The University of Kansas

Information and Telecommunication Technology Center

A Technical Report of the
Networking and Distributed Systems (NDS) Laboratory

A General Model for Designing Unsupervised Segmentation Techniques in Image Analysis

LeenKiat Soh

ITTC-FY98-TR-11810-04

August 1998

Project Sponsor:

Naval Research Laboratory

A General Model for Designing Unsupervised Segmentation Techniques in Image Analysis

Leen-Kiat Soh

Abstract

This paper describes a general model for unsupervised image segmentation. We outline the process of imagery investigation in five stages: inspector, clues generator, determiner, justifier, and mapper. We discuss in details the functions of each stage and possible solutions for its design. Also, we focus on issues such as data filtering, information extraction and abstraction, clues organization, and knowledge interpretation and verification. In addition, we apply the model to several image segmentation approaches, ranging from knowledge-based systems to clustering. This discussion shows how our model provides the framework for designing an unsupervised image segmentation technique given a chosen implementation approach. By addressing explicitly the problem of unsupervised image segmentation in a general framework, we demonstrate that our model contributes towards automation in image analysis. Finally, we show the relationships between our model and knowledge discovery in databases (KDD) and why our model could be of importance in improving automation and segmentation quality of KDD processes.

1 Introduction

The primary objective of this paper is to describe a general model for unsupervised image segmentation. Traditionally, image segmentation is a process of pixel classification; the image is segmented into subsets by assigning the individual pixels to classes (Rosenfeld and Kak 1982). In this paper, we concentrate on visually separable classes measurable in syntactic quantities such as spectral, spatial, structural, or textural descriptors. Semantic attributes such as isa-hierarchies, predicates, positional relations and ancillary data are excluded from this discussion. Usually, these semantic attributes are helpful in image classification—giving meaningful and practical labels to the segmentation classes. By our definition, classification *is* one type of segmentation but segmentation is *not* classification. Some segmentation techniques, however, are classification techniques when syntactic information is sufficient to achieve such objective.

Pavlidis (1977) defined image segmentation as follows. Suppose γ is the grid of sample points of an image, I. The set of coordinates

$$\langle i,j\rangle \Big| i = 0,2,\ldots,N_i - 1, j = 0,2,\ldots,N_j - 1,$$

where N_i is the number of rows and N_j the number of columns. Let α be a non-empty subset of γ consisting of contiguous pixels. Then a uniform predicate $\mathrm{P}(\alpha)$ is one which assigns a boolean value to α, based upon properties derived from the image $I(\langle i,j\rangle)$ for the pixels of α. Furthermore, P has the property that if β is a non-empty subset of α, then $\mathrm{P}(\alpha) = TRUE$ implies always $\mathrm{P}(\beta) = TRUE$. Given the above definition, a segmentation of the grid γ for a uniformity predicate P is a partition of γ into disjoint non-empty subsets of $\gamma_0, \gamma_1, \ldots, \gamma_{N-1}$ such that:

(1) $\bigcup_{i=0}^{N-1} \gamma_i = \gamma$,

(2) γ_i, $i = 0,1,\ldots,N-1$ is connected,

(3) $\mathrm{P}(\gamma_i) = TRUE$ for $i = 0,1,\ldots,N-1$, and

(4) $\mathrm{P}(\gamma_i \cup \gamma_j) = FALSE$ for $i \neq j$,

where γ_i and γ_j are adjacent. Zucker (1976) described the above conditions as follows. The first condition requires every pixel be in a region, indicating that the segmentation algorithm must process every pixel in the image. The second condition implies that regions must be connected. The third condition requires that all pixels within a region must satisfy a uniformity measure. The fourth condition expresses the maximality of each region in the segmentation, that is, each region is maximally uniform by itself. Note that this definition by Pavlidis (1977) does not apply to all image segmentation techniques. For example, edge detection, a segmentation branch, assigns two labels to an image: edge pixels and non-edge pixels. The set of all edge pixels is not required to satisfy the uniformity criterion, and neither is the set of all non-edge pixels.

Over the years, many surveys have been conducted in the field of image segmentation (Fu and Mui 1981, Haralick and Shapiro 1985, Pal and Pal 1993). However, none of these surveys have provided a process model for unsupervised image segmentation, which is the focus of this paper. With the increase in the amount of imagery data, ranging from satellite to photographic images, generated everyday in today's world, streamlined manual processing of images has

become unfeasible. Unsupervised image segmentation thus is an image processing area of great importance and need. By definition, an unsupervised image segmentation technique is able to determine the number of classes in the image and assign classes to the pixels automatically. However, unsupervised image segmentation is difficult to accomplish since the number of classes in an image is highly variable (especially in remotely-sensed imagery of natural scenes), and different classes may mix and impede clear separations (Soh and Tsatsoulis, to appear). In the imagery domain of highly textured regions or simple scenes, in which segmentation classes are well defined, several unsupervised image segmentation techniques have been proposed, such as iterative dominance clustering (Goldberg and Shlien 1978), random field models (Panjwami and Healey 1995, Won and Derin 1992), fuzzy clustering (Nguyen and Cohen 1993), local Bayesian (Peng and Pieczynski 1995), and maximum likelihood (Cohen and Fan 1992). These techniques are not readily extensible to complex and ill-defined images such as satellite imagery. There are other unsupervised techniques including those discussed in machine learning (Carbonell 1990, Shavlik and Dietterich 1990) and knowledge discovery in databases (Fayyad *et al.* 1996b) that can be adopted for image segmentation. Here we propose a general model for designing an application of unsupervised image segmentation that makes use of methodologies of the various disciplines.

In this paper, we outline a general model for designing such a technique. This model is called the Data Investigation Model for Unsupervised Segmentation (DIMUS). In what follows, although classification and segmentation are used interchangeably, keep in mind that the primary objective of this paper is to study image segmentation and not classification. The outline of this paper is as follows. First, we discuss the DIMUS in detail, addressing issues of its various stages. Next, we compare our model to different implementation approaches in image processing. This section demonstrates the generality and applicability of the DIMUS. Then we discuss the roles of that unsupervised segmentation can play in Knowledge Discovery in Databases (KDD) and its importance in organizing databases. Finally, we conclude the paper.

2 The Data Investigation Model for Unsupervised Segmentation (DIMUS)

The objectives of the DIMUS are as follows: (1) to enhance the distance among pixels of different classes, (2) to handle a variety of image types in terms of resolution, texture, composition, etc., within the same domain, (3) to provide a justification metric to examine its results, and (4) to determine the number of classes in the image. The first objective requires that the technique be able to increase the separations among different classes. The second objective requires that the technique's effort in unsupervised segmentation be non-trivial and the technique be general and applicable to various scenarios of the same domain. The third objective allows specialization of the technique through justification or refinement such that a set of tools can be devised to target specific data efficiently. The fourth objective is the most important—it requires that the technique be able to determine the number of classes in the database, thus discarding the need for human intervention. Figure 1 shows the five-stage model of the unsupervised image segmentation: inspector, clues generator, determiner, justifier, and mapper.

The inspector inspects the image to determine the general characteristics of the image to provide directions for the other stages in the model. The clues generator, given the knowledge of the general characteristics, scrutinizes the characteristics to generate useful properties, called *clues*. The determiner takes as inputs the clues and determines the number of classes. The justifier reviews the result of the determiner and, if necessary, corrects it. The mapper links the

clues to the original data space so that a segmentation can be performed given the class information.

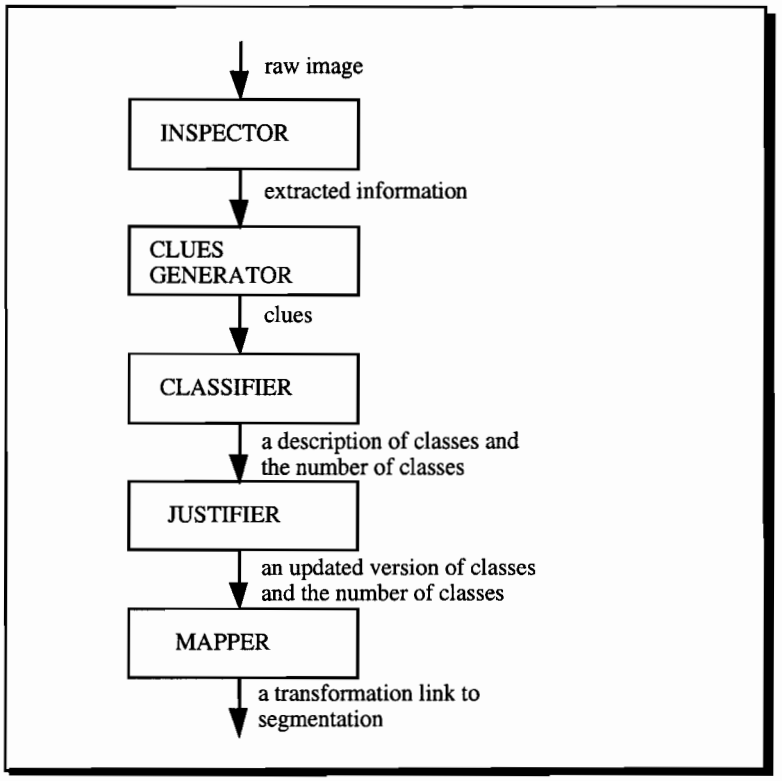

Figure 1. The Data Investigation Model for Unsupervised Segmentation (DIMUS).

2.1 Inspector

An inspector identifies the general characteristics of an image. It inspects all pixels of the image and generates an impression of the pixels. It is with this impression that the clues generator is able to generate useful clues for later stages. An inspector should be able to filter out unnecessary data, extract information, and abstract the information if necessary. In the following, we will describe these three functions and their purposes. Note that the three functions resemble the preliminary tasks of a law-enforcement inspector: weeding out non-essentials, interviewing/interrogating witnesses or suspects, extracting information, etc.

2.1.1 Data Filtering

Suppose we have an image with N_p number of pixels. A dataset in general may suffer from incomplete data, inaccurate data, unknown data, errors, and noise. For imagery data, we are usually faced with inaccurate and noisy data. For example, satellite images such as Synthetic Aperture Radar (SAR) imagery suffer from speckle noise (Raney 1985). Such speckle noise should be suppressed or removed to provide better basis for further processing. Sometimes there might be occluding factors in the image as well. These factors inhibit the true characteristics or distributions of the image from being directly obtained.

In addition to noise, poor quality imaging conditions could contribute to occluding factors. Effects are such as poor contrast, small dynamic range, intensity sloping from one side of the image to another (due to inconsistent illumination, different incidence angle, etc.), overexposure,

and underexposure. We argue that by looking at the data at a higher order, we are able to implicitly reduce the effects of occluding factors. For example, if we transform the image into a power spectrum (e.g., D'Astous and Jernigan 1984) and represent the information in the image with spikes in the frequency domain, we are able to focus on the characteristics of the image more directly than otherwise. Similarly, if we examine the histogram of the image and derive higher description of the histogram (such as parametric curve-fitting), we are able to ignore noisy peaks and valleys of the original histogram. Note that many noise-removal (Duran *et al.* 1987, Lee 1983, Mastin 1985) and image enhancement (Gauch 1992, Hummel 1977, Hummel and Zucker 1983, Paranjape 1992) techniques have been proposed and can be applied explicitly as a data filtering module.

Besides noisy and occluding data, an image could have redundant data. Data that are negligible should be filtered out from further processing to reduce computational burden and to increase interpretability. In addition, in registration tasks where two images are compared, regions not encompassed by tie-points are sometimes considered redundant and are discarded. As another example, for remotely sensed sea ice analysis, images may contain islands and inland shores which are irrelevant. These regions are usually filtered out by registering the image with its geographical coordinates and masking out the land regions from further processing.

2.1.2 Information Extraction

After data filtering, the transformed image is ready for information extraction. The goal is to extract from the data usable and reasonable information for further processing. Usable data allows the direct manipulation of the information. Reasonable data allows application of goal-specific tasks. For example, suppose we want to compare and adjust two images by their average intensity values. First, we filter the data to smooth the two images, I_1 and I_2. Then we calculate the mean value of each image. This calculation is what we mean by information extraction. The extracted information (i.e., the mean values) is usable since now we can compare the two images by, say, the difference of the two numbers. It is also reasonable since we can apply different image processing tasks based on the difference between the two numbers: for example, if I_1 is brighter than I_2, then increase the average intensity of I_2 by half of the difference; if I_2 is brighter than I_1, then decrease the average intensity of I_2 by half of the difference.

Information extraction can also be viewed as feature extraction to some degree. Suppose we are given a simple image processing task. The goal is to recognize patterns to answer the question "What are the shapes of the objects in the image?" Suppose we use a binary thresholding method to filter out non-object pixels. As a result of the data filtering phase, we obtain the object pixels. Then we apply feature extraction to extract contiguous pixels as individual regions obtaining crucial information about the image. The extracted information can be used and reasoned with in order to answer the question posed as the problem statement. Further manipulation to the regions such as boundary analysis can be treated as clues generation in the next step of our DIMUS.

