Adaptive Integrated Image Segmentation and Object Recognition

Bir Bhanu, *Fellow, IEEE*, and Jing Peng

***Abstract*—This paper presents a general approach to image segmentation and object recognition that can adapt the image segmentation algorithm parameters to the changing environmental conditions. Segmentation parameters are represented by a team of generalized stochastic learning automata and learned using connectionist reinforcement learning techniques. The edge-border coincidence measure is first used as reinforcement for segmentation evaluation to reduce computational expenses associated with model matching during the early stage of adaptation. This measure alone, however, can not reliably predict the outcome of object recognition. Therefore, it is used in conjunction with model matching where the matching confidence is used as a reinforcement signal to provide optimal segmentation evaluation in a closed-loop object recognition system. The adaptation alternates between global and local segmentation processes in order to achieve optimal recognition performance. Results are presented for both indoor and outdoor color images where the performance improvement over time is shown for both image segmentation and object recognition.**

***Index Terms*—Adaptive image segmentation, adaptive object recognition, closed-loop recognition, closed-loop segmentation, model-based recognition, learning for object recognition.**

I. Introduction

A MODEL-BASED object recognition system has three key components: image segmentation, feature extraction, and model matching. The goal of image segmentation is to extract meaningful objects from an input image. Image segmentation is an important and one of the most difficult low-level image analysis tasks [5], [10]. All subsequent tasks including feature extraction and model matching rely heavily on the quality of the image segmentation process.

The inability to adapt the image segmentation process to real-world changes is one of the fundamental weaknesses of typical model-based object recognition systems. The real-world changes are caused by variations in environmental conditions, imaging devices, lighting conditions, time of the day, sun position, shadows, weather conditions, etc. Despite the large number of image segmentation algorithms available [6], [11], [23], no general methods have been found to process the wide diversity of images encountered in real world applications. Usually, an object recognition system is *open-loop*. Segmentation and feature extraction modules use default values of algorithm parameters, and generally work as pre-processing steps to the model matching component. These default values of algorithm parameters, however, generally degrade the system performance and cannot adapt to changes in real-world applications. The default settings of algorithm parameters are usually obtained by the system designer by following a trial and error method. Parameters obtained in this way are not robust, since when the conditions for which they are designed are changed slightly, these algorithms generally fail without any graceful degradation in performance.

The usefulness of a set of algorithm parameters in an image analysis system can only be determined by the system's output, for example, recognition performance. To recognize different objects or instances of the same object in an image, we may need different sets of parameters locally due to the changes in local image properties, such as brightness, contrast, etc. Also the changing environmental conditions affect the appearance of an image that requires the ability to adapt the algorithm parameters for multi-scenario object recognition. To achieve robust performance in real-world applications, therefore, a need exists to apply learning techniques that can efficiently find parameter values yielding optimal results for the given recognition task. In this paper, our goal is to develop a general approach to model-based object recognition that has the ability to continuously adapt to normal environmental variations.

The original contributions of the adaptive integrated image segmentation and object recognition system presented in this paper are

1) model matching confidence is used as feedback to influence the image segmentation process, thereby providing our object recognition system with adaptability in real world scenarios;
2) a team of generalized stochastic learning automata is used to represent both global and local image segmentation parameters, making faster learning possible;
3) edge-border coincidence, when combined with model matching confidence, reduces overall computational costs of the learning process;
4) explicit bias is introduced in a reinforcement learning system in order to speed up the learning process for adaptive image segmentation.

In the remainder of Section I, we present an overview of the approach and related research. Section II gives the details of the approach and discusses algorithms used in this research. Section III provides the experimental results for segmentation and

Manuscript received April 12, 1999; revised September 30, 2000. This work was supported in part by DARPA/AFOSR Grant F49620-97-1-0184 at the University of California at Riverside. The contents of the information do not necessarily reflect the position or the policy of the U.S. Government.

B. Bhanu is with the Center for Research in Intelligent Systems, University of California, Riverside, CA 92521-0425 USA (e-mail: bhanu@cris.ucr.edu).

J. Peng is with the Oklahoma State University, Stillwater, OK 74078 (e-mail: jpeng@cs.okstate.edu).

Publisher Item Identifier S 1094-6977(00)11204-0.

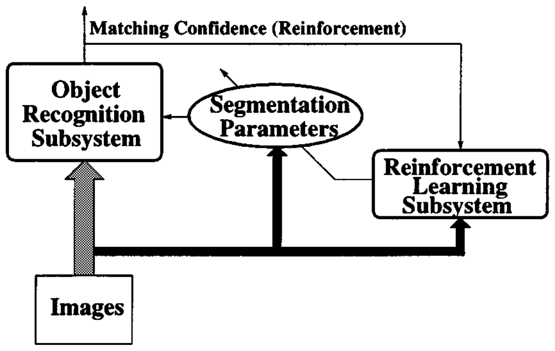
Matching Confidence (Reinforcement)
Object Recognition Subsystem
Segmentation Parameters
Reinforcement Learning Subsystem
Images

Fig. 1. Adaptive integrated image segmentation and object recognition system.

recognition on both indoor and outdoor color images. Finally, Section IV presents the conclusions of the paper and future directions.

A. Overview of the Approach

We develop a general approach to adaptive integrated image segmentation and object recognition. The basic *assumption* is that we know the models of the objects that are to be recognized, but we do not know the number of objects and their locations in the image. Fig. 1 shows the functional structure of the adaptive system. The system consists of two key subsystems: reinforcement learning subsystem and object recognition subsystem. The object recognition subsystem consists of three basic modules: image segmentation, feature extraction and model matching. The image segmentation module extracts meaningful objects from input images. Feature extraction performs polygonal approximation of connected components. The model matching module then carries out matching between the stored models and the polygonal approximation of the connected components in order to recognize model objects. This module indirectly evaluates the performance of the image segmentation and feature extraction processes by generating a real valued matching confidence indicating the degree of success. This real valued confidence is then used as feedback to a reinforcement learning subsystem to drive adaptation for image segmentation parameters. The goal is to adaptively compute segmentation parameters that, when applied to the segmentation algorithm, maximize the matching confidence for the given recognition task.

It is important to note that since significant differences in characteristics exist between an image and its subimages, operating parameters of algorithms need to be tuned to these differences to achieve optimal performance of segmentation and model matching. For example, to recognize two objects in an image or a single object at different locations, it is often difficult, if not impossible, to extract and recognize objects with a single fixed set of parameters of a given algorithm. It is essential to localize computation to meet each individual requirement. In order to achieve this objective our system performs both global and local adaptation. The global adaptation process finds segmentation parameters that are likely to result in the recognition of model objects from the entire image while the local process computes parameters that are best suited for selected subimages. In this way, our system is highly adaptive to variations within an image.

The particular framework adopted in this paper is reinforcement learning, which closes the loop between model matching and image segmentation. There are good reasons for using reinforcement learning in our image processing and analysis system. *First*, reinforcement learning requires knowing only the goodness of the system performance rather than the details of algorithms that produce the results. It is natural to use matching confidence as reinforcement. *Second*, convergence is guaranteed for several reinforcement learning algorithms. *Third*, reinforcement learning performs efficient hill-climbing in a statistical sense without excessive demand for computational resources. Furthermore, it can generalize over unseen images. *Fourth*, reinforcement learning can systematically assign credit to different levels in a multi-level image processing and analysis system.

The adaptive integrated image segmentation and object recognition system is designed to be *fundamental* in nature and is not dependent on any specific image segmentation algorithm or type of input images. In order to represent segmentation parameters suitably in a reinforcement learning framework, the system only needs to know the segmentation parameters and their ranges. In our approach, a binary encoding scheme is used to represent the segmentation parameters. While the same task could be learned in the original parameter space, for many types of problems, including image segmentation, the binary representation is expected to learn much faster [20]. In this sense, our system is independent of a particular segmentation algorithm used. Of course, different segmentation algorithms are designed and well suited for different kinds of images in various bands of the electromagnetic spectrum. For example, the *Phoenix* algorithm used in this paper is designed for the segmentation of color images.

B. Related Work

There is no published work on reinforcement learning integrated image segmentation and object recognition using multiple feedbacks. The adaptive parameter control of segmentation algorithm and the adaptive selection and combination of different algorithms in a learning integrated system are *unsolved* problems in the field of image processing and computer vision [1]. Most threshold selection techniques in image processing and computer vision do not involve any learning to improve future performance with experience.

In [8], Burges *et al.* describe a method for coupling recognition and segmentation by the principle of heuristic over segmentation. The basic idea is that a segmentation algorithm generates a graph that summarizes a large number of segmentation hypotheses that are scored by a recognition algorithm. A globally optimal decision is then made that combines uncertainties in segmentation and recognition. Each time a new input comes in a over segmented hypothesis graph must be generated and traversed in order to classify the input. In contrast, the system presented in this paper uses a learned mapping to compute segmentation parameters for a given input to achieve optimal model matching. In addition, the learning of mapping in our system is driven completely by the matching confidence, whereas their graph generation is largely based on heuristics. In another work [15], graph generation is actually learned by minimizing global errors that take into account both segmentation hypotheses and recognition scores. In [2], a method is described for fitting segmentation parameters to maximize the likelihood of a model of an object. In comparison, our system attempts to maximize a classification (conditional) probability.

