomodality threshold, the value of acuity, etc.), such that the algorithm operates properly. After this initial parameter design phase, the algorithm can run unsupervised. The adaptability of the algorithm to various types of sensors and natural scenes is one of its advantages: a small set of user-defined parameters allows the same basic technique to be segment to a large set of natural scenes.

We have tested the technique on satellite imagery of natural scenes. The results show that the technique is capable of grouping similar classes together, using spatial descriptors as the conceptual clustering attributes and second-order statistical textures as the discriminator to resolve conflicts in the first tier clustering results.

Although we have dealt with single-spectral segmentation, the technique is extensible to multispectral segmentation and classification. In our opinion, our technique can be used to accommodate multispectral segmentation in two ways: 1) apply our technique to each spectrum separately and fuse the results of all spectra as a concluding stage and 2) process each spectrum similarly to extract abstractions, link all abstractions for all regions of the same location across spectrum, and apply conceptual clustering to the linked abstractions. The first approach allows the incorporation of domain knowledge in the data fusion stage, thus lessening the burden of our technique. The second approach, on the other hand, puts the data fusion task on conceptual clustering, and thus, useful domain knowledge about each spectrum might be excluded.

ACKNOWLEDGMENT

The authors would like to thank C. Bertoia of the National Ice Center, Washington, DC, for her expert classification of SAR sea ice imagery; K. Price, J. Whistler, and R. Lee of the Kansas Applied Remote Sensing Program (KARS) for providing the Landsat TM and NOAA-AVHRR imagery; and the anonymous reviewers whose input improved the presentation of our work.

REFERENCES

[1] A. J. Gow and W. B. Tucker, III, "Sea ice in the polar regions," *Polar Oceanogr., Part A: Phys. Sci.*, pp. 47–121, 1990.
[2] R. G. Onstott, "SAR and scatterometer signatures of sea ice," *Geophysical Monograph 68, Remote Sensing of Sea Ice*, F. D. Carsey, Ed. Washington, DC: Amer. Geophys. Union, pp. 73–104.
[3] R. A. Shuchman and R. G. Onstott, "Remote sensing of the polar oceans," *Polar Oceanogr., Part A: Phys. Sci.*, pp. 123–169, 1990.