To illustrate further, consider the task of building gray level co-occurrence matrices (Haralick *et al.* 1985) of a region. In this case, if we choose to approach noise removal implicitly, we can utilize a quantized co-occurrence matrix. This is because quantization has the characteristics of removing noise. As a result, the matrix is the filtered representation of the region. If we compute texture features such as energy, entropy, contrast, homogeneity, and correlation, then we are extracting information. Sometimes, the extracted information may be overwhelming in

volume. For example, if we implement the co-occurrence matrix using a range of displacement and orientation values, we could end up with a vector of large length for each texture feature. (If there are 32 displacement values and 4 orientation values, then there are 128 combinations, yielding a vector of 128 elements.) Then we need information abstraction.

2.1.3 Information Abstraction

We apply information abstraction to obtain a summary of the extracted information. This is needed only when the extracted information is deemed to be voluminous and a more concise representation is better for the clues generator at the next stage. For example, to describe a Gaussian curve, one can either plot the curve with some minute interval between each unit of the abscissa axis and the ordinate axis, or, on the other hand, completely describe a Gaussian by simply supplying the mean and the standard deviation values given the Gaussian equation. By supplying the two parameters, we have achieved information abstraction. In the example of implementing co-occurrence matrices, one can apply information abstraction by, say, computing the average of the 128 combinations.

Note that information abstraction must not deteriorate the quality of the extracted information in terms of its usable and reasonable content at a later stage. For example, suppose we are to describe a boundary of an object and the object has N_b boundary pixels. We can apply information abstraction by fitting the object boundary with a polynomial (e.g., Sklansky *et al.* 1972). If the specific task is to describe the general shape of the object, then this abstraction might be effective. However, if we are to locate the convex and concave curves of the boundary, then this abstraction might prove disastrous since the piecewise polynomial segments have resulted in reduced resolution. So one must be aware of the effects of over-abstraction on the task.

In addition, in order for the information to be reasonable, one must be careful in balancing between the aggregates and the abnormalities. For example, suppose we are to summarize a signal in a signal processing task. Suppose the signal oscillates at a constant frequency and magnitude; then we can say the waveform is operating at a certain frequency and magnitude. This is an abstraction by *total aggregation*, in which a set of higher-order measurements are taken of the data without considering any individuality. However, if the signal oscillates at a constant frequency but occasionally with significant energy bursts or changes in phase, or other abnormalities, one must now consider how to represent the waveform adequately. If we choose total aggregation, then we lose possibly valuable information in the abnormalities. If we choose *total abnormalization*—that is to describe the whole waveform in detail, then we are faced with too much information. Sometimes the abnormalities are desirable; sometimes they are not. This dilemma can be found in image processing as well. If we are to find intensity lines or ridges in homogeneous regions, then these pixels of lines or ridges are abnormalities, and they are desirable. If we are to find homogeneous regions, then these lines or ridges are undesirable abnormalities. Usually, information extraction employs partial aggregation and partial abnormalization.

Information extraction and abstraction can also be viewed as a representation problem in Artificial Intelligence (Woods 1975, Brachman 1985). We must deal with issues such as data relevance and importance, types of content property (or attributes), level of granularity of the representation of the information, and the frame problem (McCarthy and Hayes 1969).

2.2 Clues Generator

There are two essential components to segmentation: implementation approach and content property. The implementation approach serves as a mechanism that drives the set of content properties. The content property serves as a means by which pixels of the image are judged and provides the basis on which the implementation approach operates. For example, thresholding is one implementation approach, the gray levels of the image is one content property, and gray level thresholding is one image segmentation technique. In this module, the set of content properties is determined for a successful application of the DIMUS. One can derive theoretically or discover experimentally the adequate set of content properties that can increase the separation among inter-class pixels and the similarity among intra-class pixels. Also, this is a module where domain knowledge can be inserted to influence the search for a set of good content properties. For example, if we are working with highly noisy images, then we might want to use statistical or frequency domain textures as content properties since their higher-order properties are more robust than first-order statistics. Suppose we know that the pixels form distinct local neighborhoods for different classes, then we might want to try spatial analysis to measure and identify those neighborhoods. In object-background segmentation for detection of linear objects such as airplanes, one might want to use symbolic properties to locate potential airplane structures and subsequently enhance them. The selection of a set of good content properties is also a major issue in Artificial Intelligence (AI) research areas such as knowledge representation, expert systems, data mining, and machine learning. In knowledge representation, one must determine what information one should collect and represent so as to be descriptive of the event or object. In building rules or cases for an expert system, one would ask what attributes are important and useful. In data mining, we analyze the issues of how one should pre-process and transform the data into usable information from which meaningful knowledge can be extracted. Similarly, in machine learning, one focuses on certain attributes of a new instance and certain attributes that are worth learning. In image segmentation, we want to know how we should extract information and clues from the image. Should we use the original image, or should we transform the image first? For example, if we choose to use thresholding as our implementation approach, we are faced with the question of what content property (or properties) we should threshold upon.

In this module, clues are generated given the information produced by the inspector. The generation process can examine the image by following the guidelines of the extracted information, or examine the extracted information directly, or both. Note that the clues can be organized in several ways with the extracted information. For convenience, we call this organization of the extracted information and clues *clues*. In the following, we discuss two operations of the clues generator. First, we describe clues organization. Its task is to generate clues and combine them with the extracted information. Second, we describe clues refinement. Its task is to refine the clues for the consumption of the determiner.

2.2.1 Clues Organization

Clues organization is important in providing a cohesive organization of clues such that they can be effectively utilized by the determiner. The organization of the extracted information and the clues can be described in several modes of relationships. Also, how the clues are generated and dictated by the extracted information will be described as the *guided-search* and *transformed-search* strategies.

First, the extracted information can serve as a guideline for generating clues. For example, if the extracted information indicates that the image is highly textured, then a textural analysis can be applied to the image to generate textural clues. To illustrate further, suppose we implement a knowledge-based segmentation technique. The control of flow of operations is governed by several rules, triggered by matching antecedents. Each rule may correspond to a certain type of extracted information regarding the image: if the image's average contrast is very low, then increase it before analysis; if the extracted information indicates the existence of only one class in the image, then stop further processing; and so on. This in effect is a *guided-search* for clues.

Second, the extracted information can serve as the basis for clues. For example, suppose as a result of the inspector module, the extracted information is a multiresolution pyramid of the image. A tracking analysis across different resolution levels can be applied to the pyramid to identify, say, true and false edges (Sjoberg and Bergholm 1988). To illustrate further, by applying our model to the technique in (Panda and Rosenfeld 1978), the inspector module essentially created a gray level–edge value two-dimensional space as the extracted information. From this two-dimensional space, the authors generated clues which were a threshold between the dark and bright regions, a threshold dissecting the high-gradient regions, and a two-dimensional line connecting the two thresholds. This in effect is a *transformed-search* for clues because the clues were extracted from the transformed space of the original image.

Third, the extracted information can serve as a modifier for the clues. A modifier can function as a weighting factor or a probability factor. For example, suppose after the inspector, we achieve an initial segmentation of different regions. Each region has its own group of pixels. The sizes of these groups can serve as weights used in, say, relaxation to compute compatibility coefficients (Peleg and Rosenfeld 1978, Pelillo and Refice 1994). The sizes can also be treated as prior probabilities for Bayesian modeling of image classes (Yu and Berthod 1995) and initial guesses of probabilities of a multivariate Gaussian mixture.

Fourth, the extracted information can serve as an organizer to arrange the clues into a usable form for the consumption of the determiner module. There are several ways to organize the clues and the extracted information:

A Hierarchy

The extracted information can be used as parents for groups of clues. To illustrate a hierarchy application, consider different transformations of the original image as part of the tasks of the inspector module. These transformations could be of edge operators to create a list of gradient images. To each gradient image one can apply thresholding to obtain strong edges, and region filling to obtain a segmentation. Therefore, each gradient image spawns two child images—edge image and region image. As a result, a hierarchy tree is formed, and information can be passed up and down the tree to finally arrive at a consistent conclusion. Another example of hierarchy organization is associated with the multiresolution approach as in a pyramid. A pyramid is an organization of information extracted from the image. During the clues generation phase, each pyramid (or resolution) level is subjected to processes such as blob detection, histogram manipulation, and thresholding. The resulting clues are then attached to each corresponding resolution level in a hierarchy. Another form of hierarchy is with the utilization of multispectral images. Each band of spectrum corresponds to part of the extracted information, and measurements (or clues) computed correspond to each band's children in the hierarchy.

B Description

Each set of clues can be used to describe each item of the extraction information. For example, in representing textures using local extrema, suppose that the inspector module produces local minima and maxima. Then, by analyzing the neighborhood surrounding each extremum, one can derive a list or vector of properties to describe that extremum. Also, in a region growing scheme, the inspector module first conducts a fast growing of regions based on initial guesses of seeds. The clues generator then can compute properties such as homogeneity, gradient, average intensity, and size for each region. This list of descriptors or clues thus describes the extracted information. Furthermore, in edge-based segmentation technique, an inspector might generate edges as the extracted information of the image. Then, the clues generator can compute edge strength, edge length, edge connectivity, and neighborhood gradient for each edge segment such that the next module can classify true segments accurately.

C Relation

Each clue (or set of clues) and each item of the extracted information are paired (or linked) in a relational table. Suppose the clues generator generates a set of clues. Each set is a vector of N elements. By linking each element to an item of the extracted information, an $N+1$ relational table is created, which can be visualized as a $N+1$-dimensional space. For example, applying our general model to (Abutaleb 1989), the inspector module generated a 1-D space by computing the entropy of the original intensity. Then, the clues generator generated another dimension by computing the entropy of the average of the neighborhood of a pixel. Thus, linking the two, a 2-D space was achieved. The relational organization differs from the description organization in the following: The descriptors in the description organization describe an item (of the extracted information) that cannot be manipulated in further analysis because each is similar to a label. However, in the relational organization, the items of the extracted information are actually manipulatable entities.

D Channeling

In this setting, the sets of clues are for different purposes and will be fanned out to different channels. For example, suppose the extracted information consists of three classes: background, object1, and object2. After the clues have been generated for each class, the background class can be channeled to a background analysis to analyze noise, and both the object classes can be channeled to task-specific analyses. Note that channeling, as a result, separates the clues and does not treat them as a constructive conglomerate. This type of inspector–clues generator relationship usually is involved with a system of different task-specific modules, which at a later stage, may or may not fuse together all its results.

Note that, as hinted above, a set of clues can consist of several types of content properties such as frequency-based, spatial, statistical, and symbolic. Each property in turns can have several measurements. For example, suppose we use symbolic and statistical properties to describe a region. The symbolic properties may include size, shape, boundary wiggliness, etc. The statistical properties may include homogeneity, entropy, mean, standard deviation, etc.

2.2.2 *Clues Refinement*

In the clues generator module, in addition to the organization of the extracted information and clues (hierarchy, description, relation, or channeling), its tasks include transformation and selection of the organization into a format usable by the determiner in the next stage. We call those processes clues refinement. These two operations share similar objectives as information extraction and abstraction in Section 2.1 of this paper. Note that determiners may use different methodologies and strategies. They may take different sets of clues, or work better on a specific set of clues, or require certain forms of clues.

For example, suppose the task is to extract round objects from the image. Suppose the inspector and clues generator produce a group of boundaries. The determiner is a neural network to be trained to recognize the behavior of round objects. The clues fed into it then will be, say, the chain code (Freeman 1974) of each contiguous boundary pixel. However, for a large object, its boundary can be of hundreds and thousands of pixels. To feed all these boundary pixels into a neural network is not viable since it will complicate the architecture and increase the training time for network convergence. Thus, either a transformation or a selection is needed to reduce the computational burden. One option is to transform the boundary by replacing a boundary segment with a single pixel at the average Cartesian coordinates of all pixels within the segment. This is *transformation*. Another alternative is to sample the boundary by taking every n th point of the boundary. This is *selection*. Either way, the objective is to reduce the number of inputs into the neural network.

To illustrate further, suppose the inspector and the clues generator produce a description organization such that each region in the segmented image has numerical values of homogeneity, contrast, entropy, and energy. Suppose the determiner we choose to use is a symbolic determiner; let us say, a rule based system which qualifies its antecedents in the following manner: if the region's entropy is HIGH then perform action 1, etc. Thus, we need to transform the numerical values into symbolic values. Suppose each attribute is to be divided into five classes: VERY LOW, LOW, AVERAGE, HIGH, and VERY HIGH. We can use the mean of each group as the threshold for AVERAGE, while one standard-deviation away as the LOW and HIGH regions, and the two or more standard-deviations away as the VERY LOW and VERY HIGH regions. This is *statistical translation*. Another alternative is through visual judgment (e.g., Carbonell 1990) in which the regions are judged, assigned the symbolic labels (VERY LOW, LOW, etc.), and then the average measurements of the similarly-labeled regions are used to threshold each label. This is *visual translation*.