Bhanu and Lee [5] presented an image segmentation system which incorporates a genetic algorithm to adapt the segmentation process to changes in image characteristics caused by variable environmental conditions. In their approach, multiple segmentation quality measures are used as feedback. Some of these measures require ground-truth information which may not be always available. Peng and Bhanu [20] presented an approach in which a reinforcement learning system is used to close the loop between segmentation and recognition, and to induce a mapping from input images to corresponding segmentation parameters. Their approach is based on global image segmentation which is not the best way to detect objects in an image; we need the capability of performing segmentation based on local image properties (local segmentation). Another disadvantage of their method is its time complexity that makes it problematic for practical application of computer vision. Also, the technique introduced in [21] is primarily concerned with multi-stage computer vision systems. Here we are addressing issues facing adaptive image segmentation in single stage systems that use multiple performance measures, and that exploit local image properties.

For object recognition applications, the efficiency of the learning techniques is very important. How to add bias, a prior or domain knowledge in a reinforcement learning based system is an important topic of research in reinforcement learning [9], [16], [26]. For the *RATLE* system, Maclin and Shavlik [16] accept "advice" expressed in a simple programming language. This advice is compiled into "knowledge-based" connectionist Q-learning network. They show that advice-giving can speed up Q-learning when the advice is helpful (though it need not be perfectly correct). When the advice is harmful, back propagation training quickly overrides it. Dorigo and Colombetti [9] show that by using a learning technique called learning classifier system (LCS), an external trainer working within a RL framework can help a robot to achieve a goal. Thrun and Schwartz [26] have discussed methods for incorporating background knowledge into a reinforcement learning system for robot learning.

In our approach, the edge-border coincidence is used to locate an initial good point from which to begin the search through weight space for high matching confidence values. Although as a segmentation evaluation measure the edge-border coincidence is not as reliable as the matching confidence, lower edge-border coincidence values always result in poor model matching. Likewise, higher edge-border coincidence values suggest with high probability that the current set of segmentation parameters is in a close neighborhood of the optimal one. It is an inexpensive way to arrive at an initial approximation to a set of segmentation parameters that gives rise to the optimal recognition performance. The control switches between global and local segmentation processes to optimize recognition performance. To further speed-up the learning process the reinforcement learning is biased when the model matching confidence is used as the reinforcement signal (note that the reinforcement learning is unbiased *initially* when the edge-border coincidence is used as the reinforcement signal). We achieve better computational efficiency of the learning system and improved recognition rates compared to an earlier system [20].

II. Technical Approach

Our goal is to adaptively compute segmentation parameters using matching confidence as feedback to optimize its performance for a given recognition task. However, this feedback involves expensive computation of feature extraction and model matching. In order to minimize this computation edge-border coincidence is first used to obtain an initial estimate of the parameters. While the edge-border coincidence alone may not reliably predict the outcome of model matching, it is simple to compute, thereby improving overall efficiency without causing performance degradation.

The adaptive segmentation process in our system has two distinct phases: global and local. While global segmentation is performed for the entire image, local segmentation is carried out only for selected subimages. For a set of input images, the system takes inputs sequentially. This is similar to human visual learning process, in which the visual stimuli are presented temporally in a sequential manner. For the first input image, the system having no accumulated experience begins learning with a set of random weights. For each input image thereafter, the learning process starts from the set of weights that are obtained based on all the previous input images. The following are the main steps of our learning algorithm:

Initial Approximation: A network of Bernoulli units is used to represent segmentation parameters. The edge-border coincidence is used as a direct segmentation evaluation during earlier stages of adaptation to drive weight changes without going through the expensive feature extraction and model matching. Once the edge-border coincidence has exceeded a given threshold, the weight changes will be driven by the matching confidence, which requires more expensive computation of feature extraction and model matching. Fig. 2 illustrates the blocks needed for the computation of the edge-border coincidence as reinforcement and for parameter adaptation.

Global Segmentation Adaptation: Unlike initial approximation, the global segmentation adaptation process relies on model matching to provide indirect evaluation of segmentation performance. We assume that we have a prior knowledge of approximate size of objects of interest in the images. After segmentation, the connected components that pass through the size filter based on the expected size of objects of interest in the image, we perform feature extraction and model matching. The highest matching confidence is taken as reinforcement to the learning system. Fig. 3 shows the main steps needed for model matching computation and for parameter adaptation. If the highest matching confidence level is above a given switching theshold and a given maximum number of iterations has not been reached, we focus image segmentation and model matching on the connected component and switch to the local adaptation process.

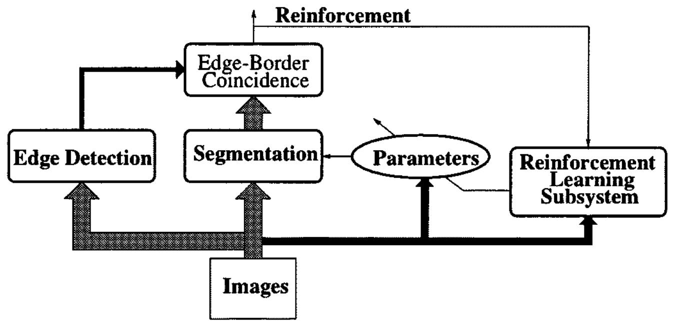

Fig. 2. Direct segmentation evaluation using edge-border coincidence.

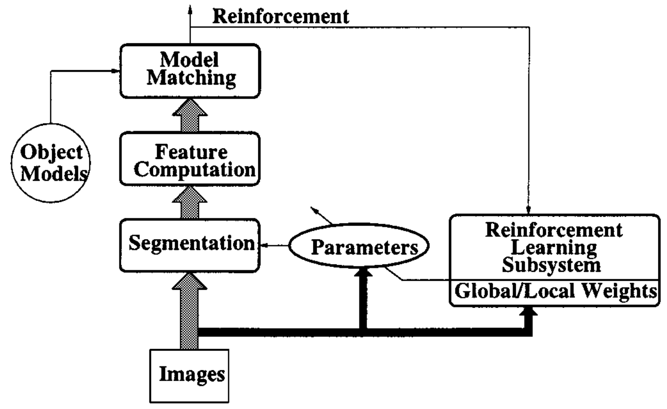

Fig. 3. Indirect segmentation evaluation.

Local Segmentation Adaptation: Once a connected component has been extracted from the input image, local adaptation begins to find the best fit segmentation parameters for the subimage. It starts from the current estimate of weights that resulted from global adaptation. Similar to global adaptation, the matching confidence is used to update local weights until the confidence exceeds an acceptable level in which case the adaptation process for the current input subimage is terminated, or a given maximum number of iterations has been reached in which case the control switches from local to global and the global adaptation process continues from the point where we switched to the local adaptation process.

A. Phoenix Image Segmentation Algorithm

Since we are working with color imagery in our experiments, we have selected the *Phoenix* segmentation algorithm [14], [19] developed at Carnegie-Mellon University and SRI International. The *Phoenix* segmentation algorithm has been widely used and tested. It works by recursively splitting regions using histogram for color features. *Phoenix* has seventeen different control parameters, fourteen of which are adjustable. The four most critical ones (see Table I) that affect the overall results of the segmentation process are selected for adaptation: *Hsmooth, Maxmin, Splitmin,* and *Height*. *Hsmooth* is the width of the histogram smoothing window. *Maxmin* is the lowest acceptable peak-to-valley height ratio. *Splitmin* represents the minimum area for a region to be automatically considered for splitting. *Height* is the minimum acceptable peak height as a percentage of the second highest peak. Each parameter has 32 possible values. The resulting search space is 2^{20} sample points. Each of the *Phoenix* parameters is represented using 5 bit binary code, with each bit represented by one Bernoulli unit. To represent four parameters, we need a total of 20 Bernoulli units. A brief description of *Phoenix* is given in Appendix A. More details about *Phoenix* are given in the report by Laws [14] and related papers [19], [24].

TABLE I
SAMPLE RANGES FOR SELECTED *PHOENIX* PARAMETERS

Parameter	Sampling Formula	Test Range
Hsmooth: hsindex $\in [0:31]$	$Hsmooth = 1 + 2 * \text{hsindex}$	1 – 63
Maxmin: mmindex $\in [0:31]$	ep = ln(100) + 0.05 $*$ mmindex; $Maxmin = \exp(\text{ep}) + 0.5$	100 – 471
Splitmin: smindex $\in [0:31]$	$Splitmin = 9 + 2 * \text{smindex}$	9 – 71
Height: htindex $\in [0:31]$	$Height = 1 + 2 * \text{htindex}$	1 – 63

B. Edge-Border Coincidence

Given that feature extraction and model matching are computationally expensive processes, it is imperative that initial approximation be made such that overall computation can be reduced. In order to achieve this objective, we introduce a direct feedback signal that measures the image segmentation quality. There is a large number of segmentation quality measures that have been proposed [5]. The segmentation evaluation metric that we have selected is the *edge-border coincidence* (EBC) [5], [18]. It measures the overlap of the region borders in the segmented image relative to the edges found using an edge detector, and does not depend on any ground-truth information about the objects to be recognized. In our approach, we use the *Sobel* edge operator [23] to compute the necessary edge information. Edge-border coincidence is defined as follows. Let E be the set of pixels (x_{pi}, y_{pi}) extracted by the edge operator after thresholding and S be the set of pixels (x_{qi}, y_{qi}) found on the region boundaries obtained from the segmentation algorithm

$$\begin{aligned} E &= \{p_1, p_2, \ldots, p_E\} \\ &= \{(x_{p_1}, y_{p_1}), (x_{p_2}, y_{p_2}), \ldots, (x_{p_E}, y_{p_E})\} \end{aligned}$$

and

$$\begin{aligned} S &= \{q_1, q_2, \ldots, q_S\} \\ &= \{(x_{q_1}, y_{q_1}), (x_{q_2}, y_{q_2}), \ldots, (x_{q_S}, y_{q_S})\}. \end{aligned}$$