[4] K. Mikkola, "A remote sensing analysis of vegetation damage around metal smelters in the Kola Peninsula, Russia," *Int. J. Remote Sensing*, vol. 17, no. 18, pp. 3675–3690, 1996.
[5] V. V. Kryuchkov, "Extreme anthropogenic load and the state of the North Taiga ecosystem," in *Air Pollutants and Acidification in Combination with Climatic Factors on Forests, Soils, and Waters in Northern Feenoscandia*, K. Kinnunen and M. Varmola, Eds. Copenhagen, Denmark: Nordic Council of Ministers, 1990, pp. 197–205.
[6] A. C. Millington, P. J. Styles, and R. W. Critchley, "Mapping forests and savannas in sub-Saharan Africa from advanced very high resolution radiometer (AVHRR) imagery," in *Forest-Savanna Boundaries*, P. A. Furley, J. Proct, and J. A. Ratter, Eds. London, U.K.: Chapman & Hall, 1992, pp. 37–62.
[7] J. R. G. Townshend, C. Justice, W. Li, C. Gurney, and J. McManus, "Global land-cover classification by remote sensing: Present capabilities and future possibilities," *Remote Sens. Environ.*, vol. 35, pp. 242–255, 1991.
[8] W. T. Liu and F. N. Kogan, "Monitoring regional draught using the vegetation condition index," *Int. J. Remote Sensing*, vol. 17, no. 14, pp. 2761–2782, 1996.
[9] J. Shukla, C. Nobre, and P. Sellers, "Amazon deforestation and climate change," *Science*, vol. 247, pp. 1322–1325, 1990.
[10] J. S. Levine, Ed., *Global Biomass Burning.* Cambridge, MA: MIT Press, 1991.
[11] H. Vare, R. Ohtonen, and J. Oksanen, "Effects of reindeer grazing on understory vegetation in dry pinus sylvestris forest," *J. Vegetat. Sci.*, vol. 6, pp. 523–530, 1995.
[12] S. Wang, D. B. Elliott, J. B. Campbell, R. W. Erich, and R. M. Haralick, "Spatial reasoning in remotely sensed data," *IEEE Trans. Geosci. Remote Sensing*, vol. GE-21, pp. 94–101, Jan. 1983.
[13] J. C. Shi, J. Dozier, and H. Rott, "Snow mapping in alpine regions with synthetic aperture radar," *IEEE Trans. Geosci. Remote Sensing*, vol. 32, pp. 152–158, Jan. 1994.
[14] J. S. Ferris and R. G. Congalton, "Satellite and geographic information system estimates of Colorado River Basin snowpack," *Photogramm. Eng. Remote Sensing*, vol. 55, pp. 1629–1635, 1989.
[15] H. Kerdiles and R. Diaz, "Mapping of the volcanic ashes from the 1991 Hudson eruption using NOAA-AVHRR data," *Int. J. Remote Sensing*, vol. 17, no. 11, pp. 1981–1995, 1996.
[16] U. M. Fayyad, S. G. Djorgovski, and N. Weir, "From digitized images to on-line catalogs: Data mining a sky survey," *AI Mag.*, vol. 17, no. 2, pp. 51–66, 1996.
[17] C. Ichoku, A. Karnieli, A. Meisels, and J. Chorowicz, "Detection of drainage channel networks on digital satellite images," *Int. J. Remote Sensing*, vol. 17, no. 9, pp. 1659–1678, 1996.
[18] X. Li, "A method to improve classification with shape information," *Int. J. Remote Sensing*, vol. 17, no. 8, pp. 1473–1481, 1996.
[19] D. Haverkamp, L.-K. Soh, and C. Tsatsoulis, "A comprehensive, automated approach to determining sea ice thickness from SAR data," *IEEE Trans. Geosci. Remote Sensing*, vol. 33, pp. 46–57, Jan. 1995.
[20] M. Ehlers, D. Greenlee, T. Smith, and J. L. Star, "Integration of remote sensing and GIS: Data and data access," *Photogramm. Eng. Remote Sensing*, vol. 57, pp. 669–676, 1991.
[21] T. Pavlidis, *Structural Pattern Recognition.* New York: Springer, 1977.
[22] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*, vol. II. Reading, MA: Addison-Wesley, 1993.
[23] K. S. Fu and J. K. Mui, "A survey of image segmentation," *Pattern Recognit.*, vol. 13, pp. 3–16, 1981.
[24] R. K. Aggarwal and J. W. Bacus, "A multi-spectral approach for scene analysis of cervical cytology smears," *J. Histochem. Cytochem.*, vol. 25, pp. 668–680, 1977.
[25] R. M. Haralick and I. Dinstein, "A spatial clustering procedure for multi-image data," *IEEE Trans. Circuits Syst.*, vol. CAS-22, pp. 440–450, 1975.
[26] M. Goldberg and S. Shlien, "A cluster scheme for multispectral images," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-8, pp. 86–92, Jan. 1978.
[27] D. K. Panjwami and G. Healey, "Markov random field models for unsupervised segmentation of textured color images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 17, pp. 939–954, Oct. 1995.
[28] H. H. Nguyen and P. Cohen, "Gibbs random fields, fuzzy clustering, and the unsupervised segmentation of textured images," *CVGIP: Graph. Models Image Processing*, vol. 55, no. 1, pp. 1–19, 1993.
[29] F. S. Cohen and Z. G. Fan, "Maximum likelihood unsupervised textured image segmentation," *CVGIP: Graph. Models Image Processing*, vol. 54, no. 3, pp. 239–251, 1992.
[30] D. M. Smith, "Speckle reduction and segmentation of synthetic aperture radar images," *Int. J. Remote Sensing*, vol. 17, no. 11, pp. 2043–2057, 1996.
[31] J. N. Kapur, P. K. Sahoo, and A. K. C. Wong, "A new method for gray-level picture thresholding using the entropy of the histogram," *Comput. Vision, Graph. Image Processing*, vol. 29, pp. 273–285, 1985.
[32] J. Kittler and J. Illingworth, "Minimum error thresholding," *Pattern Recognit.*, vol. 19, pp. 41–47, 1986.
[33] T. Ridler and S. Calvard, "Picture thresholding using an iterative selection method," *IEEE Trans. Syst., Man, Cybern.*, vol. 8, pp. 630–632, June 1978.
[34] S. Cho, R. Haralick, and S. Yi, "Improvement of Kittler and Illingworth's minimum error thresholding," *Pattern Recognit.*, vol. 22, pp. 609–617, 1989.
[35] B. W. Dasarathy, Ed., *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques.* Los Alamitos, CA: IEEE Comput. Soc., 1990.
[36] R. W. Conners and C. A. Harlow, "A theoretical comparison of texture algorithms," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-2, pp. 204–222, Mar. 1980.
[37] J. S. Weszka, C. R. Dyer, and A. Rosenfeld, "A comparative study of texture measures for terrain classification," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-6, pp. 269–285, Apr. 1976.
[38] R. M. Haralick, K. Shanmugan, and I. Dinstein, "Texture features for image classification," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, pp. 610–521, June 1973.
[39] J. H. Gennari, P. Langley, and D. Fisher, "Models of incremental concept formation," in *Machine Learning: Paradigms and Methods*, J. Carbonell, Eds. Cambridge, MA: MIT Press/Elsevier, 1990, pp. 11–61.
[40] D. H. Fisher, "Knowledge acquisition via incremental conceptual clustering," *Machine Learning*, vol. 2, pp. 139–172, 1987.
[41] K. Thompson and P. Langley, "Concept formation in structured domains," in *Concept Formation: Knowledge and Experience in Unsupervised Learning*, D. H. Fisher, M. Pazzani, and P. Langley, Eds. San Mateo, CA: Morgan Kaufmann, 1991, pp. 127–161.
[42] K. Thompson and K. McKusick, "COBWEB/3: A portable implementation," Ames Res. Center, Tech. Rep. FIA-90-6-18-2, version 1.4, 1993.
[43] M. Gluck and J. Corter, "Information, uncertainty and the utility of categories," in *Proc. 7th Annu. Conf. Cognitive Sci. Soc.*, Irvine, CA, 1985, pp. 283–287.
[44] T. Y. Young and G. Coraluppi, "Stochastic estimation of a mixture of normal density functions using an information criterion," *IEEE Trans. Inform. Theory*, vol. IT-16, pp. 258–263, May 1970.

Leen-Kiat Soh (S'91–M'98), for a photograph and biography, see this issue, p. 795.

Costas Tsatsoulis (M'88–SM'98), for a photograph and biography, see this issue, pp. 795.