Other transformation techniques include reduction, projection, translation, convolution, and other mathematical and statistical manipulations. *Attribute reduction*, for example, allows fewer attributes to be considered; *resolution reduction*, on the other hand, allows fewer instances to be considered, as exemplified by the neural network case above. Another example in reduction is quantization where different values occupying a certain interval are treated as one value. Projection, for example, allows a multidimensional space to be analyzed in one particular dimension. Translation allows measurements to be converted into if–then rules, induction trees, constraints, ranked sequence, symbols, etc. For example, in the relational organization, suppose we have N items. We can perform a translation such that the maximum of each dimension is replaced with a symbol or a higher-than-usual number to enhance these maxima's differences from others during the classification. Convolution allows different masks to be applied to the clues. For example, suppose we have a description organization and we treat the vector

describing each item of the extracted information as a signal. Then, we can use known distribution-based masks, or 1-D spatial masks to refine the signal.

2.3 Determiner

The determiner assigns classes to the clues, and thus determining the number of classes in the image and the clues that make up each class. As we have mentioned, the responsibility of the unsupervised segmentation in the DIMUS lies with this module. The task of determining the number of classes and finding the classes falls upon the algorithm that implements the determiner. Here we describe several unsupervised segmentation techniques which have been used in knowledge discovery of databases, machine learning, and image processing.

Unsupervised segmentation distinguishes itself from image segmentation solely because of its ability of performing segmentation automatically. This ability can be derived from different disciplines. For example, ISODATA (Ball and Hall 1967, Holt *et al.* 1989) and K-Means (Hartigan and Wong 1979, Le Hegarat-Mascle *et al.* 1996) and its fuzzy variants (Hall *et al.* 1992, Lim and Lee 1990, Trivedi and Bezdek 1986) are based on numerical taxonomy; histogram smoothing (Lee and Jurkervich 1989, Smith 1996) is based on speckle noise model of remotely-sensed imagery; AutoClass (Cheeseman 1990) is based on Bayesian probability; COBWEB/3 (Thompson and McKusick 1993) is based on conceptual clustering (Kolodner 1983, Fisher 1987, Lebowitz 1987, Gennari *et al.* 1990, Thompson and Langley 1991) and incremental learning; non-linear regression (Acton 1996) is based on regularization theory (Poggio *et al.* 1985, Marroquin *et al.* 1987); multi-thresholding (O'Gorman 1994) is based on peaks in the imagery; SNOB (Wallace and Dow 1994) is based on minimum message length (Rissanen 1978, Leclerc 1989, Rissanen 1989, Cook and Holder 1994); and some are based on the self-organizing behavior of neural networks such as the adaptive resonance theory or ART (Carpenter and Grossberg 1987), the learning vector quantization (LVQ) algorithm (Kohonen 1989), and the self-organizing map (SOM) method (Kohonen 1989). Some of these segmentation techniques, such as histogram smoothing, non-linear regression, and multi-thresholding, have been tailored for imagery data; while the others for numerical data. However, bear in mind that techniques dealing with numbers are still applicable to imagery data as long as we are able to extract meaningful, representative numerical attributes from the images. This process is indeed a key stage in any processing areas: signal, speech, data, image, etc.

Some of the techniques use a similar approach that allows for their automation. The algorithm first uses an initial number of classes to find clusters of data, then evaluates the clustering based on an optimization metric, and repeats with other number of classes. Finally, the algorithm selects the number of classes that yields the best score. As a result, that some of these techniques are computationally expensive has prompted several authors to introduce assumptions, reductions, and local optimizations that deviate from the original concepts and theories.

Each of the above approaches can be used as a determiner. The basic tasks of a determiner are to analyze the clues, derive from the clues the number of classes and the composition of each class, and classify the clues accordingly. These functions can be extremely complicated and intractable when the number of dimensions (the set of content properties) is large, and a consistent classification—or determination—cannot be achieved without a concerted framework. In what follows, we describe the components of a determiner. Note that in other literature, the determiner is sometimes identified as a unsupervised technique that is able to determine the number of classes within the data and groups the data accordingly. The technique may be a

neural network, a clustering algorithm, a linear regression, a histogram-based peak-detection module, or one that is based on other methodology. To avoid confusion and to be more representative, we have chosen to use the word *determiner* instead of *classifier*, and preferred *determination* to *classification*.

2.3.1 Components of a Determiner

There are four essential components that build a determiner. First, the determination system starts with a starting condition from which it begins analyses on the clues. Second, it has a stopping condition that commands the analyses to stop automatically. Note that for many *supervised* determiner, the starting condition is provided by the user, sometimes specifically for each set of data, or the stopping condition is determined by the user's human judgment of the result. Third, it has an evaluation metric to justify inter- or intra-class operations for data maneuvers. Fourth, it has an underlying process that controls the flow of the system. Note that we use AutoClass, COBWEB/3, and SNOB as representative implementations of Bayesian clustering, conceptual clustering, and minimum message length (MML) principled clustering, respectively.

A Initial State

A determiner starts with an initial condition from which it can evolve to arrive at the goal state when a stopping condition is satisfied. Table 1 shows a list of initial states of the unsupervised systems discussed above. A popular approach that transforms a supervised segmentation technique to an unsupervised one is to use a range of initial values, process each accordingly, evaluate each result based on a devised goodness of clusters, and pick the best result. ISODATA, K-Means, AutoClass, and SNOB have taken this approach. Another popular approach is to use a pre-processor to compute and provide the initial guesses of classes such that iterations are not necessary. For example, Hartigan (1975) chose seed points by splitting variables (or content properties) with large variance to obtain initial cluster means in K-means clustering. Again, ISODATA, K-Means, and Minimum Message Length have been implemented with pre-processors.

Note that COBWEB/3 does not require specification of the number of classes. Due to its incremental learning nature, it creates a hierarchy of best evaluation without the need for the knowledge of classes *a priori*.

System	Initial State
ISODATA	The number of clusters and initial cluster points are (1) randomly selected, (2) a range of values to be iteratively processed, or (3) pre-determined using other methods.
K-Means	The number of clusters and initial cluster means are (1) randomly selected, (2) a range of values to be iteratively processed, or (3) predetermined using other methods.
Histogram Smoothing	Iterative noise-removal filtering to find the number of peaks (which corresponds to the number of classes)

Bayesian (AutoClass)	Start with the number of classes that are to be higher than the expected number of classes, and then process iteratively a range of configuration values from there
Conceptual Clustering (COBWEB/3)	Not required; simply the set of initial data instances
Nonlinear Regression	Specify control parameters in the simulated annealing process and the segment length of the piecewise constant feature
Multi-Thresholding	Specify the flatness deviation (for curve analysis) to control the number of peaks allowed to qualify
Minimum Message Length (SNOB)	Estimate the number of classes and class structures, and process a range of values
Neural Networks	In general, start with a set of initial weights for the links among nodes in the network

Table 1. Unsupervised segmentation systems and their required initial states.

B End State

An unsupervised segmentation system is able to stop on its own. Thus, it decides when the clustering process is deemed satisfactory. There are several approaches to achieve this decision making. First, one uses *convergence* of certain measures. In ISODATA and K-means, if each cluster stabilizes to a certain location, then the group of such clusters is said to have converged. In SNOB, an MML-principled system, the system stops when the total message length converges. Second, the system stops when all data points have been processed. A system of this sort usually employs incremental learning. It views each datum, absorbs the influence of that datum into the learning memory by strengthening existing information, introducing new information, or resolving conflicting information, and updates the learning memory before processing another datum. Conceptual clustering and three implementations of neural networks discussed here follow this pattern of learning. Third, the system stops when the optimal solution has been found. Usually, this type of approach involves solving the combinatorial problem via searches such as hill-climbing, simulated annealing, and other mathematical techniques. First, a function based on several variables is formulated to describe the goodness of the results; that is, it achieves an optimum when the result is best. Then, an updating scheme of the variables is derived; this defines the search space. The search ends either when the result has converged or when an extremum has been located. Table 2 shows the systems and their corresponding end states.

System	End State
ISODATA	Convergence of the average response patterns
K-Means	Convergence of the cluster means
Histogram Smoothing	Stage 1: Convergence of the root mean square change in data values after iterations of noise removal filtering Stage 2: After all data have been converged to a peak

Bayesian (AutoClass)	When the range of initial configuration values has been exhausted
Conceptual Clustering (COBWEB/3)	When all data instances have been processed; for each data instance, the search ends when all operations have been exhausted
Nonlinear Regression	When the optimal solution of the energy function has been achieved
Multi-Thresholding	When all histogram bins have been masked out
Minimum Message Length (SNOB)	Convergence of the total message length
Neural Networks	ART, LVQ, SOM: When all data instances have been processed; for each data, the search ends when all existing reference vectors have been visited

Table 2. Unsupervised segmentation systems and their end states.

C Evaluation Metric

The evaluation is necessary to provide a justification and selection for actions such as merging of two classes, splitting of one class, pruning of a class, etc., during the determination. For example, suppose we are to merge a class with one of its neighbors. One approach is to compute the similarity measure of the class with each of each of its neighbors. Such evaluation metric could be Euclidean distance, Kolmogorov-Smirnov distance (Rosenfeld and Davis 1979). Mahalanobis distance (Mahalanobis 1936), Hausdorff distance (Le Moigne and Tilton 1995), vector degree of match metric (Baraldi and Parmiggiani 1996), root mean square (RMS) error, and other hybrid variants. After evaluating each pair of classes, one picks the lowest score as the most similar pair. Another approach is to compute the effects of such merging within the system. For example, COBWEB/3 accepts the inclusion of a new instance into the concept hierarchy by propagating its occurrence throughout the tree and producing new probabilities on which a category utility function is based. SNOB calculates the resulting total message length to examine the effects of data movements on the system globally. Table 3 shows the systems and their evaluation metrics.

There are two popular measurements used to decide whether merging or splitting should take place. The measure is called the intra-group variability (or within-group, within class variability). It measures either (1) the differences between all members of the group with the mean of the group, i.e., the variance, or (2) the differences between each pair of different members in the group. If this measurement is greater than some experimentally pre-determined threshold, then the group is split into two. The second measure is called the inter-group similarity (or without-group, without class similarity). It measures either (1) the differences between the means of the two involved groups, or (2) the differences between each pair that consists of one member of a group and one member of the other group. If this measurement is lower than some experimentally pre-determined threshold, then the two groups are merged into one. There are variants to these two measurements. Indeed, content properties discussed in region merging and splitting (Horowitz and Pavlidis 1976) are suitable for this task.

The evaluation metric also allows the system to formulate an optimization strategy to achieve the best solution—the result which achieves the best score on the evaluation metric. For

example, in our example of nonlinear regression (Acton 1996), the author derived an evaluation metric called the energy function to measure the solution quality of the image, and used the simulated annealing process to obtain the optimal solution. In our example of Bayesian classification, Cheeseman *et al.* (1990) computed prior distribution of the parameters and used an integral approximation process to obtain the relative probability of a certain number of classes. This approach is more global and thus can be considered order-independent at the expense of higher computational burden, compared to those using a distance metric.

Another approach is incremental. For example, in COBWEB/3, the evaluation metric, the *category utility*, is constantly updated after processing each data point. This evaluation metric provides a principled tradeoff between specificity of a class (exclusiveness) and generality of a class (inclusiveness). In a determiner system, if a class is specific, then it is highly-informative; if a class is general, then it is descriptive and thus able to tolerate noisier data points. This utility picks at each data point the best action to perform: creating a new node, absorbing the data point into an existing cluster, merging two similar clusters, or splitting a cluster. Thus, at each data point, the solution based on the category utility score is locally optimal. However, each solution is not guaranteed to be globally optimal. Hence, this method is order-dependent. Table 3 shows the systems and their corresponding evaluation metrics.

System	Evaluation Metric
ISODATA	Euclidean distance and related intra-group variability and inter-group similarity
K-Means	Various distance metrics and related intra-group variability and inter-group similarity
Histogram Smoothing	Stage 1: 2-standard deviation (2-sigma) range statistics when locating peaks Stage 2: Directional spatial distance in the histogram when labeling pixels
Bayesian (AutoClass)	Probability of the number of classes and the prior distribution of the parameters
Conceptual Clustering (COBWEB/3)	Category utility
Nonlinear Regression	Energy function
Multi-Thresholding	The goodness of a peak of a Gaussian-based sliding profile of an image
Minimum Message Length (SNOB)	Total message length
Neural Networks	ART, LVQ: Various distance metrics SOM: Various distance metrics and spatial distance of the output nodes

Table 3. Unsupervised segmentation systems and their evaluation metrics.