Then EBC is defined as

$$\text{EBC} = n(E \cap S)/n(E) \tag{1}$$

where $n(\cdot)$ computes the number of elements of its argument and

$$E \cap S = \{(x, y) \mid (x, y) \in E \text{ and } (x, y) \in S\}.$$

Fig. 4(b) shows the Sobel edge image of an experimental indoor color image given in Fig. 4(a). The region boundaries obtained using the *Phoenix* image segmentation algorithm are shown in Fig. 4(c). The parameters of the *Phoenix* algorithm that are used include: $Hsmooth = 7$, $Maxmin = 128$, $Splitmin = 47$, $Height = 60$. In this example, EBC for the segmented image

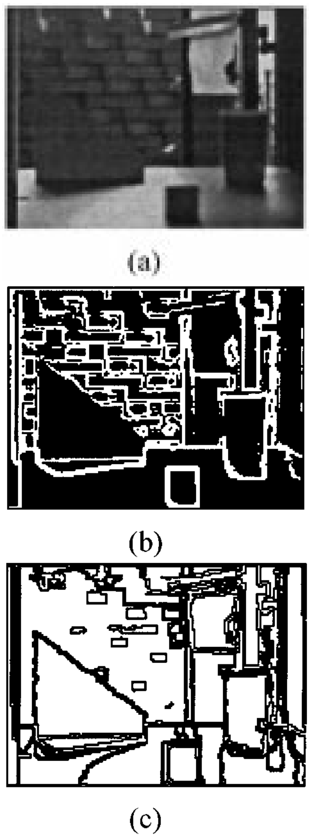

Fig. 4. Edge-border coincidence (EBC): (a) input image, (b) Sobel edge magnitude image (threshold $= 200$), and (c) boundaries of the segmented image.

is 0.6825. EBC measures, to the extent possible, the quality of the segmentation process. Matching confidence, the recognition system's output, indicates the confidence of the model matching process, and indirectly measures the segmentation quality of the recognized object. It is possible that EBC is high while matching confidence level is low, or EBC is low while matching confidence is high. Fig. 5(a) shows that EBC, when measuring segmentation quality for the whole image, does not correlate well with matching confidence. On the other hand, EBC does correlate with the matching confidence when applied to subimages, as shown in Fig. 5(b). However, the model matching confidence is arguably the only measure that can conclusively evaluate the performance of the segmentation process.

Although EBC does not correctly predict the matching confidence, for our purpose it is sufficient to drive initial estimates. Moreover, low EBC values indicate that the segmentation is unlikely to result in good recognition performance. As such, the system repeats the initial estimation process using EBC as the only reinforcement signal until it exceeds a prespecified level. At that time, the segmentation performance will be determined completely by model matching.

C. Reinforcement Learning for Image Segmentation

Reinforcement learning is the problem faced by an agent that must learn behavior through trial-and-error interactions with a dynamic environment. It is appropriately thought of as a class of problems, rather than as a set of techniques [13], [25]. This type of learning has a wide variety of applications, ranging from modeling behavior learning in experimental psychology to building active vision systems. The term *reinforcement* comes from studies of animal learning in experimental psychology. The basic idea is that if an action is followed by a satisfactory state of affairs or an improvement in the state of affairs, then the tendency to produce that action is reinforced. Reinforcement learning is similar to supervised learning in that it receives a feedback to adjust itself. However, the feedback is *evaluative* in the case of reinforcement learning. In general, reinforcement learning is more widely applicable than supervised learning and it provides a competitive approach to building autonomous learning systems that must operate in real world. For a comprehensive overview of this subject, the reader is referred to [25].

The particular class of reinforcement learning algorithms employed in each (global/local) region for our object recognition system is the connectionist REINFORCE algorithm [27], where units in such a network are *Bernoulli quasilinear units*, in that the output of such a unit is either 0 or 1, determined stochastically using the Bernoulli distribution with parameter $p_i = f(s_i)$, where f is the logistic function

$$f(s_i) = 1/(1+\exp(-s_i)) \tag{2}$$

and $s_i = \sum_j w_{ij}x_j$ is the weighted summation of input values to that unit. For such a unit, shown in Fig. 6, p_i represents its probability of choosing 1 as its output value.

In the general reinforcement learning paradigm, the network generates an output pattern and the environment responds by providing the reinforcement r as its evaluation of that output pattern, which is then used to drive the weight changes according to the particular reinforcement learning algorithm being used by the network. REINFORCE has the following generic update rule

$$\Delta w_{ij} = \alpha_{ij}(r - b_{ij})\frac{\partial}{\partial w_{ij}}\ln(g_j) \tag{3}$$

where

α_{ij}	learning rate factor;
r	immediate reinforcement;
b_{ij}	baseline;
g_j	density function for randomly generating output patterns.

For the Bernoulli quasilinear units used in this research, the above weight updating rule (3) is reduced to

$$\Delta w_{ij} = \alpha(r-b)(y_i - p_i)x_j \tag{4}$$

where

x_j	input to each Bernoulli unit;
y_i	output of the ith Bernoulli unit;
p_i	internal parameter to a Bernoulli random number generator.

It can be shown [27] that, regardless of how b is computed, whenever it does not depend on the immediately received reinforcement value r, and when r is sent to all the units in the network, such an algorithm satisfies

$$E\{\Delta\mathbf{W} \mid \mathbf{W}\} = \alpha\nabla_{\mathbf{W}}E\{r \mid \mathbf{W}\} \tag{5}$$

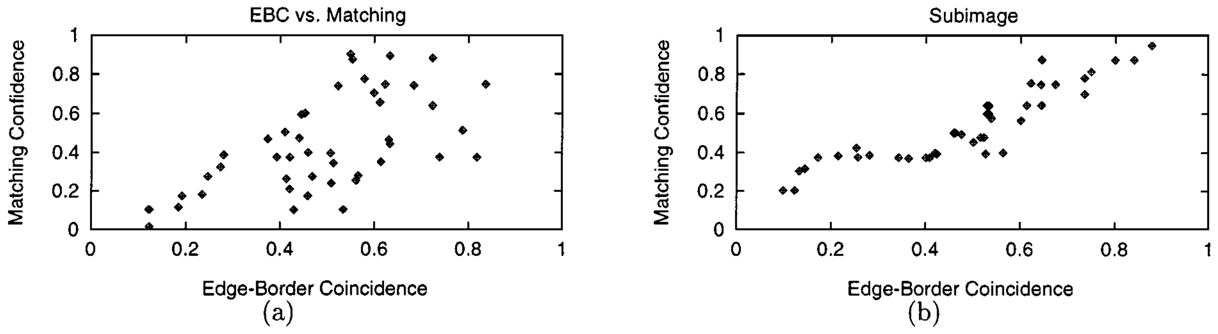

Fig. 5. Edge-border coincidence versus matching confidence for recognizing the cup in the image shown in Fig. 2(a). (a) Global image and (b) subimage.

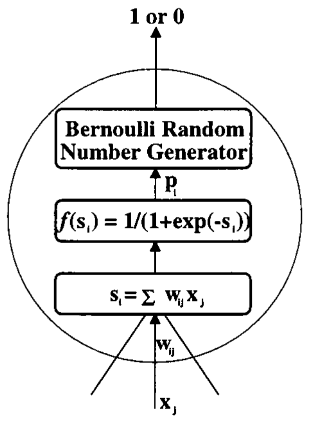
1 or 0
Bernoulli Random Number Generator
p_i
$f(s_i) = 1/(1+\exp(-s_i))$
$s_i = \Sigma\ w_{ij} x_j$
w_{ij}
x_j

Fig. 6. Bernoulli semilinear unit.

where E denotes the expectation operator. Here $\mathbf{W}$ represents the weight matrix of the network, that is, $\mathbf{W}_{ij}$ denotes the jth input weight to the ith unit in the network, and $\Delta\mathbf{W}$ is the change of the weight matrix. A reinforcement learning algorithm satisfying the above equation has the convergence property that the algorithm statistically climbs the gradient of expected reinforcement in weight space. For adapting parameters of the segmentation algorithm, it means that the segmentation parameters change in the direction along which the expected edge-border coincidence and/or matching confidence increases. It should be noted that the algorithm is scale variant in that the rate of convergence decreases with increasing weight space.

The specific algorithm we use here has the following form: At the tth time step, after generating output $\mathbf{y}(t)$ and receiving reinforcement $r(t)$, i.e., EBC or the confidence level indicating the matching result, increment each weight w_{ij} by

$$\Delta w_{ij}(t) = \alpha(r(t) - \bar{r}(t-1))(y_i(t) - \bar{y}_i(t-1))x_j - \delta w_{ij}(t) \tag{6}$$

where α, the learning rate, and δ, the weight decay rate, are parameters of the algorithm. The term $(r(t) - \bar{r}(t-1))$ is called the *reinforcement factor* and $(y_i(t) - \bar{y}_i(t-1))$ the *eligibility* of the weight w_{ij} [27]. Generally, the eligibility of a weight indicates the extent to which the activity at the input of the weight was connected in the past with unit output activity. Note that this algorithm is a variant of the one described in (4), where b is replaced by $\bar{r}$ and p_i by $\bar{y}_i$.