D Underlying Process of Operations

A determiner has a control of flow which guides the clustering or segmentation process by determining which stage the system is in and what operations the system should be performing. The basic three steps of operations are (1) assign classes to the data, (2) evaluate the resulting configuration, and (3) perform necessary operations on the classes or data. There are two common approaches to carry out these operations. First, one can perform iterations. By limiting the number of classes within a range, the system iterates by using different initial states. Then, it picks the configuration that yields the best result to be the solution. ISODATA, K-means, AutoClass, and SNOB have used this approach. Second, a modular processing has also been used to have a pre-processor, the classification, and a post-processor. Histogram smoothing, nonlinear regression, and SNOB are modular processing systems. These systems allow the initial states to be derived during pre-processing and refinements of the end states during post-processing. Third, a one pass system such as incremental learning has been utilized. This type of systems scans each data point in sequence and the process is completed after the last data point has been examined. COBWEB/3 and several neural network architectures fall into this category. Table 4 shows the systems and their corresponding underlying process of operations.

System	Underlying Process of Operations
ISODATA	Iteration: Assign classes, evaluate classes, split and merge classes, and pick the best at the end of iteration
K-Means	Iteration: Assign classes, evaluate classes, move data from class to class, an pick the best at the end of iteration
Histogram Smoothing	Perform iterative filtering to obtain the number of classes, process each data instance to obtain association of classes
Bayesian (AutoClass)	Increase the number of classes if all resulting classes have significant probability, and vice versa, search for the parameter attribute values, approximate the relative probability of that number of classes
Conceptual Clustering (COBWEB/3)	Process each data point, evaluate each possible configuration: merging, splitting, and creation, pick the best configuration and execute it, iterate until all data instances have been processed
Nonlinear Regression	Process each data point by calculating its content property, setup the energy function, perform simulated annealing to obtain the optimal solution
Multi-Thresholding	Create histograms of horizontal and vertical runs, determine sliding profile, identify peaks, filter peaks, and threshold the data with the qualified peaks.
Minimum Message Length (SNOB)	Iteration: Assign classes, compute total message length; After convergence, perform partial assignment, splitting, merging, and pruning
Neural Networks	Process each data point, updates the weights forward and/or backward, iterate until all data instances have been processed

Table 4. Unsupervised segmentation systems and their underlying processes of operations.

For operations, normally, there are the merging of two classes, the creation of a new class, the splitting of a class into two, and the absorption of a data point into an existing class. For example, two classes are merged in COBWEB/3 if the resulting configuration of the whole concept hierarchy scores the locally best category utility. In the ART neural network implementation, if the incoming data point is found to be significantly different from the existing class vectors, a new class is created with the data point as the mean. In ISODATA, if the within-group variability of a class or cluster is found to be higher than a threshold, then the cluster is split into two at the mean. In K-means, for each data point, if the distance between itself and its parent cluster's centroid is larger than the distance between itself and another cluster, then the data point is merged with this other cluster. There are additional operations such as pruning to trim the structure off of residual classes.

2.4 Justifier

The concept of work of a justifier is different from most determination systems that we have discussed above. Strictly speaking, it does not deal with the data to make sense out of them. Instead, a justifier interprets the classes—a higher level of representation—and validates them. The interpretation is a process of establishing some sort of understanding about the classes; while validation is a process of applying knowledge to the interpretation to resolve inconsistencies and refine the determination. In the next subsection, we will describe several types of knowledge that can be used in the justification processes. Then, we will discuss the two sub-processes in greater detail.

Note that justification is sometimes necessary. This is because the determiner may not be domain- or application-specific. For general determiners such as COBWEB/3, the underlying approach and implementation are not related to a particular application domain. In addition, there are no facilities to support the (convenient) infusion of domain or application knowledge—helpful constraints or rules cannot be applied—into the determination process. Nevertheless, it is possible to incorporate the justification process into the determiner as the similarity or dissimilarity criteria. In doing so, even though we can eliminate considering useless classes and speed up the clustering process, this implementation does not have the generality and flexibility of our two-stage version. Under our modular modeling, a determiner validly identifies all classes in the data, and there may be a bank of justifiers with particular applications. This fanning-out alternative is thus more general. In the following, we will discuss several useful types of knowledge. Then, we will devote a subsection to describe a number of approaches to justification.

Before leaving this discussion, bear in mind that we must not lose sight of what the objective of this DIMUS is. Its primary task is to automatically segment a database into separate classes, based on the content properties derived from the data. This DIMUS is not designed to achieve classification. *However*, one may supplement the model with a powerful justifier by providing large amount of domain and application knowledge. Then the task of the model becomes classification: assigning data points into meaningful, semantically correct classes. This fine line between segmentation and classification is sometimes elusive or blurred: if the segmentation is powerful and effective, then the classification can be omitted; if the segmentation is preliminary, then the burden of labeling data points correctly (in terms of semantics) lies on the classification phase.

2.4.1 Types of Useful Knowledge

It is true that at this stage, from the standpoint of the determination algorithm, all classes and the number of classes are valid. However, some classes may not be correct, complete, justifiable, or desirable for a particular domain or application. To help improve the quality of determination, a justifier derives useful hints from two major types of knowledge bases: domain knowledge and application knowledge. Suppose we are given an optical character recognition (OCR) task as the domain, and suppose further that we limit our work on recognizing handwritten, Arabic as the application. Suppose a determiner system has segmented the image into N classes. The domain knowledge tells us that there should be optimally only two classes: the text and the background. Thus, we can assert a constraint that regroups the classes into two:

If the Euclidean distance between the adjacent pair of classes C_m and C_{m+1} is the largest among all adjacent pair of classes

then Merge first $m < N$ classes into C_1 and later $N - m$ classes into C_2.

Given the two classes, one may be able to refine the text regions further. For example, the letters of the Arabic alphabet consist of mostly curvilinear segments. This application-specific knowledge may be used to include image pixels which has been classified as background:

If a segment of background pixels has intensity levels close to the text **and** the segment lies on a curve extended between two nearby text regions

then Assign the segment to the text class.

As shown in the above two constraints, one may use domain and application knowledge to improve the quality of segmentation or classification before further processing such as recognition and matching.

Another type of knowledge is data-related. Suppose we have a set of clues in the description format and a particular attribute, say, A_i, is more *important* than others in differentiating classes. Note that we judge *importance* based on domain and application knowledge, not numerical or statistical implications. Now, assuming that the determiner has located N number of clusters, and two of such clusters have similar A_i, during the justification process, one may derive a constraint from the knowledge base that

If two clusters have similar A_i **and** the Euclidean distance between them is within a pre-determined threshold

then Merge the clusters as one.

With this constraint, one can assert preference and favoritism towards attribute A_i, thus involving the knowledge base. Scalar constraints can be used to justify symbolic attributes. For example, suppose that attribute *size* has possible values of *very small*, *small*, *average*, *big*, and *very big*. Suppose the determiner, based on the *size* and other attributes, has grouped several instances of different sizes (*very small* and *very big*) into a same cluster, which may not be justifiable. Thus, a scalar constraint as the following may be of use:

If a cluster has members of *very small* size and *very big* size
then Split the clusters into two.

For imagery data, to counter sporadic fragmentations and holes in classified regions, spatial constraints may be used. Suppose after determination, a region of class C_i consists of holes of another class, C_j. Some application knowledge may require the extraction of compact regions for image processing purposes, thus a constraint as the following may be of use:

If a region of class C_i contains holes of another class C_j
then Assign the holes to class C_i.

This example also illustrates the difference in domain and application knowledge, which has real-world applications in, say, sea ice image analysis. Suppose the image is of Synthetic Aperture Radar (SAR) sea ice imagery, and the region is a large multiyear ice floe. The domain knowledge tells us that during winter, the floe looks bright because of high backscatter. On the surface of the ice floe, there might be melt ponds (with low backscatter) and linear ridges (with even higher backscatter). These two features are small and create holes within the multiyear ice floe region. Under the domain knowledge, if the determiner finds three classes within this region, it is perfectly acceptable. However, if the application is to locate compact ice floes, then one *must* eliminate the holes created by the melt ponds and linear ridges. In general, domain knowledge in sea ice research provides geophysical observations regarding the intensities of ice types, textures, contexts, etc.; whereas application knowledge might focus on extracting multiyear ice or other ice types only, ice classes or other geophysical sea ice features. As illustrated, application knowledge is thus more goal-oriented than domain knowledge.

Knowledge can be incorporated into constraints or rules as shown above, and also weights or probabilities, thresholds, and algorithms.

A Weights or Probabilities

Weighting factors in equations can be based on the knowledge regarding certain types of classes. For example, suppose after numerical clustering, an equation is used to combine different attribute values in order to, say, compute the total contributing factor of all these attribute values. A case in point is the determination of different types of environment pollution sources. Suppose we are given a task to classify regions based on their pollution strengths and sources. Pollution factors such as chemical plants, farming pesticides, automobiles, factories, etc., are measured such that each region is described by an attribute vector. After determination, the cities are labeled by their patterns of pollution. Perhaps, regions of one class have higher percentages of automobile pollution than others; or regions in industrial areas consistently have higher percentages of factory-related pollution cases. Next, the domain knowledge can be used to reclassify the regions within each class (or as a whole) the total damage of the combined pollution factors. Domain knowledge provides information such as how many units of pollutants are emitted by a chemical plant normally, how many units of pollutants are emitted per vehicle, etc. This information can be used as weights in a global equation to classify the regions into different classes of *pollution load*. Similarly, we can use knowledge to derive probabilities from the numbers to refine determination.

B Thresholds

Knowledge has been implemented as thresholds in many classification or segmentation techniques. These thresholds can either be knowledge-based or empirically determined. A very common approach to justification is to merge and split classes found by the determiner. To do so, the inter-cluster or intra-cluster measurements are used. Whether a merge or a split is called for is based on these measurements—whether they are lower or higher than some thresholds.

C Algorithms

With more complexity, knowledge can also be represented as algorithms. These algorithms perform class interpretation to transform the raw class information to a level such that the knowledge can be applied. In (Canny 1986), the extraction of edge pixels was refined by first segmenting the edge pixels at a αT level, where $\alpha = 2...3$, and T is a threshold found during the determination process. Then, for justification of the determination, the author used an algorithm called *hysteresis* that promoted pixels with edge values greater than T and also pixels lying on the same contour of the existing labeling to edge pixels. As a result, the occurrences of streaking and isolated false edge pixels were reduced. Also, for example, if the domain knowledge tells us that the edges should be linear, then we could use the Hough transform (Hough 1962) to fill in the gaps between detected edge segments. In some cases, arbitrary shapes could also be imposed to detect edge pixels (Ballard 1981).

2.4.2 Class Interpretation

This step prepares the class information for interpretation and interprets it. Note that to interpret the classes, one can examine either the raw class information or the transformed class information. The raw class information consists of the number of classes and the classes, as described by their attribute vectors. The interpretation process of this type is straight forward and does not require additional processing. The transformed class information is more complicated. For example, one possible transformation is spatial relationships of the classes in the image. Contextual statistics can be computed to establish links among the classes. Based on these links, one can decide whether to merge or split classes. Some of the techniques that we have mentioned in dealing with information extraction and clues generation apply here.

2.4.3 Class Validation

This module actually performs the operations to change labels of the data points, given the interpreted information of the classes. There are three essential approaches to class validation: (1) verification, (2) refinement, and (3) conflict resolution.

A Verification

The task of this step is to either preserve or discard. It examines conflicts between the interpreted class information and the knowledge base. If the application calls for one particular class and ignores other classes, one may discard the verification process on other classes by grouping them all together. If the application is interested in analyzing only the class with the largest number of members or a certain transformed class information, then the class can be

singled out for further processing. If there is a maximum number of classes for a certain type of imagery (for example, for SAR Amazon images, there are three classes: river, forest, and swamp), then any determination with a larger number of classes should be rejected.

B Refinement

While verification is passive, refinement is active. This step modifies the determination by pruning the classes, merging similar classes (especially between one large region and its smaller neighbors), splitting a class, migration of borderline data points between neighboring classes, etc. To allow for a simpler representation, pruning is sometimes used to eliminate small, insignificant classes. Merging and splitting are common approaches as a post-determination processor to improve the compactness of classes and eliminate noise effects. As for refining the borders of neighboring classes, for example, Lim and Lee (1990) used a fuzzy C-means algorithm to refine segmentation by migrating ambiguous data points among clusters to their nearest neighboring clusters.

C Conflict Resolution

In the verification approach, the conflicts are between the domain or application knowledge and the interpreted class information. In conflict resolution, however, a conflict is between the different outputs of a determiner. An order-dependent determination system, given the same set of data but ordered differently, will yield different determination results or *configurations*. The classes will be different, and sometimes even the number of classes varies. There are several ways to go about resolving such inconsistencies or conflicts. First, one may select the best overall determination by imposing a measure of goodness on the classified configurations. One possible measure of goodness is the product of specificity and generality of the classes. Specificity requires the attribute values of each class to be unique, while generality requires each class to have as many members as possible. Second, one may use knowledge-based rules or constraints to favor one particular configuration over the other. For example, suppose the application requires the classes in the configuration be compact (spatially) or homogeneous (texturally), then the configuration with the better such measurement wins out. A variant of this approach is to selectively comb through each determination output to find the best class, and then iteratively find the second best class and so on. At each iteration, once the best class is located, data points assigned to the class are masked out from further consideration. These data points are said to have been *drafted* by the best class. Thus, the compactness or homogeneity of the remaining classes (without the drafted data points) can be recalculated for each determination output, and the process iterates until all pixels have been drafted. Third, one may pick the class by its frequency. Suppose a data point is assigned class C_i the most number of times. We thus assign it to that class. One question surfaces as to how to identify each class in different configurations since a class C_i in one configuration may not correspond to a class C_i in another configuration. In this case, one may use the centroids to determine whether the two classes are similar. Fourth, one may combine the configurations by preserving the similarity and migrating other borderline points via refinement. Each data point that has been classified similarly through out different data orders will not be re-assigned. However, for a data point that belongs to a class in one configuration, and another class in another configuration, refinements such as described above can be used to migrate it.