$\bar{r}(t)$ is the exponentially weighted average, or *trace*, of prior reinforcement values

$$\bar{r}(t) = \gamma\bar{r}(t-1) + (1-\gamma)r(t) \tag{7}$$

with $\bar{r}(0) = 0$. The trace parameter γ was set equal to 0.9 for all the experiments reported here. Similarly $\bar{y}_i(t)$ is an average of past values of y_i computed by the same exponential weighting scheme used for $\bar{r}$. That is

$$\bar{y}_i(t) = \gamma\bar{y}_i(t-1) + (1-\gamma)y_i(t). \tag{8}$$

Note that (5) does not depend on the eligibility. Note also that p_i in (4) is the theoretical mean of y_i, whereas $\bar{y}_i$ in (6) and (8) is the actual estimate. Empirical study has shown superior performance with this form of weight update [28]. Note that a team of 20 Bernoulli units represents the four image segmentation parameters selected for learning. Each bit of a parameter is independent of each other.

D. "Biased" Reinforcement Learning

In the RL algorithm as described in Section II-C, each of the bits encoding segmentation parameters chooses its output independently according to

$$y_i = \begin{cases} 1, & \text{with probability } p_i \\ 0, & \text{with probability } 1 - p_i. \end{cases} \tag{9}$$

It is "unbiased" in that the output of a bit is governed solely by the Bernoulli probability distribution. The advantage of this algorithm is that rapid changes in output values allow giant leaps in the search space, which in turn enables the learning system to quickly discover suspected high pay-off regions. However, once the system has arrived at the vicinity of a local optimum, as will be the case after the initial estimation, changes in the most significant bit will drastically alter the parameter value, often jumping out of the neighborhood of the local optimum. Ideally, once the learning system discovers that it is within a possible high pay-off region, it should attempt to capture the regularities of the region. This then biases future search toward points within it. The challenge, of course, is to have a learning algorithm that allows the parameters controlling the search distribution to be adjusted so that this distribution comes to capture this knowledge. The algorithm described here shows some promise in this regard.

Suppose that a parameter is represented by n Bernoulli units of which m are deemed significant. Here a unit is significant if

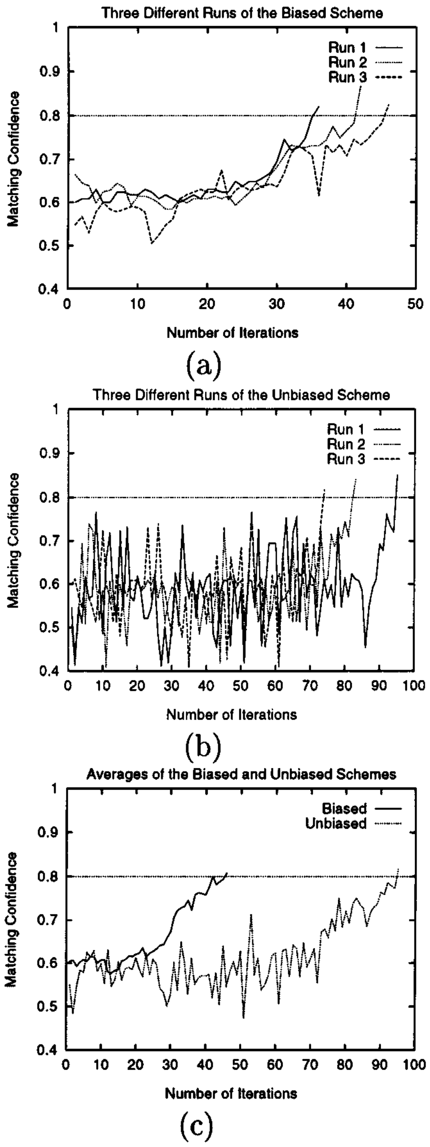

Fig. 7. Matching confidence history of three runs of the biased and unbiased RL algorithms on the image shown in Fig. 4(a). (a) Biased, (b) unbiased, and (c) average of these runs.

a change in its value causes a large change in parameter values. In order to force parameters to change slowly, after initial estimation, we apply a *biased* RL algorithm in which the m most significant units of a parameter are forced to change in a "lazy" fashion as

$$y_i = \begin{cases} 1, & \text{if } p_i > 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

while the remaining $n - m$ units update their outputs using the same rule (9). That is, a significant bit changes its output (selecting 0 or 1) in a less random fashion. Fig. 7 shows the experimental results of the two schemes on the image shown in Fig. 4(a). In this experiment, $m = 2$ and we only apply the initialization followed by global learning without switching between global and local learning. The results show that the biased RL algorithm speeds up learning by a factor of 2 to 3.

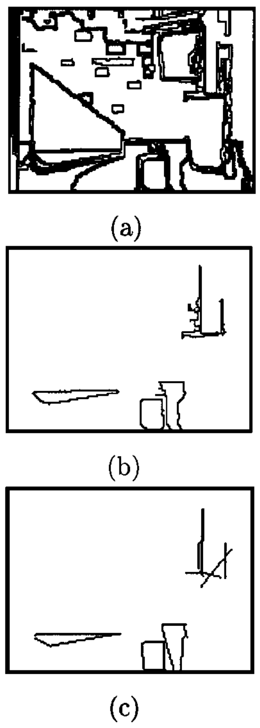

Fig. 8. (a) Boundaries of the segmented image shown in Fig. 2(a); (b) selected regions whose areas are in the expected range (200–450 pixels); and (c) polygonal approximation of the regions shown in (b) with fixed parameters as specified in this section.

```
• Main ()
    1. Let $\mathbf{W}_g$ and $\mathbf{W}_l$ be global and local weight matrices;
    2. Let $i$ be an input image;
    3. InitEstimate ($i$, $\mathbf{W} = \mathbf{W}_g$);
    4. $\mathbf{W}_l = \mathbf{W}_g$;
    5. RLglobal ($i$, $\mathbf{W} = \mathbf{W}_l$);
• end Main
```

Fig. 9. Main algorithm for adaptive integrated segmentation and recognition.

E. Feature Extraction and Model Matching

Feature Extraction: Feature extraction consists of finding polygon approximation tokens for each connected component obtained after image segmentation. The polygonal approximation is implemented by calling the polygon approximation routine in *Khoros* [22]. The resulting polygon approximation is a vector image to store the result of the linear approximation. The image contains two points for each estimated line. The polygonal approximation has a fixed set of parameters: (a) minimal segment length for straight line -5. When the estimated straight line has a length less than this threshold, it is skipped over; (b) elimination percentage -0.1. Percentage of line length rejected to calculate parameters of the straight line; and (c) approximation error -0.6. Threshold Value for the approximation error. When the calculated error is greater than this value, the line is broken.

To speed up the learning process, we assume that we have the prior knowledge of the *approximate area* (number of pixels) of the object, and only those connected components whose area is comparable with the area of the model object are approximated

```
• InitEstimate ($i$, $\mathbf{W}$)
  • LOOP:
      1. Compute segmentation parameters based on (9) and segment image $i$;
      2. Compute $r$ based on (1);
      3. Update $\mathbf{W}$ using $r$ as reinforcement according to (6);
  • UNTIL $r \geq EB1$;
• end InitEstimate
```

Fig. 10. Initial estimation procedure.

```
• RLglobal ($i$, $\mathbf{W}$)
  • $n = 1$; $r = 0$;
  • While $r < Accept$ and $n \leq MaxGlobal$ do
      1. $n = n + 1$;
      2. Compute segmentation parameters based on (10) and segment image $i$;
      3. Extract features (i.e., compute polygon approximation tokens) for each
         blob within size constraints and perform model matching;
      4. Let $r$ be highest matching confidence;
      5. Update $\mathbf{W}$ using $r$ as reinforcement according to (6);
      6. if $r \geq Switch$ then
      7.    For each blob $B$ and $r < Accept$ do
      8.       $r$ = RLlocal ($B$, $\mathbf{W} = \mathbf{W}_l$);
• end RLglobal
• RLlocal ($i$, $\mathbf{W}$)
  • $n = 1$; $r = 0$;
  • While $r < Accept$ and $n \leq MaxLocal$ do
      1. $n = n + 1$;
      2. Compute segmentation parameters based on (10) and segment $i$;
      3. Extract features (i.e., compute polygon approximation tokens) for each
         blob within size constraints and perform model matching;
      4. Let $r$ be highest matching confidence;
      5. Update $\mathbf{W}$ using $r$ as reinforcement according to (6);
  • if $r \geq Accept$ then $\mathbf{W}_l = \mathbf{W}$;
  • Return $r$
• end RLlocal
```

Fig. 11. Global/local adaptation procedures.

by a polygon. For example, in our experiment on indoor images 4(a), the cup is the target object. The expected area is from 200 to 450 pixels. Fig. 8 shows the boundaries of a segmented image, selected regions whose areas are in the expected range, and the polygonal approximation of these regions, where segmentation parameters are: $Hsmooth = 7$, $Maxmin = 128$, $Splitmin = 47$, $Height = 54$.

Model Matching: Model matching employs a cluster-structure matching algorithm [7] which is based on forming the clusters of translational and rotational transformations between the object and the model. The algorithm takes as input two sets of tokens, one of which represents the stored model and the other represents the input region to be recognized. It then performs topological matching between the two token sets and computes a real number that indicates the confidence level of the matching process. Basically, the technique consists of three steps: clustering of border segment transformations; finding continuous sequences of segments in appropriately chosen clusters; and clustering of sequence average transformation values. In the current implementation this algorithm can handle models and objects of various sizes. In fact, there can be up to ±30% change in scale from model to object. A brief description of the algorithm is given in Appendix B. More details about this algorithm can be found in [7].

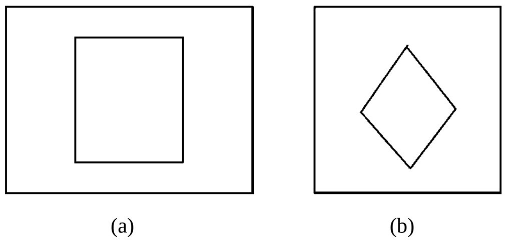

Fig. 12. Models: (a) cup and (b) traffic sign.

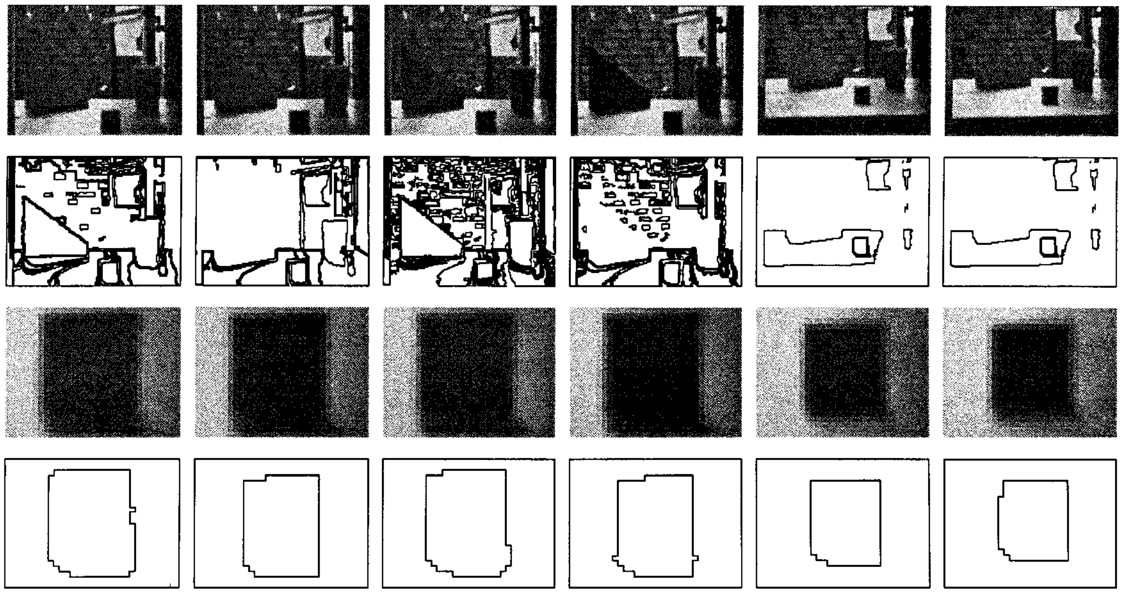
Fig. 13. Row 1: input images 1–6; rows 2–4: corresponding segmentations, subimages, and recognized objects.

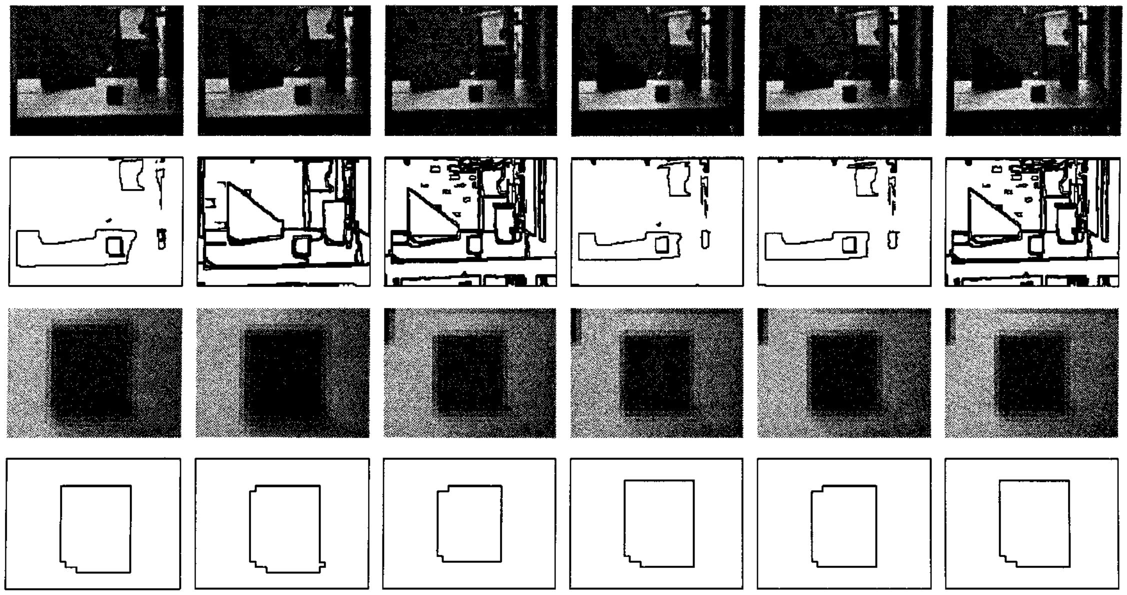
Fig. 14. Row 1: input images 7–12; rows 2–4: corresponding segmentations, subimages, and recognized objects.

F. Algorithm Description

This section describes the detailed implementation of our algorithm. For each input image the main procedure (Fig. 9) initializes $\mathbf{W}_g$ (global weight matrix) and $\mathbf{W}_l$ (local weight matrix), and then calls procedures **InitEstimate ()** and **RLglobal ()** to adaptively compute segmentation parameters in order to carry out optimal model matching. Learning is continuous and on-line, and there are three distinct phases. For a given image, learning begins with $\mathbf{W}_g$ and $\mathbf{W}_l$. The initial estimation does not involve the polygonal approximation and model matching. $\mathbf{W}_g$ is updated according to (6) using edge-border coincidence (1) as reinforcement. The initial estimation terminates when EBC has exceeded *EB1* (a threshold parameter) at which point global adaptation begins. The main steps of initial estimation are shown in Fig. 10.

In global adaptation, $\mathbf{W}_g$ is updated using the lazy learning strategy [(6) and (10)] driven by model matching. When the matching confidence is sufficiently large ($\geq$*Switch*—a threshold parameter), local adaptation kicks in with connected components as input whose size passes a region filter. The global adaptation process terminates when either a given number (*MaxGlobal*) of iterations has been reached in which case the model object is declared not to be present in the image, or the matching confidence exceeds a acceptable level (*Accept*) in which case a successful recognition has been achieved.

Similar to global adaptation, model matching drives the update of $\mathbf{W}_l$ in local adaptation. The local adaptation procedure

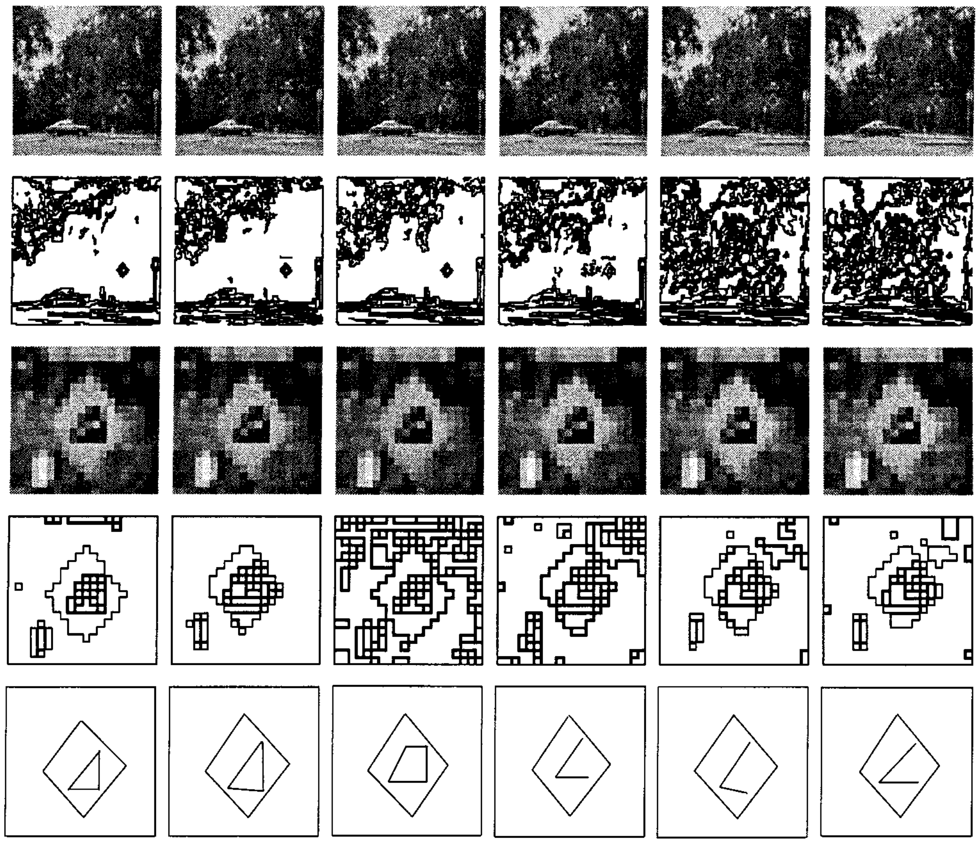
Fig. 15. Row 1: input images 1–6; rows 2–5: corresponding global segmentations, subimages, segmented images, and recognized objects.

terminates when one of the following two conditions is met: (1) the matching confidence exceeds *Accept*, indicating that the connected component is the recognized model object; (2) the number of iterations reaches *MaxLocal* in which case the model object is unlikely to be extracted from the subimage and local weight changes are discarded. The global and local procedures are shown in Fig. 11.