2.5 Mapper

After we have justified the classes and updated the number of classes, we have segmented the transformed image space. At this mapper stage, we perform the actual segmentation on the imagery data. After the inspection and clues generation, the clues (onto which the determination and justification are applied) are some content properties transformed from the original data. Thus, the class information may not be readily applicable to segmentation on the original data, if that is the goal. In some segmentation, the objective is to determine the number of classes in an image. Thus, a mapper is not necessary. On the other hand, in addition to the determination of the number of classes, some segmentation tasks require the identification of the classes. Such a process includes localization and description. Localization tells us where each class occupies in the multidimensional space. Description tells us how each class behaves in the multidimensional space. Again, these tasks do not require the service of a mapper. In image segmentation, however, a mapper is often needed to perform the segmentation—assigning classes to image pixels based on the *mapped* class information.

The necessity for a mapper depends on the roles of the clues as labeling factors. Denote the clue that describes a data item x_i as C_i. Note that a data item could be a region, a segment, a pixel, a number, or a feature of other types. Further, C_i is a vector of attribute values $\{a_{i,1}, a_{i,2}, \cdots, a_{i,N_A}\}$, and N_A is the number of attributes. A mapper is required when there exists a partition of the vector C_i as the *primary clues* and the rest as the *secondary clues*. We denote the set of the primary clues of C_i as $\lambda(C_i)$, such that $\lambda(C_i) \in C_i$, the set of the secondary clues as $\tau(C_i)$, such that $\lambda(C_i) \cap \tau(C_i) = \varnothing$ and $\lambda(C_i) \cup \tau(C_i) = C_i$. Formally, a mapper is required when $\lambda(C_i) \neq \varnothing$ and $\tau(C_i) \neq \varnothing$. We define a set of attributes as the primary clues, $\lambda(C_i)$, when the data can be segmented without ambiguity using this set of attributes alone. In other words, leader attributes can completely segment a database. For example, in an image, if the *intensity* of the pixels can be used to generate consistent segmentation, then the attribute *intensity* is a primary clue. Then why do we need the secondary clues? There are two situations that require the usage of a secondary clue: refinement and determination of the primary clue. For refinement, a secondary clue is used to assign borderline data items into classes determined dominantly by the attributes in the $\lambda(C_i)$. To understand the second case, one must realize that even though a database can be segmented by using the primary clue alone, it is not guaranteed that the classes can be found using the $\lambda(C_i)$ alone. For example, suppose we are given a task to find out how many classes of different sinusoid signals there are in a database. Each signal has attributes average magnitude and frequency. In this case, assume that the signals can be completely segmented using magnitude and frequency alone. So, these two attributes become the primary clues. However, is a signal of magnitude 1 dB and frequency of 50 Hz within the same class of a signal of magnitude 1.5 dB and frequency of 46 Hz? How much tolerance do we give to a class to include different signals? We need more information about the database and more information about the signals. So one may derive signal-to-noise ratios (SNRs), variance of the magnitudes, maximum peaks, phase, etc. from the signals to classify the magnitude and frequency into groups. Consider image segmentation where each pixel is a data item. Assume that the image can completely be segmented using intensities alone. Thus, the attribute intensity is the primary clue. But in an 8-bit image with 256 intensity levels, how do we find classes based on simply the

intensity? We need an additional basis that tells us, for example, to group intensity levels 0 to 157 into one class, and intensity levels 158 to 255 to another class. This additional basis is provided by the set of content properties, and, more specifically, by the secondary clues.

Our mapper performs two tasks: map the class information to the dimension of the attributes of the primary clues, and segment the data with the attributes of the primary clues. One solution of this mapping behavior is *hierarchical reduction*—reducing as much as possible different attributes to one single *focus point* (for example, the intensity) and then applying secondary discriminations. If there is no focus point found for the reduction, we will resort to a *focus complex* to which more than one descriptions of the data are used. This iterative process allows the primary clue set to be examined to filter out negligible attributes. Another mapping behavior is to ensure that every data item is described with all attributes of the primary clues. Suppose the data item is a single point, be it a pixel in an image. In addition, suppose a textural measurement has been computed of a region of pixels. Thus, a single point within this region does not have a *direct* textural measurement. If we segment the image using the textural attribute, then we must label each pixel with some sort of textural value. If the regions are overlapping, a pixel could then be assigned the average of textural values of the regions that it is a part of; if the regions are non-overlapping, a pixel could be given the textural value of the region that contains it; etc. As a result, all pixels can be labeled.

3 Application of the DIMUS to Implementation Approaches

As mentioned above, a segmentation techniques has two components. One of them is the implementation approach. There are generally two types of implementation approaches: formal and informal. Formal approaches involve mathematical, statistical, probabilistic, or fuzzy modeling of the image, including the signal and noise components. These are approaches such as estimation via maximization, stochastic relaxation, and fuzzy sets. They employ strong theoretical foundations such as those of Gaussian, Bayesian, Markov, etc., and allow theoretical manipulations that treat signal and noise formally. Informal approaches usually involve raw handling of the image based more on common sense and experimental characteristics. These are approaches such as knowledge-based systems, neural networks, split and merge, multiresolution and pyramid, region growing, thresholding, and clustering. They use intuitive, logical impression of the image (and its domain) and, through experiments, develop processing quality informally. Note that both types of approaches sometimes work hand in hand, and sometimes pre- or post-process each other.

This section describes the different approaches in the framework of the DIMUS. As a result, we show that the DIMUS is general and applicable for a variety of image segmentation approaches.

3.1 Knowledge-Based Systems

Knowledge-based segmentation techniques usually evolve around three phases. First, a pre-processing stage that initially segments the image. Second, a feature extraction stage that computes desirable properties of the features found in the segmented image. And finally, a reclassification system applied domain knowledge based on the properties to improve the segmentation. Some knowledge-based segmentation techniques in the literature include (Binford 1982, Wilkinson and Megier 1990, Ton *et al.* 1991, Wang 1993, Haverkamp *et al.* 1994, Haverkamp *et al.* 1995). For this implementation approach, the inspector module can function as

described in Section 2.1 of this paper. Its clues generator is the module that extracts attributes from the data, which can be numerical, symbolic, or imagery. These clues will be subjected to refinement such as translation and transformation. The determiner is then a knowledge-based system, which can be a rule-based, or a case-based, or a model-based system. Since if we have chosen to use a knowledge-based system, this indicates that we are able to gather strong knowledge regarding the classes in the data. Thus, the determiner should be able to determine the number of classes in the data and the composition of each class. As we have discussed, justification is needed for domain- and application-specific tasks and may not be necessary. For a knowledge-based system, the mapper's task is to relate the final, stable output of the system back to the data. For example, suppose each input to the determiner is a region with different properties. Suppose the determiner is a rule- based system, and labeling rules are fired when triggered by certain combinations of properties. This results in a labeling of each region, and the number of classes. The mapper then uses this class information to process the image, assigning pixels of each region with the correct label.

3.2 Neural Networks

The inspector and clues generator modules of this approach follow the guidelines outlined in Section 2 of this paper. The clues generator is used to produce an input vector for each data instance. However, to satisfy the requirements of our determiner, the neural network design considered must be of types such as ART, LVQ, and SOM, as discussed above. The justification and mapping processes then proceed accordingly.

3.3 Estimation via Optimization

This methodology takes pixels of an image as observations e of certain dataset, D, given a known probabilistic or statistical hypotheses set, h, and estimates the parameters p from the data $D = e$. There are basically two general categories of such estimation. First, the maximum likelihood approach estimates by maximizing the probability $D = e$ given $p = h$. Second, the maximum *a posteriori* (MAP) approach estimates $p = h$ for which the posterior probability of h, given $D = e$ is maximum. The inspector in this approach models the image or the data using either Bayesian (Geman and Geman 1984), Gibbs (Derin and Cole 1986), Gaussian (Hassner and Sklansky 1980), or Markov fields (Derin *et al.* 1984). Then, the clues generator extracts from the model essential parameters and descriptors. A determiner of this sort looks at the model-based descriptors and derives the number of classes from them. Given the justified class information, the mapper can then invoke the optimization process. Through optimization, pixel classification can be finalized. Note that in this case, the mapper performs the segmentation by following an optimization algorithm, such as simulated annealing (Kirpatrick *et al.* 1983), upon the supplied content properties. We also assume that the optimization algorithm is initialized with the number of classes in the image.

3.4 Regularization Theory

Assuming that an image is corrupted by noise and blur, segmentation information can be derived by imposing continuity and conformance constraints in the context of reconstructing an image from pixels. By regulating this ill-posed problem with such constraints is the basis of the regularization theory (Torre and Poggio 1986, Marroquin *et al.* 1987). Examples of this

regularization theory approach include the scales determination for edge detection proposed by Jeong and Kim (1992) and the non-linear regression, unsupervised segmentation technique described by Acton (1996). First, the inspector is responsible of transforming the image into a usable and reasonable platform. Then, the clues generator extracts content properties that build the energy function. Given this energy function, the task of the determiner is simplified to optimization. This optimization process thus must be able to differentiate and label pixels. The justifier and mapper modules then proceed accordingly if necessary.

3.5 Mathematical Morphology

The language of mathematical morphology is set theory. Sets in mathematical morphology represent the shapes of objects in an image. With its definitions in set theory, many operations such as dilation, erosion, opening, closing, region filling, edge extraction, thinning, and thickening can be used to improve the quality of image segmentation. The original work was developed by Matheron (1975) and later extended by Serra (1982, 1986). Here, we describe one example of using mathematical morphology in image segmentation, according to the DIMUS framework. Consider the edge detection approach that uses multiple versions of blurred images and multiple window sizes. The output of the inspector then is the blurring resolutions created by applying morphological operators of different sizes to the image. The clues generator then create dilated and eroded images from the resolutions. These clues are then fed into a determiner to determine the number of classes and the make-up of the classes. Justification is inserted if called for. Finally, the mapper uses the class information to label the original image, possibly linking the different resolutions together.

3.6 Relaxation

The goal of relaxation (e.g., Hummel and Zucker 1983, Kittler and illingworth 1985) is to improve the quality of image segmentation mainly by incorporating the knowledge of the neighboring areas to improve the classification confidence of the area of interest. It provides globality during the initial segmentation phase and locality during the iterative re-assignment phase. It is able to achieve global consistency through a parallel exploitation of contextual information, which is expressed quantitatively in terms of a set of compatibility coefficients (Peleg and Rosenfeld 1978, Pelillo and Refice 1994). For this approach, the inspector carries out the necessary data filtering, information extraction, and information abstraction tasks. Then, the clues generator derives confidence calculations and compatibility measures for every pixel in the image. The determiner then takes in these descriptors to determine the number of classes in the image. In many relaxation technique, the number of classes is either pre-determined or ignored. When ignored, the segmentation takes place locally without a global restriction. This is sometimes desirable and practical if the image is highly homogeneous. For noisy images, relaxation should be guided to speed up its convergence by preventing over-fragmentation. Finally, given the class information, the mapper then performs the relaxation. Thus, in this case, we have assigned the mapper a vital role. Its objective remains the same: linking the class information back to the image. However, the process involved is generally more significant.

3.7 Split and Merge

Splitting is done by iteratively partitioning an image into several segments until each region satisfies a homogeneity test. On the other hand, merging is done by merging regions together if

the segments satisfy a homogeneity test. The split and merge (Horowitz and Pavlidis 1976) approach has been implemented with numerous modifications over the years (Brice and Fennema 1970, Feldman and Yakimovsky 1974, Chen and Pavlidis 1979, Beaulieu and Goldberg 1989, Pavlidis and Liow 1990, Le Moigne and Tilton 1995). The inspector conducts the split and merge procedure on the image, dividing the image into regions based on some homogeneity criterion. Then, the clues generator computes, for each region, a vector of content properties. Using the description organization, the input is fed into the determiner. The justifier and mapper modules follow accordingly if necessary.