It should be stated that while many procedural parameters have been introduced to optimize four segmentation parameters, these procedural parameters, to a large extent, do not depend on inputs and can be simply determined *a priori*. In addition, the combined search space of these procedural parameters is much smaller than that of the four segmentation parameters. Moreover, the convergence of the algorithm to a local optimum does not depend on some of the thresholds mentioned, such as γ and δ [see (5)–(7)]. In practice, however, these threshold parameters affect only the speed of convergence, as shown by various empirical studies conducted by several researchers including us [20], [28]. Clearly, α [see (5)] has to be chosen sufficiently small to prevent oscillation. The claim that good performance can be obtained under a wide range of these parameter values is that once the algorithm has converged, many of these values give rise to similar segmentation performance, as verified visually. The complexity of the method can be justified by

1) necessity to adapt segmentation to changes in image characteristics caused by changes in external conditions;
2) large size of the search space (10^6);
3) quality of the results to be shown in the next section.

Therefore, the advantages of the proposed method outweigh its disadvantages in practical applications.

III. Experimental Results

The system is verified on a set of 12 indoor and a set of 12 outdoor color images. The indoor images are acquired at different viewing distances with varying lighting conditions. The outdoor images are collected every 15 min over a 3-h period using a JVC GXF700U color video camera. These images simulate a photo interpretation/surveillance scenario in which the camera position is fixed and the images exhibit significant changes over time due to changing environmental conditions (time of the day, position of the sun in the sky, and cloud cover). Varying light level is the most prominent change throughout the outdoor images. Although the environmental conditions also created varying object highlights, moving shadows and many subtle contrast changes between the objects in the image. The size of indoor images is 120 by 160 pixels, and the size of outdoor images is 120 by 120 pixels. Each image is decomposed into four images for *Phoenix* segmentation—red, green, blue components, and the Y component of YIQ model of color images. For the indoor images, the desired object is the cup in the image, whereas the target object is the traffic sign for the outdoor images. The expected sizes of the cup and the traffic sign are 200 to 450 pixels and 36 to 100 pixels, respectively.

Based on the size of the object to be recognized in the image, we divide the Y component image into 48 subimages for the indoor images, and 36 subimages for the outdoor images. The size

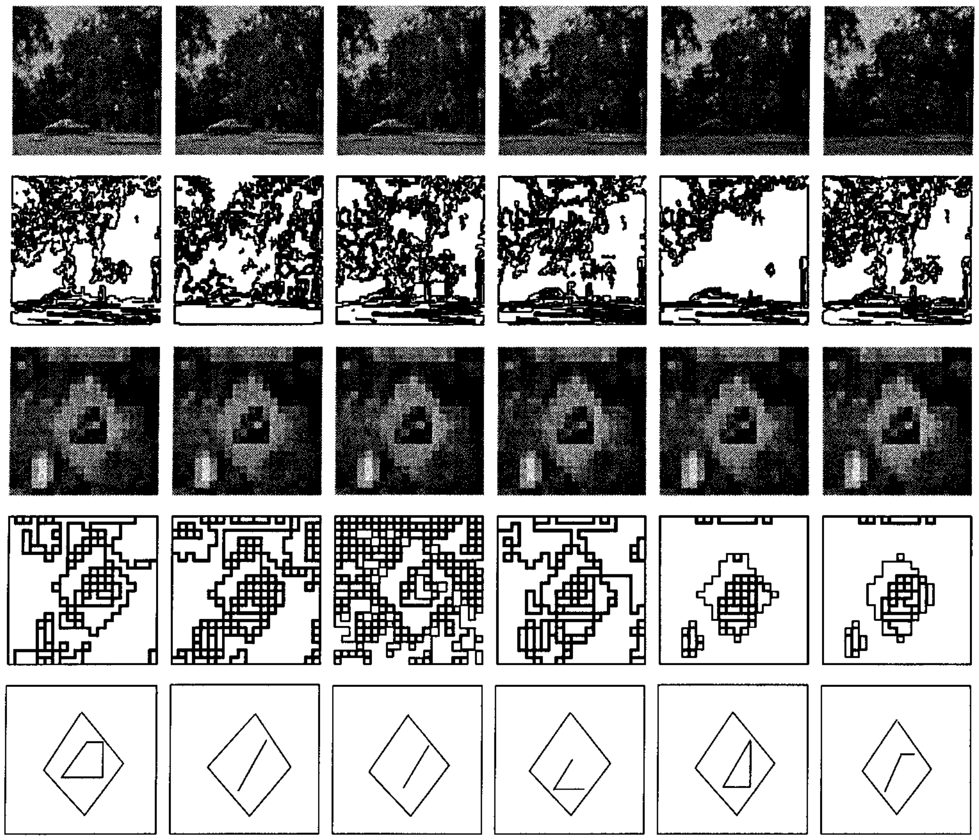

Fig. 16. Row 1: input images 7–12; rows 2–5: corresponding global segmentations, subimages, segmented subimages, and recognized objects.

TABLE II
LEARNED SEGMENTATION PARAMETERS AND MATCHING CONFIDENCE VALUES FOR INDOOR IMAGES

Image	Hsmooth	Maxmin	Splitmin	Height	Confidence	Image	Hsmooth	Maxmin	Splitmin	Height	Confidence
1	7	122	47	52	0.87	7	7	122	47	52	0.81
2	7	128	47	52	0.93	8	7	471	11	60	0.86
3	7	471	11	60	0.86	9	7	471	11	60	0.87
4	5	471	19	58	0.91	10	5	471	19	58	0.91
5	11	192	59	48	0.92	11	9	157	61	66	0.93
6	9	157	59	61	0.97	12	9	157	59	61	0.91

TABLE III
LEARNED SEGMENTATION PARAMETERS AND MATCHING CONFIDENCE VALUES FOR OUTDOOR IMAGES

Image	Hsmooth	Maxmin	Splitmin	Height	Confidence	Image	Hsmooth	Maxmin	Splitmin	Height	Confidence
1	11	367	43	26	0.82	7	11	300	69	40	0.88
2	9	259	69	46	0.91	8	11	259	23	46	0.85
3	15	259	23	54	0.86	9	13	212	27	40	0.86
4	11	234	27	56	0.90	10	11	259	29	56	0.88
5	13	264	31	56	0.90	11	11	289	29	54	0.87
6	13	276	29	54	0.91	12	9	276	31	46	0.92

of each subimage is 20 by 20 pixels. The standard deviations of these subimages serve as inputs to each Bernoulli unit, i.e., each Bernoulli unit has a total of 48 inputs (and therefore, 48 weights) for the indoor image, and has a total of 36 inputs (36 weights) for the outdoor image. In order to learn the four selected *Phoenix* segmentation parameters, we need 20 Bernoulli units. So there are a total of 980 weights ($\mathbf{W} = 20 \times 49$) for the indoor images, and 740 ($\mathbf{W} = 20 \times 37$) for the outdoor images. Note that because Bernoulli units are independent, the effective number of free parameters is 49 for the indoor images, 37 for the outdoor ones.

For the team of 20 Bernoulli units, the parameters α, γ, and δ are determined empirically, and they are kept constant for all images. In our experiments, $\alpha = 0.02, \gamma = 0.9, \delta = 0.01$, $EB1 =$ 0.5, $MaxGlobal = 20$, and $MaxLocal = 20$. The threshold for matching confidence $Switch = 0.6$, and $Accept = 0.8$.

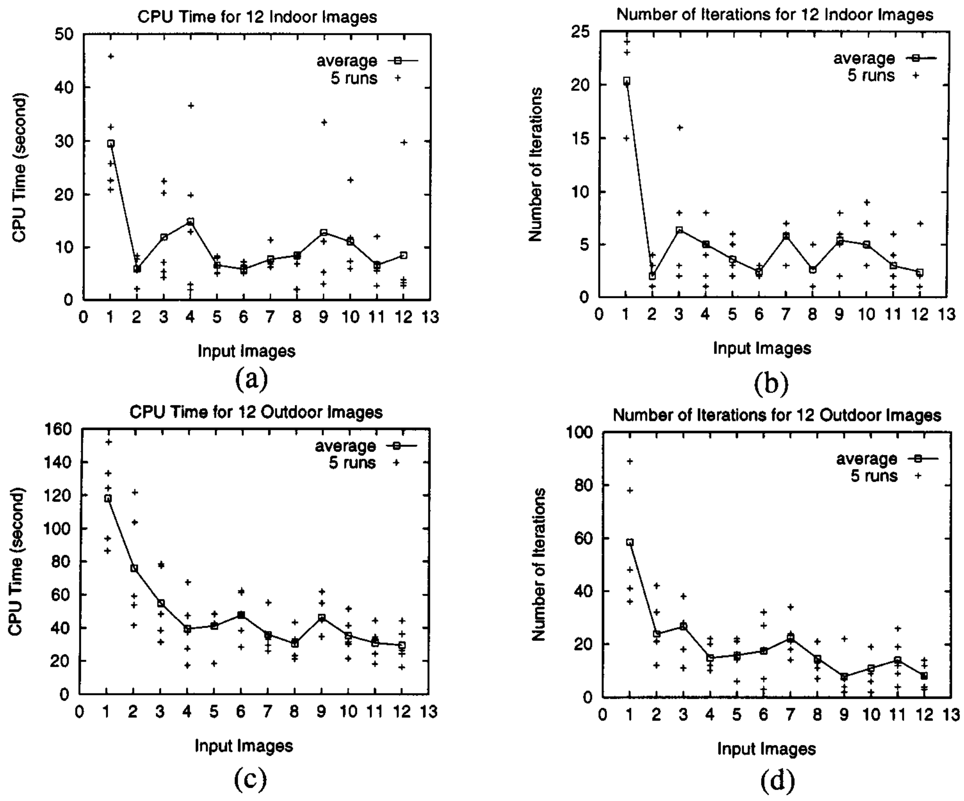

Fig. 17. (a) CPU time for five different runs on 12 indoor images and the average; (b) number of iterations for five different runs on 12 indoor images and the average; (c) CPU time for five different runs on 12 outdoor images; and (d) number of iterations for five different runs on 12 outdoor images.