3.8 Multiresolution and Pyramid

Image analysis at a fine resolution yields noise and unnecessary details and at a coarse level distorts local deviations. Therefore, multiple scales of resolution are considered to counter these two problems. The representation of multiple scales of resolution is usually fulfilled by a pyramid structure. At the highest resolution level of the pyramid is the original image. In general, the second highest resolution level is the shrunk image by a certain scale k, and so on. For a more detailed treatment of pyramids, please refer to (Rosenfeld 1984). Some applications of multiresolution in segmentation include (Tanimoto and Pavlidis 1975, Bergholm 1981, Hong and Shneier 1984, Meer 1989, Bouman and Liu 1991, Muzzolini *et al.* 1993). For this approach, the inspector builds the pyramid of the image. Then, the clues generator computes the set of content properties, based on the resolution levels and other transformations. This multiresolution clues are used in the classification to determine the number of classes. Finally, the justifier and mapper modules follow accordingly if necessary.

3.9 Fuzzy Sets

The idea of using fuzzy sets (Zadeh 1965) in classifying images stems from the fact that different classes of objects co-exist spatially. For example, an aerial image of an urban area could be an area of buildings and trees and so could a suburban area. The distinction between the two areas is the proportion of buildings and trees in the mixtures. Several applications of fuzzy sets in image processing include (Pal *et al.* 1983, Murthy and pal 1990, Pedrycz 1990, Caillol *et al.* 1993). An example of using fuzzy sets in data investigation is as follows. First, the inspector perform data filtering and information extraction on the image. Then, given the information, the clues generator extracts content properties. The clues refinement tasks involve visual or statistical translation to obtain fuzzy membership curves for data instances. Then, we may use the Fuzzy C-Means algorithm mentioned in Section 2 of this paper to classify the data into different classes. Finally, the justifier and mapper modules follow accordingly if necessary.

3.10 Region Growing

The basic idea of region growing is as follows: a seed is chosen, and then a region is grown from the seed. The growing process assigns the same label to a neighboring point of a seed if the neighbor is similar to the seed. The similarity criterion is a measurement of distance in terms of some property such as intensity, homogeneity, convergence, etc. Several region growing applications in the literature are (Haralck and Kelly 1969, Jarvis and Patrick 1973, Haralick and Shapiro 1985, Gauch and Pizer 1993). First, the inspector locates seeds for growing. This might involve data filtering to remove noise and information extraction to pinpoint the potential seeds. Then, the clues generator grows the seeds until all pixels in the image have been assigned in

contiguous regions. Furthermore, the clues generator also computes different attributes for each region. Similar to split and merge, these regions, each attached with a vector of descriptors, are fed into the determiner. Finally, the justifier and mapper modules follow accordingly if necessary. A popular justifier approach is to merge neighboring regions together to improve compactness and increase boundary strength.

3.11 Edge Detection

A possible implementation of this approach is to use the inspector to compute the gradient of the image and extract the edges. From these edges, further refinement is performed to ensure *closedness*, and region filling is thus applied by the clues generator. In this case, for most image segmentation work, the inspector and the clues generator have completed the work. However, for unsupervised image segmentation, we are interested in the number of classes in the image—that similar regions should be considered as one class. As a result, the clues generator proceeds to generate attributes from each region, and the determiner uses the clues to determine the number of classes. This is one unique characteristic and strength of our DIMUS, motivated by its objective of *unsupervised segmentation.* Finally, the justifier and mapper modules follow accordingly if necessary.

3.12 Thresholding

The inspector of this approach transforms the image and the clues generator extracts clues from the transformation. Then, given the transformation, the determiner produces the number of classes and also the number of corresponding thresholds. The justifier can merge the thresholds based on domain- or application-specific constraints. Finally, the mapper performs the thresholding task, which could be multidimensional. For more detail of an implementation of DIMUS using thresholding, see (Soh 1998).

3.13 Clustering

Clustering is the grouping of similar objects in a multidimensional space (Fisher 1987, Goldberg and Shlien 1978, Nguyen and Cohen 1993, Postaire *et al.* 1993, Thopson and McKusick 1993, Uchiyama and Arbib 1994, Wong and Posner 1993, Wu and Leahy 1993). This implementation approach provides the most natural setting for the implementation of DIMUS. In general, the issues addressed by clustering resemble the inspector, clues generator, and the determiner modules. Given a set of content properties describing an object—a pixel, a region, a segment, or other image elements, a clustering algorithm categorizes each object into different groups, with members of the same group similar to each other, and objects from different groups dissimilar to each other. As long as the determiner is an unsupervised clustering algorithm, we can fully describe such an approach in the framework of the DIMUS.

4 Unsupervised Segmentation and KDD

Segmentation is a process of data classification in which data instances are divided into subsets. Up to now, there has been no work in defining segmentation in databases and no discussion of its importance in data mining. In this paper, we define what segmentation is in databases, identify the differences between this approach and other conventional methodologies, argue why it is

essential in knowledge discovery in databases (KDD), present a general model for unsupervised segmentation of databases, and describe two applications designed and implemented based on the model. In the following discussions, we view KDD as a process that includes data management, warehousing, data mining and analysis, and interpretation. Meanwhile, data mining is considered as the module that performs the actual knowledge extraction (or pattern recognition) part of KDD. These viewpoints follow the KDD model outlined in (Fayyad *et al.* 1996b).

4.1 Definitions

In the literature of knowledge engineering and database management, segmentation can be viewed as data chunking, clustering, or knowledge discovery. Each identified segmentation class is a piece of discovered knowledge, described by a certain set of attributes, rules, functions, or other forms of properties. In image processing, where segmentation is virtually the basis for all applications such as detection and classification, segmentation is viewed as a process of pixel labeling in which pixels are data instances, as mentioned above in the introduction section of this paper. Extending the definitions of image segmentation (Pavlidis 1977, Zucker 1976) to our case that deals with databases, we define segmentation as follows:

(1) *Every data instance of the database must be within a segmentation class.*
(2) *No segmentation class is isolated.*
(3) *All data instances within a segmentation class must satisfy a uniformity measure.*
(4) *Each segmentation class is maximally uniform by itself.*

The first condition requires that every data instance of the database be processed and labeled as a result of segmentation. However, in practice, databases are subjected to filtering and selection to weed out data instances not pertinent or even occluding to the segmentation process. Thus, the relaxed version of the first condition is that every data instance that qualifies for the segmentation process must eventually be labeled. The second condition follows from the first. If there exists an isolated segmentation class (with no neighboring classes), then that implies that either there is only one segmentation class in the database or the segmentation class is surrounded by unlabeled data instances. In the latter case, that signals a violation of the first condition, and that means the segmentation is incomplete. In the former case, if the objective of the segmentation is to discover knowledge regarding the composition of the data instances from a single database, then grouping the whole database as one single class does not amount to any beneficial discovery. But, if the objective of the segmentation is to examine a bank of databases, that information might be helpful in telling us how the databases differ. The third condition requires each segmentation be consistent and describable by a set of certain properties. In database segmentation, this measure can be represented in rules, conjunctions and disjunctions, predicate logic, attribute value sets, and functions. The fourth condition follows from the third. It emphasizes the requirement of having distinctive segmentation classes. For example, if one set of rules fully describes a segmentation class, then that set of rules must not describe another class, and other data instances may not fall under the coverage of that set of rules. However, in some designs, fuzzy representation, probability, and non-deterministic decision are allowed to assign a data instance into two or more segmentation classes. So, the relaxed version of the fourth condition should be that each segmentation class is probabilistically or fuzzily maximal by itself. This accommodates for the possibility of other data instances corrupting the class by chance, and of its own data instances migrating from the class by chance as well.

The above definitions of segmentation in databases *differ* from those in data mining and KDD. In the latter research areas, the objective is to mine the data and extract knowledge; and hence it does not require that every single selected data instance be labeled. In segmentation, every selected data instance is required to be explained and represented. This imposes a more rigorous constraint on the knowledge to be discovered. There are several advantages of adhering to this definition when we perform KDD or data mining, compared to conventional guidelines and systems. First, the resulting knowledge is more general in the sense that since it has to cover *all* data instances in the selected portion of the database. This translates to complete data warehousing and cataloging. In cases where each data instance must be indexed and stored, our segmentation definitions thus provide a set of design goals for constructing a data warehouse. Second, since all data instances must be accounted for as required by the definitions, a system abide by the definitions is efficient in terms of data usage. Third, guided by the conditions, data miners do not have as much freedom in excluding data instances from analysis based on their intuitions and experiences as they do when working from the conventional KDD or data mining viewpoints. This implies less influence of biases and inconsistencies from data miners during the mining or the design process. This restrain potentially allows more objective observations to be mined from the database. However, by considering the whole database, we face the risk of responding to noise or redundant data instances. In practice, data filtering or selection is used to clean or target a desired subset of the database, as mentioned above. Conversely, the above definitions pose several weaknesses. First, it is harder to achieve a complete segmentation since it requires the system or data miner to completely describe the database. This might result in over-generalization and over-fitting for some data instances, and the discovered knowledge might not be as insightful as obtained from a system focused only on a specific subset of the database. Second, it is more computational intensive due to its coverage of the whole database. In the following, we show areas where segmentation could be used to improve the KDD results.

4.2 Segmentation in KDD

Now, we have defined our viewpoint of segmentation, we turn our attention to its roles in data mining and KDD. The increase in the quality and speed of computer-aided tools, the reduction in cost of data storage and processing, customer customization, the computerization of masses, and the realization that databases contain valuable (but obscured) information have called for data mining in commercial and governmental activities. Moreover, the increase in the amount and speed of data collected and distributed, the establishment of the information superhighway in terms of communications and connectivity, the availability of computer software and resources have similarly pushed for research and applications in data mining in scientific areas. Segmentation, as defined above, has many different roles when examined in the framework of KDD, from pre-processing tasks such as indexing and management to final analyses such as knowledge discovery and interpretation. In the following, we outline several areas where segmentation can be used to improve the products of KDD.

4.2.1 Data Access

First, there is a need to *access* data. Without access to data, the intelligence and utility of many operational Artificial Intelligence (AI) or knowledge-based systems are compromised. To improve accessibility, we usually encounter two issues: (1) data starvation, and (2) data indexing. The first issue suggests the number of data instances to be stored in the database, as to not to incur starvation on the knowledge base. In segmentation, we deal with all data instances

and thus no starvation. In addition, segmentation provides a basis for indexing data. It has been used to catalog images such as the SKICAT system (Fayyad *et al.* 1996a) and the content-based cataloging at IBM. It can also be used to generate metadata for cataloging database repository such as business data warehouses, digital image libraries, and multimedia document libraries such as websites and webpages.

4.2.2 Data Abstraction

Second, there is a need to *abstract* data. In general, data abstraction reduces noise effects and allows the end users to view the database at a higher level. In corporate companies or research laboratories, a re-organization or summary of data for end users such as colleagues or computer-aided tools is usually needed to report and summarize findings and progress. Business intelligence applications such as decision support systems (DSS) and executive information systems (EIS) benefit from segmentation because of its descriptive power and its completeness—segmentation attaches a vector of properties to each class, and each data instance is labeled under a class. Not only is data abstraction desired for clarity and interpretability, it is also desired for logistic reasons. Faced with the avalanche of data generated everyday, segmentation can be used to abstract data to improve efficiency in data delivery, transmission, and storage. Finally, data abstraction has been used in forecasting, prediction, and understanding in data analysis, and matching, presentation, and classification in image processing.

4.2.3 Data Management

Third, there is a need to *manage* data, particularly data warehouses and data marts. Data warehousing is the means for managing a data interface and storage for user access, delivery, maintenance, and analysis. Corporate data warehouses may sometimes become an amalgamation of various departments and sources such that customized or in-depth mining is hindered. In view of this, there have been efforts to compartmentalize data warehouses into data marts. Due to its smaller and more restricted coverage of databases, data marts provide flexibility in customization, operation, and data mining, in addition to requiring a lower maintenance cost. Segmentation is a process naturally suited for such compartmentalization. It allows hierarchical refinement of a class at different granularity levels.

4.2.4 Data Analysis

Fourth, there is a need to *analyze* data. Data needs to be analyzed in order to be useful. Databases are multidimensional, and multidimensional relationships among hundreds of attributes can be difficult to detect and identify without segmentation. In databases, we define data analysis as the task of investigating data under the guidance of human experts. After segmentation, each class will be associated with a vector of properties. Experts interpret these vectors to draw conclusions and record observations.

4.2.5 Data Exploration

Finally, there is a need to *explore* data. Data exploration differs from data analysis. The latter is sometimes guided by human expertise within the domain of the data. The former, however, is mostly guided by human expertise in the data mining process with no or minute amount of

knowledge about the data domain. For example, a database consists of protein segments analyzed by a bio-chemist is usually influenced by his or her familiarity with the data; whereas upon exploration by a knowledge engineer or data miner, the result is usually a product guided by the person's experience in deciding the methods of filtering, transformation, clustering strategies, etc. Therefore, while data analysis provides us with practical and sensible knowledge, data exploration gives us an alternative or fresh look at the data and previously unobserved knowledge. Again, segmentation can play a major role in data exploration.

Recent advancement in building digital libraries [Wilensky 1996, Smith 1996], in which data is stored and managed such that indexing and retrieval are made convenient, has increased the importance of metadata extraction. Metadata is a set of high-level data that describes another set of low-level data. In one scenario, the low level representation of a database is its data instances. One can compute, for example, the averages for all its attribute values and use them to describe that database. The averages—the metadata—help us label each database and distinguish among significant groups of databases. In another scenario, metadata can be extracted to describe groups (identified through segmentation) within a single database to provide a finer resolution of description. In short, metadata can be used to describe different databases within a repository or different groups within a database. That segmentation extracts metadata indicates that it is an important for building digital libraries.

4.3 Importance of Unsupervised Segmentation in KDD

So far, we have defined what segmentation is and why it is important in KDD. Here, we proceed to discuss the motivations behind unsupervised segmentation. Unsupervised segmentation has become increasingly important in today's world: data has been generated in remote sensing such as satellite and aerial images, geophysical monitoring such as volcanic activities and seismographic signals, commercial databases such as bank accounts and financial transactions, telecommunications such as traffic behavior, documents such as pages on the World Wide Web and internet and many other areas, ranging from imagery to numerical to symbolic formats in an unprecedented amount and speed. To analyze, catalog, and access the data, fully or partially automated analyses are of great necessity as either stand-alone programs or assistants to humans. As knowledge engineering and data mining being popularly applied to discover knowledge from large databases, to prevent human pre-judgment and biases from influencing the process, unsupervised segmentation is called for. For example, in some business applications, segmentation is used to divide the potential customer base into demographic groups. One of the most common approaches is by the usage of census data: all addresses in a certain zip code area whose head of the household has an average age between 20 and 35 and middle income might be singled out to send camping and hiking advertisements and magazines. As a result, most segmentation that is done is *biased* segmentation. In addition, the bias is a combination of the data miner's experience (which varies from person to person) and judgment (which may change from month to month for the same person), and thus it is not always justifiable and consistent. In this example, the data miner believes segmenting the database according to age, income, and zip core is the correct approach to obtain meaningful groups which may not be true. On the contrary, unsupervised segmentation is not faced with these problems. Its self-organizing capability is able to keep human influence (especially those of the data domain) at a minimum while partitioning the database into significant classes based on domain-neutral methodologies. In addition, in cases where the classification of the database is not known *a priori*, to use supervised segmentation such as induction, one is required to assign classes to the data. Not only can this action introduce human bias into the process, but also may induce irrelevant information from

the data. Unsupervised segmentation can be utilized as a complementary tool to such process to provide verification or even as an exploratory tool alone to discover natural classes without *a priori* knowledge about the domain. Finally, in the advent of technological modernization, computer-aided tools are needed to efficiently and effectively manage the data mining process. Efficiency is necessary because of the resources invested in such research can be costly and demanding. Effectiveness is required when the amount of data is vast. Unsupervised segmentation can provide parts of the solution for speed, automation, and consistency. To illustrate, when building a digital library, when the data streams in continuously such as video-feed, audio-feed, or textual news, to avoid data congestion and delay in access, unsupervised segmentation is important since it can be implemented as a tool or an agent to automatically segment the data streams and correctly index them into proper storage space real-time.

One requirement of segmentation is the determination of the number of classes in the data. Subsequently, segmentation is able to identify the composition of each of those classes. The ability of determining the number of classes automatically is the key characteristic of unsupervised segmentation. In our research, the meaning of *unsupervised* segmentation is two-fold. If we treat the segmentation as a process that involves iterations of selection, decision making, and feedbacks in order to obtain the desired or useful results from the database, the *unsupervised* part refers to the classification or clustering module we use—the technique that determines the number of classes. If that module does not require human intervention or pre-determined number of classes, then it is unsupervised. The second viewpoint evolves around the end-product of the design process. As mentioned above, our general model can be used to prescribe the design of a segmentation technique. If this technique is fully automated, then it is unsupervised, regardless of what the classification or clustering module is. For example, if we pre-determine the number of classes into three and if the technique segments the database accordingly without human intervention, then the technique is an unsupervised segmentation technique. However, as we shall see later in Section II, we require that a technique be general and applicable to a variety of scenarios within the same domain; this diminishes the practical possibility of using pre-determined parameters regarding the number of classes in an unsupervised setting. From the standpoint of the relationship between computer tools and humans, the first viewpoint is the interaction between human expertise (which may not be domain-biased) and various decision making stages of the process to extract information; the second viewpoint of unsupervised segmentation is the process of automated extraction of information that is subsequently reviewed by human experts.

5 Conclusion

Note that many nontrivial implementation details are not addressed. Most of these approaches require applying the general model to a specific domain and application in order to derive the correct content properties and determine the implementation parameters. The task of unsupervised segmentation is of great concerns not only in image processing, but also in data processing. For image processing, unsupervised segmentation is the basis towards automation of image analysis since segmentation precedes many tasks such as classification, recognition, understanding, and registration. Our Data Investigation Model for Unsupervised Segmentation (DIMUS) has been conceived with the concept that it be general and applicable to different implementation approaches, as we have illustrated in this paper.

Through a discussion based on the relationship between unsupervised segmentation and KDD, we have also hinted that our model, DIMUS, can be applied to databases other than

images. Indeed, our model addresses issues well-known in the KDD community. We have also argued for the importance of unsupervised segmentation in KDD which might involve digital image library, or imagery repositories. For those cases, DIMUS can serve as a guideline to develop an automated system or tool to aid humans in data management and analysis.

In summary, we have proposed a general model for unsupervised image segmentation which determines the number of classes and identifies the composition of each of the classes automatically. This model, called the Data Investigation Model for Unsupervised Segmentation (DIMUS), consists of five modules: inspector, clues generator, determiner, justifier, and mapper. We have discussed in details the functions of each module and possible solutions for its design. In addition, we have applied the model to several image processing approaches such as knowledge-based systems, neural networks, estimation via optimization, regularization theory, mathematical morphology, relaxation, split and merge, multiresolution and pyramid, fuzzy sets, region growing, edge detection, thresholding, and clustering. The discussion offers possible solutions of the model in different requirements of implementation choices, showing the model's generality and flexibility.

6 References

Abutaleb, A. S. (1989). Automatic Thresholding of Gray-Level Pictures Using Two-Dimensional Entropy, *Computer Vision, Graphics, and Image Processing*, **47**(1), 22–32.

Acton, S. T. (1996). On Unsupervised Segmentation of Remotely Sensed Imagery Using Nonlinear Regression, *International Journal of Remote Sensing*, **17**(7), 1407–1415.

Ball, G. H. and D. J. Hall (1967). A Clustering Technique for Summarizing Multivariate Data, *Behavioral Science*, **12**, 153–155.

Ballard, D. H. (1981). Generalizing the Hough Transform to Detect Arbitrary Shapes, *Pattern Recognition*, **13**(2), 111–122.

Baraldi, A. and F. Parmiggiani (1996). Single Linkage Region Growing Algorithms Based on the Vector Degree of Match, *IEEE Transactions on Geoscience and Remote Sensing*, **34**(1), 137–148.

Beaulieu, J.-M. and M. Goldberg (1989). Hierarchy in Picture Segmentation: A Stepwise Optimization Approach, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **11**(2), 150–163.

Bergholm, F. (1981). Edge Focusing, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **9**(6), 726–741.

Binford, T. O. (1982). Survey of Model-Based Images Analysis Systems, *International Journal of Robotics Research*, **1**(1), 18–64.

Bouman, C. and B. Liu (1991). Multiple Resolution Segmentation of Textured Images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **13**, 99–113.

Brachman, R. J. (1985). I Lied about the Trees, *AI Magazine*, **6**(3), 80–93.

Brice, C. R. and C. L. Fennema (1970). Scene Analysis Using Regions, *Artificial Intelligence Journal*, **1**(3), 205–226.

Caillol, H., A. Hillion, and W. Pieczynski (1993). Fuzzy Random Fields and Unsupervised Image Segmentation, *IEEE Transactions on Geoscience and Remote Sensing*, **31**(4), 801–810.

Canny, J. F. (1986). A Computational Approach to Edge Detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **8**(6), 679–698.

Carbonell, J. (ed.) (1990). *Machine Learning, Paradigms and Methods*, Cambridge, MA: MIT/Elsevier Press.

Carpenter, G. A. and S. A. Grossberg (1987). The ART of Adaptive Pattern Recognition by Self-Organizing Neural Network, *IEEE Computer*, **21**, 77–88.

Cheeseman, P., J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman (1990). AutoClass: A Bayesian Classification System, in *Readings in Machine Learning*, J. W. Shavlik and T. G. Dietterich (eds.), San Mateo, CA: Morgan Kaufmann, 296–306.

Chen, P. C. and T. Pavlidis (1979). Segmentation by Texture using a Co-occurrence Matrix and a Split-and-Merge Algorithm, *Computer Graphics and Image Processing*, **10**(2), 172–182.

Cohen, F. S. and Z. G. Fan (1992). Maximum Likelihood Unsupervised Textured Image Segmentation, *CVGIP: Graphical Models and Image Processing*, **54**(3), 239–251.

Cook, D. J. and L. B. Holder (1994). Substructure Discovery Using Minimum Description Length and Background Knowledge, *Journal of Artificial Intelligence Research*, **1**, 231–255.

D'Astous, F. and M. E. Jernigan (1984). Texture Discrimination based on Detailed Measures of the Power Spectrum, in *Proceedings of 7th International Conference on Pattern Recognition*, Montreal, July 20–August 2, 83–86.

Derin, H. and W. S. Cole (1986). Segmentation of Textured Images Using Gibbs Random Fields, *Computer Vision, Graphics, and Image Processing*, **35**(1), 72–98.

Derin, H., H. Elliott, R. Cristi, and D. Geman (1984). Bayes Smoothing Algorithms for Segmentation of Binary Images Modeled by Markov Random Fields, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **6**, 697–720.

Durand, J. M., B. J. Gimonet, and J. R. Perbos (1987). SAR Data Filtering for Classification, *IEEE Transactions on Geoscience and Remote Sensing*, **25**(5), 629–637.

Fayyad, U. M., S. G. Djorgovski, and N. Weir (1996a). From Digitized Images to On-Line Catalogs: Data Mining a Sky Survey, *AI Magazine*, **17**(2), pp. 51-66.

Fayyad, U. M., G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (1996b). *Advances in Knowledge Discovery and Data Mining*, Menlo park, CA: AAAI/MIT Press.

Feldman, J. A., and Y. Yakimovsky (1974). Decision Theory and Artificial Intelligence, I. A Semantics Based Region Analyzer, *Artificial Intelligence*, **5**, 349–371.

Fisher, D. H. (1987). Knowledge Acquisition via Incremental Conceptual Clustering, *Machine Learning*, **2**, 139–172.

Freeman, H. (1974). Computer Processing of Line Drawing Images, *Computer Surveys*, **6**(1), 57–98.

Fu, K. S. and J. K. Mui (1981). A Survey on Image Segmentation, *Pattern Recognition*, **13**, 3–16.

Gauch, J. M. (1992). Investigations of Image Contrast Space Defined by Variations on Histogram Equalization, *CVGIP: Graphical Models and Image Processing*, **54**(4), 269–280.

Gauch, J. M. and S. M. Pizer (1993). The Intensity Axis Symmetry and Its Application to Image Segmentation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **15**(8), 753–770.

Geman, S. and D. Geman (1984). Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **6**(6), 721-741.

Gennari, J. H., P. Langley, and D. Fisher (1990). Models of Incremental Concept Formation, in *Machine Learning: Paradigms and Methods*, J. Carbonell (ed.), 11–61, MIT Press/Elsevier.

Goldberg, M. and S. Shlien (1978). A Cluster Scheme for Multispectral Images, *IEEE Transactions on Systems, Man, and Cybernetics*, **8**, 86–92.

Hall, L. O., A. M. Bensaid, L. P. Clarke, R. P. Velthuizen, M. Silbiger, and J. C. Bezdek (1992). A Comparison of Neural Network and Fuzzy Clustering Techniques in Segmenting Magnetic Resonance Images of the Brain, *IEEE Transactions on Neural Networks*, **3**(5), 672–681.

Haralick, R. M. and G. L. Kelly (1969). Pattern Recognition with Measurement Space and Spatial Clustering for Multiple Image, *Proceedings of IEEE*, **57**, 654–665.

Haralick, R. M., K. Shanmugan, and I. Dinstein (1973). Textural Features for Image Classification, *IEEE Transactions on Systems, Man, and Cybernetics*, **3**(6), 610–621.

Haralick, R. M. and L. G. Shapiro (1985). Survey—Image Segmentation Techniques, *Computer Vision, Graphics, and Image Processing*, **29**, 100–132.

Hartigan, J. A. (1975). *Clustering Algorithms*, New York: John Wiley & Sons.

Hartigan, J. A. and M. A. Wong (1979). A K-means Clustering Algorithm: Algorithm AS 136, *Applied Statistics*, **28**, 126–130.

Hassner, M. and J. Sklansky (1980). The Use of Markov Random Fields as Models of Texture, *Computer Graphics and Image Processing*, **12**(4), 357–370.