Threshold used for extracting edges using Sobel operator is set at 200. The parameters for feature extraction are fixed as specified in Section II-E. Fig. 12 shows the stored models for the cup in indoor images and the traffic sign in outdoor images. Note that during local adaptation, we extract and enlarge subimages by a factor such that the enlarged subimages are of the same size as the input image. The stored models are scaled by the same factor.

A. Results on Indoor and Outdoor Images

Figs. 13–16 show the experimental results on the 12 indoor and 12 outdoor color images. For each indoor image, its segmentation using the set of learned parameters and the extracted object that has been recognized are presented. For each outdoor image, corresponding global segmentation, subimage, segmented subimage and recognized object are shown. Subimages are computed automatically by our algorithm. For each set of images, inputs are taken sequentially. Except for the first image, the learning process for each image starts from the global segmentation parameters learned from all the previous images. For the first input image, weights are initialized randomly. Usually, it takes less than 45 iterations to find a set of segmentation algorithm parameters that produces high edge-border coincidence. Tables II and III show four learned segmentation parameter values and matching confidence after local adaptation for indoor and outdoor images, respectively. Accuracy of segmentation results can be observed from Figs. 13–16 and the matching confidence in Tables II and III.

Fig. 17 shows the CPU time for the 12 indoor images and 12 outdoor images for five different runs, and the number of iterations for each input image, which is the sum of all the iterations involved in the global learning and local learning processes. These two curves show the learning capability of the system, i.e., the system uses less and less CPU time with experience to find a set of segmentation parameters and correctly recognizes the object. The number of iterations decreases with the accumulation of experience.

B. Comparison of Approaches

In this section we compare the performance of our system as shown in Fig. 1 with an earlier approach [20]. We show the effect of incorporating segmentation evaluation using the edge-border coincidence into the learning system and the impact of global and local segmentations on model matching.

The key differences between the two methods are the introduction of the local segmentation process, the biasing of RL algorithm, and the use of edge-border coincidence as an evaluation of the segmentation performance during the early stage of learning in order to reduce the computational expense stemming from model matching. In the method presented in this paper the segmentation process alternates between the whole image and its subcomponents. The local segmentation is highly desirable when there are multiple objects or a single object at multiple locations with different local characteristics. It can dramatically improve the recognition performance. The biasing of RL algorithm reduces computational time as illustrated in Fig. 7.

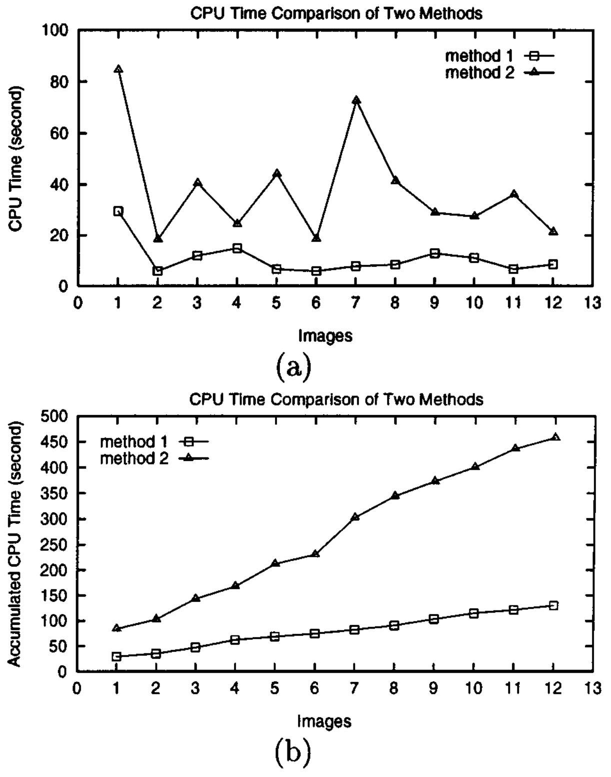

Fig. 18. Comparison of two approaches: method 1 approach presented in this paper, method 2 approach reported in [20]. (a) Comparison of the average CPU time of five different runs on 12 indoor images and (b) comparison of the accumulated average CPU time of five different runs on 12 indoor images.

In previous research [20], the matching confidence is the only feedback that drives learning. Although it is undoubtedly the most reliable measure, it is relatively expensive to compute. The edge-border coincidence provides us with a cheap way to find a good point from which to begin the more expensive search for high matching confidence values. Fig. 18 shows the comparison results of the two methods: method 1 (this paper) and method 2 [20]. The improvement in CPU time for method 2 and its consistency for different images are very clear. Although good initial estimates may not always result in faster discovery of high matching confidence values, the edge-border coincidence seems to work well in practice for all the problems we have experimented.

IV. Conclusions and Future Work

We have presented a proof-of-the-principle of a general approach for adaptive image segmentation and object recognition. The approach combines a domain independent simple measure for segmentation evaluation (edge-border coincidence) and domain dependent model matching confidence in a reinforcement learning framework in a systematic manner to accomplish robust image segmentation and object recognition simultaneously. Experimental results demonstrate that the approach is suitable for continuously adapting to normal changes encountered in real-world applications. Although the empirical evaluation involves a small number of images, our system is designed to continuously adapt in a real-time environment, such as robot navigation, where image streams would provide sufficient information to confine free parameters in our system.

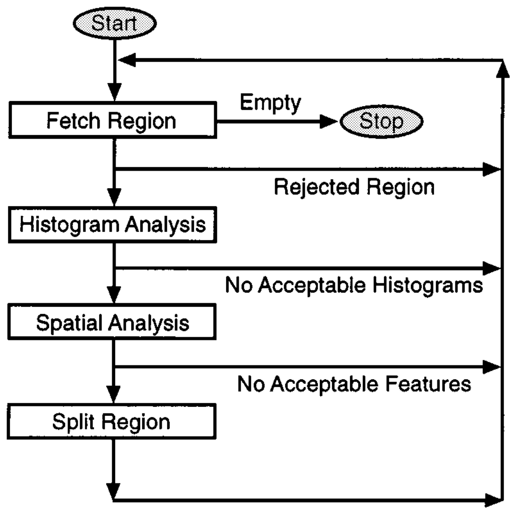

Fig. 19. Conceptual diagram of the Phoenix segmentation algorithm.

For adapting to the wide variety of images encountered in real-world applications, we can develop an autonomous gain control system which will allow the matching between different classes of images taken under significantly different weather conditions (sunny, cloudy, snowy, rainy) and adapt the parameters within each class of images. We use image context to divide the input images into several classes based on image properties and external conditions, such as time of the day, lighting condition, etc. [5]. When an image is presented, we use an image property measurement module and the available external information to find the stored information for this category of images, and start learning process from that set of parameters. This will overcome the problem of adapting to large variations between consecutive images. It is also possible to use our approach for classification of segmentation methods or as a testbed for new segmentation methods, where the evaluation is conducted from object recognition point of view. These are areas of future research that we intend to pursue.

Appendix A
Phoenix Segmentation Algorithm

The Phoenix image segmentation algorithm is based on a recursive region splitting technique [14]. It uses information from the histograms of the red, green, and blue image components to split regions in the image into smaller subregions on the basis of a peak/valley analysis of each histogram. An input image typically consists of red, green, and blue image planes, although monochrome images, texture planes, and other pixel-oriented data may also be used. Each plane is called a feature or feature plane.

Fig. 19 shows a conceptual description of the Phoenix segmentation process. It begins with the entire image as a single region. It then fetches this region and attempts to segment it using histogram and spatial analyses. If it succeeds, the program fetches each of the new regions in turn and attempts to segment them. The process terminates when no region can be further segmented.

The histogram analysis phase computes a histogram for each feature plane, analyzes it and selects thresholds or histogram cutpoints that are likely to identify significant homogeneous regions in the image. A set of thresholds for one feature is called an interval set. During the analysis, a histogram is first smoothed with an unweighted window average, where the window width is *Hsmooth*. It is then broken into intervals such that each contains a peak and two valleys, called "shoulder," on either side of the peak. A series of heuristics is applied to eliminate noise peaks. When an interval is removed, it is merged with the neighbor sharing the higher of its two shoulders. *Splitmin* is the minimum area for a region to be automatically considered for splitting.

Two tests determine if an interval should be retained. First, the ratio of peak height to the height of its higher shoulder must be greater than or equal to the *Maxmin* threshold. Second, the interval area must be larger than an absolute threshold and the relative area, percent of the total histogram area. The second highest peak can now be found, and peaks lower than the *Height* percent of this peak are merged. The lowest valley is then determined, and any interval whose right shoulder is higher than *absmin* (Phoenix's parameter) times this valley is merged with its right neighbor. Finally, only *intsmax* (Phoenix's parameter) intervals are retained by repeatedly merging intervals with low peak-to-shoulder ratio.