Haverkamp, D., C. Tsatsoulis, and S. Gogineni (1994). The Combination of Algorithmic and Heuristics Methods for the Classification of Sea Ice Imagery, *Remote Sensing Journal*, **9**(2), 135–159.

Haverkamp, D., L.-K. Soh, and C. Tsatsoulis (1995). A Comprehensive, Automated Approach to Determining Sea Ice Thickness from SAR Data, *IEEE Transactions on Geoscience and Remote Sensing*, **33**(1), 46–57

Holt, B., R. Kwok, and E. Rignot (1989). Ice Classification Algorithm Development and Verification for the Alaska SAR Facility Using Aircraft Imagery, *International Geoscience and Remote Sensing Symposium*, **2**, 751–754.

Hong, T. H. and M. Shneier (1984). Extracting Compact Objects Using Linked Pyramids, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **6**(20), 229-236.

Horowitz, S. L. and T. Pavlidis (1976). Picture Segmentation by a Tree Traversal, *Journal of the Association for Computing Machinery*, **23**(2), 368–388.

Hough, P. V. C. (1962). Methods and Means for Recognizing Complex Patterns, U.S. Patent 3069654.

Hummel, R. (1977). Image Enhancement by Histogram Transformation, *Computer Graphics and Image Processing*, **6**, 184–195.

Hummel, R. A. and S. W. Zucker (1983). On the Foundations of Relaxation Labeling Processes, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **5**(3), 267–287.

Jarvis, R. A. and E. A. Patrick (1973). Clustering Using a Similarity Measure Based on Shared Near Neighbors, *IEEE Transactions on Computers*, **22**, 1025–1034.

Jeong, H. and C. I. Kim (1992). Adaptive Determination of Filter Scales for Edge Detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **14**(5), 579–585.

Kirpatrick, S., C. D. Gelatt, Jr., and M. P. Vecchi (1983). Optimization by Simulated Annealing, *Science*, **220**(4598), 671–680.

Kittler, J. and J. Illingworth (1985). Relaxation Labelling Algorithms — a Review, *Image and Vision Computing*, **3**(4), 206–216.

Kohonen, T. (1989). *Self-Organization and Associative Memory*, Berlin: Springer-Verlag.

Kolodner, J. L. (1983). Reconstructive Memory: A Computer Model, *Cognitive Science*, **7**, 281–328.

Le Hegarat-Mascle, S., D. Vidal-Madjar, and P. Olivier (1996). Applications of Simulated Annealing to SAR Image Clustering and Classification Problems, *International Journal of Remote Sensing*, **17**(9), 1761–1776.

Le Moigne, J. and J. C. Tilton (1995). Refining Image Segmentation by Integration of Edge and Region Data, *IEEE Transactions on Geoscience and Remote Sensing*, **33**(3), 605–615.

Lebowitz, M. (1987). Experiments with Incremental Concept Formation: UNIMEM, *Machine Learning*, **2**, 103–138.

Leclerc, Y. G. (1989). Constructing Simple Stable Descriptions for Image Partitioning, *International Journal of Computer Vision*, **3**(1), 73–102.

Lee, J.-S. (1983). Digital Image Smoothing and the Sigma Filter, *Computer Vision, Graphics, and Image Processing*, **24**(2), 255–269.

Lee, J.-S. and I. Jurkervich (1989). Segmentation of SAR Images, *IEEE Transactions on Geoscience and Remote Sensing*, **27**, 674–680.

Lim, Y. W. and S. U. Lee (1990). On the Color Image Segmentation Algorithm Based on the Thresholding and the Fuzzy C-Means Techniques, *Pattern Recognition*, **23**(9), 935–952.

Mahalanobis, P. C. (1936). On the Generalized Distance in Statistics, *Proceedings of the National Institute of Science of India*, **2**(49), 49–55.

Marroquin, J., S. Mitter, and T. Poggio (1987). Probabilistic Solution of Ill-Posed Problems in Computational Vision, *Journal of the American Statistical Association*, **62**(397), 76–89.

Mastin, G. A. (1985). Adaptive Filters for Digital Image Noise Smoothing: An Evaluation, *Computer Vision, Graphics, and Image Processing*, **31**(1), 103–121.

Matheron, G. (1975). *Random Sets and Integral Geometry*, New York: Wiley.

Matsuyama, T. and V. Hwang (1990). SIGMA, a Knowledge-Based Aerial Image Understanding System, in *Advances in Computer Vision and Machine Intelligence*, New York: Plenum.

McCarthy, J. and P. J. Hayes (1969). Some Philosophical Problems from the Standpoint of Artificial Intelligence, in *Machine Learning 4*, B. Meltzer and D. Michie (eds.), Edinburgh: Edinburgh University Press.

Meer, P. (1989). Stochastic Image Pyramids, *Computer Vision, Graphics, and Image Processing*, **45**, 269–294.

Murthy, C. A. and S. K. Pal (1990). Fuzzy Thresholding: Mathematical Framework, Bound Functions and Weight Moving Average Technique, *Pattern Recognition Letters*, **11**, 197–206.

Muzzolini, R., Y.-H. Yang, and R. Pierson (1993). Multiresolution Texture Segmentation with Application to Diagnostic Ultrasound Images, *IEEE Transactions on Medical Imaging*, **12**(1), 108–123.

Nguyen, H. H. and P. Cohen (1993). Gibbs Random Fields, Fuzzy Clustering, and the Unsupervised Segmentation of Textured Images, *CVGIP: Graphical Models and Image Processing*, **55**(1), 1–19.

O'Gorman, L. (1994). Binarization and Multithresholding of Document Images Using Connectivity, *CVGIP: Graphical Models and Image Processing*, **56**(6), 494–506.

Pal, N. R. and S. K. Pal (1993). A Review on Image Segmentation Techniques, *Pattern Recognition*, **26**(9), 1277–1294.

Pal, S. K., R. A. King, and A. A. Hashim (1983). Automatic Gray Level Thresholding Through Index of Fuzziness and Entropy, *Pattern Recognition Letters*, **1**, 141–146.

Panda, D. P. and A. Rosenfeld (1978). Image Segmentation by Pixel Classification in (Gray Level, Edge Value) Space, *IEEE Transactions on Computer*, **27**, 875–879.

Panjwami, D. K. and G. Healey (1995). Markov Random Field Models for Unsupervised Segmentation of Textured Color Images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **17**(10), 939–954.

Paranjape, R. B., W. M. Morrow, and R. M. Rangayyan (1992). Adaptive-Neighborhood Histogram Equalization for Image Enhancement, *CVGIP: Graphical Models and Image Processing*, **54**(3), 259–267.

Pavlidis, T. (1977). *Structural Pattern Recognition*, New York: Springer.

Pavlidis, T. and Y.-T. Liow (1990). Integrating Region Growing and Edge Detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12**(3), 225–233.

Pedrycz, W. (1990). Fuzzy Sets in Pattern Recognition: Methodology and Methods, *Pattern Recognition*, **23**, 121–146.

Peleg, S. and A. Rosenfeld (1978). Determining Compatibility Coefficients for Curve Enhancement Relaxation Process, *IEEE Transactions on Systems, Man, and Cybernetics*, **8**(7), 548–555.

Pelillo, M. and M. Refice (1994). Learning Compatibility Coefficients for Relaxation Labeling Processes, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **16**(9), 933–945.

Peng, A. and W. Pieczynski (1995). Adaptive Mixture Estimation and Unsupervised local Bayesian Image Segmentation, *CVGIP: Graphical Models and Image Processing*, **57**(5), 389–399.

Poggio, T., V. Torre, and C. Koch (1985). Computational Vision and Regularization Theory, *Science*, **317**(6035), 314–319.

Postaire, J-G., R. D. Zhang, and C. Lecocq-Botte (1993). Cluster Analysis by Binary Morphology, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **15**(2), 170–180.

Raney, R. K. (1985). Theory and Measure of Certain Image Norms in SAR, *IEEE Transactions on Geoscience and Remote Sensing*, **23**(3), 343–348.

Rissanen, J. (1978). Modeling by Shortest Data Description, *Automatica*, **14**, 465–471.

Rissanen, J. (1989). *Stochastic Complexity in Statistical Inquiry*, Singapore: World Scientific Publishing Company.

Rosenfeld, A. (ed.) (1984). *Multiresolution Image Processing and Analysis*, Berlin: Springer-Verlag.

Rosenfeld, A. and L. S. Davis (1979). Image Segmentation and Image Models, *Proceedings of IEEE*, **67**, 784–772.

Rosenfeld, A. and A. C. Kak (1982). *Digital Image Processing, 2nd Ed.*, New York: Academic Press.

Serra, J. (1982). *Image Analysis and Mathematical Morphology*, New York: Academic Press.

Serra, J. (1986). Introduction to Mathematical Morphology, *Computer Vision, Graphics, and Image Processing*, **35**, 283–305.

Shavlik, J. W. and T. G. Dietterich (eds.) (1990). *Readings in Machine Learning*, San Mateo, CA: Morgan Kaufmann.

Sjoberg, F. and F. Bergholm (1988). Extraction of Diffuse Edge Focusing, *Pattern Recognition Letters*, **7**(3), 181–190.

Sklansky, J., R. L. Chazin, and B. J. Hansen (1972). Minimum-Perimeter Polygons of Digitized Silhouettes, *IEEE Transactions on Computers*, **21**(3), 260–268.

Smith, D. M. (1996). Speckle Reduction and Segmentation of Synthetic Aperture Radar Images, *International Journal of Remote Sensing*, **17**(11), 2043–2057.

Soh, L.-K. (1998). *Unsupervised Image Segmentation: A Data Investigation Model and SAR Sea Ice Applications*, Ph.D. Dissertation, Department of Electrical Engineering and Computer Science, University of Kansas, Lawrence, KS.

Soh, L.-K. and C. Tsatsoulis (to appear). Segmentation of Satellite Imagery of Natural Scenes Using Data Mining, accepted for publication by *IEEE Transactions on Geoscience and Remote Sensing*.

Tanimoto, S. and T. Pavlidis (1975). A Hierarchical Data Structure for Picture Processing, *Computer Graphics and Image Processing*, **4**(2), 104–119.

Thompson, K. and P. Langley (1991). 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, 127–161.

Thompson, K. and K. McKusick (1993). COBWEB/3: A Portable Implementation, *Technical Report FIA-96-6-18-2*, version 1.4, Ames Research Center.

Ton, J., J. Sticklen, and A. K. Jain (1991). Knowledge-Based Segmentation of Landsat Images, *IEEE Transactions on Geoscience and Remote Sensing*, **29**(2), 222–232.

Torre, V. and T. Poggio (1986). On Edge Detection, *IEEE Transactions on Pattern Analysis and Pattern Intelligence*, **8**, 147–163.

Trivedi, M. and J. C. Bezdek (1986). Low-Level Segmentation of Aerial Images with Fuzzy Clustering, *IEEE Transactions on Systems, Man, and Cybernetics*, **16**(4), 589–598.

Uchiyama, T. and M. A. Arbib (1994). Color Image Segmentation Using Competitive Learning, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **16**(12), 1197–1206.

Wallace, C. S. and D. L. Dow (1994). Intrinsic Classification by MML—The Snob Program, in *Proceedings of the Seventh Australian Joint Conference on Artificial Intelligence*, Armidale, New South Wales, Australia, November 1994, 37–44.

Wang, F.-J. (1993). A Knowledge-Based Vision System for Detecting Land Changes at Urban Fringes, *IEEE Transactions on Geoscience and Remote Sensing*, **31**(1), 136–145.

Wilkinson, G. G. and J. Megier (1990). Evidential Reasoning in a Pixel Classification Hierarchy—A Potential Method for Integrating Image Classifiers and Expert System Rules Based on Geographic Context, *International Journal of Remote Sensing*, **11**, 1963–1968.

Won, C. S. and H. Derin (1992). Unsupervised Segmentation of Noisy and Texture Using Markov Random Fields, *CVGIP: Graphical Models and Image Processing*, **54**(4), 308–328.

Wong, Y.-F. and E. C. Posner (1993). A New Clustering Algorithm Applicable to Multispectral and Polarimetric SAR Images, *IEEE Transactions on Geoscience and Remote Sensing*, **31**(3), 634–644.

Woods, W. A. (1975). What's in a Link: Foundations for Semantic Networks, in *Representation and Understanding*, D. G. Bobrow and A. Collins (eds.), New York: Academic Press.

Wu, Z.-Y. and R. Leahy (1993). An Optimal Graph Theoretic Approach to Data Clustering: Theory and Its Application to Image Segmentation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **15**(11), 1101–1113.

Yu, S. and M. Berthod (1995). A Game Strategy Approach for Image Labeling, *CVGIP: Image Understanding*, **61**(1), 32–37.

Zadeh, L. A. (1965). Fuzzy Sets, *Information and Control*, **8**, 338–353.

Zucker, S. W. (1976). Region Growing: Childhood and Adolescence, *Computer Graphics and Image Processing*, **5**, 382–399.