The spatial analysis selects the most promising interval sets, thresholds the corresponding feature planes, and extracts connected components for spatial evaluation. The feature and the interval set providing the best segmentation (the least noise area) are accepted as the segmentation feature and the thresholds.

The histogram cutpoints are now applied to the feature plane as intensity thresholds and connected components are extracted. After each feature has been evaluated, the one producing the least total noise area is accepted as the segmentation feature. If no suitable feature is found, the original region is declared terminal. Otherwise the valid patches, merged with the noise patches, are converted to new regions and added to the segmentation record. In either case, a new segmentation pass is scheduled. For additional details, see [14].

Appendix B
Cluster-Structure Algorithm for Matching

The cluster-structure algorithm can be divided into the following main steps:

1) determine disparity matrix;
2) initial clustering;
3) sequencing;
4) final clustering;
5) transform computation.

The algorithm first computes the disparity matrix. It determines the segment length of each line and the angles between successive lines from the set of vertices for the model and the image input to the program. At this point, every segment in the model will be compared against every segment in the image. If segment lengths and successor angles are compatible, the algorithm computes the rotational and translational disparity between pairs of segments. These values are stored in the disparity matrix and are indexed by the segment numbers in the model and the image. The algorithm continues until all segments have been compared. It then computes the range of rotational and translational values present in the matrix, and normalizes them over their appropriate range.

The initial clustering determines clusters from the normalized values in the disparity matrix. At each step, the program clusters all of the samples, recomputes the new cluster centers, and continues until none of the cluster centers change their positions. The program then selects the cluster having the largest number of samples. Also selected are the clusters that are within 20% of the largest one. Each cluster is considered separately and the final transform comes from the cluster that yields the highest confidence level.

The sequencing step uses the samples in the current cluster to find all sequences in the samples. This provides the critical structural information. Samples that are not placed in any sequence are discarded. The program also removes sequences that have a segment count of less than three (three segments comprise the basic local shape structure). It then computes the rotational and translation averages of each sequence that has been located.

Using the sequences and the sequence averages, the final clustering step clusters these values to find those sequences that lead to the same rotational and translational results. This is achieved by using the iterative technique of clustering, evaluating, clustering, etc. The program then selects the cluster that contains the largest number of sequences and passes this cluster to the final step.

The final step of the algorithm computes the confidence level of the transformation determined by each cluster. The cluster having the highest confidence level is selected as the final transformation cluster. It assembles the set of matched segments in the sequences in this cluster. The final output of the program is the rotation and the vertical and horizontal translation necessary to locate the model within the image. The program also produces a confidence level indicating the likelihood that the final matching is correct. For further details, see [7].

Acknowledgment

The authors would like to thank X. Bao for providing us with the experimental data.

References

[1] W. Au and B. Roberts, "Overview of a self-adaptive ATR system via context-based configuration and control," in *ARPA Image Understanding Workshop*, Palm Springs, CA, Feb. 1996, pp. 585–588.

[2] Y. Bengio and Y. Le Cun, "Word normalization for on-line handwritten word recognition," in *Proc. Int. Conf. Pattern Recognition*, Jerusalem, Israel, 1994, pp. 409–413.

[3] B. Bhanu, S. Lee, and J. Ming, "Adaptive image segmentation using a genetic algorithm," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, pp. 1543–1567, Dec. 1995.

[4] B. Bhanu, S. Lee, and S. Das, "Adaptive image segmentation using genetic and hybrid search methods," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 31, pp. 1268–1291, Oct. 1995.

[5] B. Bhanu and S. Lee, *Genetic Learning for Adaptive Image Segmentation*. Norwell, MA: Kluwer, 1994.

[6] B. Bhanu and T. L. Jones, "Image understanding research for automatic target recognition," *IEEE Aerosp. Electron. Syst.*, vol. 8, pp. 15–23, Oct. 1993.

[7] B. Bhanu and J. Ming, "Recognition of occluded objects: A cluster-structure algorithm," *Pattern Recognit.*, vol. 20, no. 2, pp. 199–211, 1987.

[8] C. Burges *et al.*, "Shortest path segmentation: A method for training a neural network to recognize character strings," in *Proc. Int. Joint Conf. Neural Networks*, vol. 3, Baltimore, MD, 1992, pp. 165–172.

[9] M. Dorigo and M. Colombetti, "Robot shaping: Developing autonomous agents through learning," *Artif. Intell.*, vol. 71, pp. 321–370, Dec. 1994.

[10] R. M. Haralick and L. G. Shapiro, "Image segmentation techniques," *Comput. Vis., Graph., Image Process.*, vol. 29, pp. 100–132, 1985.

[11] ——, *Computer and Robot Vision*. Reading, MA: Addison-Wesley, 1992, vol. 1.

[12] J. H. Holland, "Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems," in *Machine Learning: An Artificial Intelligence Approach, Vol. II*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds. San Mateo, CA: Morgan Kaufmann, 1986.

[13] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, 1996.

[14] K. Laws, "The Phoenix image segmentation system: Description and evaluation,", SRI Int. Tech. Rep. TR289, Dec. 1982.

[15] Y. LeCun, L. Bottou, and Y. Bengio, "Reading checks with graph transformer networks," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, vol. 1, 1997, pp. 151–154.

[16] R. Maclin and J. W. Shavlik, "Creating advice-taking reinforcement learners," *Mach. Learn.*, vol. 22, no. 1–3, pp. 251–281, 1996.

[17] T. Matsuyama, "Expert systems for image processing: Knowledge-based composition of image analysis processes," *Comput. Vis., Graph., Image Process.*, vol. 48, pp. 22–49, 1989.

[18] D. L. Milgram, "Region extraction using convergent series," *Comput. Graph., Image Process.*, vol. 11, pp. 1–12, 1979.

[19] R. Ohlander, K. Price, and D. R. Reddy, "Picture segmentation using a recursive region splitting method," *Comput. Graph., Image Process.*, vol. 8, pp. 313–333, 1978.

[20] J. Peng and B. Bhanu, "Closed-Loop object recognition using reinforcement learning," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 20, no. 2, pp. 139–154, 1998.

[21] ——, "Delayed reinforcement learning for adaptive image segmentation and feature extraction," *IEEE Trans. Syst., Man, Cybern. C*, vol. 28, pp. 482–488, Aug. 1998.

[22] J. Rasure and D. Argiro, *Khoros User's Manual*: Univ. New Mexico, Las Cruces, 1991.

[23] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*. New York: Academic, 1982.

[24] S. Shafer and T. Kanade, "Recursive region segmentation by analysis of histograms," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, 1982, pp. 1166–1171.

[25] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[26] S. Thrun and A. Schwartz, "Finding structure in reinforcement learning," in *Proc. Advances Neural Information Processing Systems 7*, Denver, CO, 1994, pp. 385–392.

[27] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, pp. 229–256, 1992.

[28] R. J. Williams and J. Peng, "Function optimization using connectionist reinforcement learning algorithms," *Connect. Sci.*, vol. 3, no. 3, 1991.

Bir Bhanu (S'72-M'82-SM'87-F'96) received the B.S. (with honors) degree in electronics engineering from the Institute of Technology, BHU, Varanasi, India, and the M.E. degree (with Distinction) in electronics engineering from Birla Institute of Technology and Science, Pilani, India. He received the S.M. and E.E. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, the Ph.D. degree in electrical engineering from the Image Processing Institute, University of Southern California, Los Angeles, and the M.B.A. degree from the University of California, Irvine.

He is the Director of the Center for Research in Intelligent Systems (CRIS), University of California, Riverside, where he has been a Professor and Director of Visualization and Intelligent Systems Laboratory (VISLab) since 1991. Previously, he was a Senior Honeywell Fellow at Honeywell, Inc., Minneapolis, MN. He has been on the faculty of the Department of Computer Science, University of Utah, Salt Lake City, and has worked at Ford Aerospace and Communications Corporation, INRIA-France, and IBM San Jose Research Laboratory, San Jose, CA. He has been the Principal Investigator of various programs for DARPA, NASA, NSF, AFOSR, ARO, and other agencies and industries in the areas of learning and vision, image understanding, pattern recognition, target recognition, navigation, image databases, and machine vision applications. He is the coauthor of books on *Computational Learning for Adaptive Computer Vision* (to be published), *Genetic Learning for Adaptive Image Segmentation* (Norwell, MA: Kluwer 1994), and *Qualitative Motion Understanding* (Norwell, MA: Kluwer 1992). He holds ten U.S. and international patents and more than 200 reviewed technical publications in his areas of interest. He was the Chair for the DARPA Image Understanding Workshop and is on the editorial board of various journals

Dr. Bhanu has received two outstanding paper awards from the Pattern Recognition Society and has received industrial awards for technical excellence, outstanding contributions and team efforts. He has been the guest editor of several IEEE Transactions and other journals. He has been General Chair for IEEE Workshops on Applications of Computer Vision, General Chair for the IEEE Conference on Computer Vision and Pattern Recognition, and Program Chair for the IEEE Workshops on Computer Vision Beyond the Visible Spectrum. He is a Fellow of the American Association for the Advancement of Science and the International Association of Pattern Recognition.

Jing Peng received the B.S. degree in computer science from Beijing Institute of Aeronautics and Astronautics, Beijing, China, the M.A. degree in computer science from Brandeis University, Waltham, MA, and the Ph.D. degree in computer science from Northeastern University, Boston, MA, in 1994.

Since 1999, he has been on the faculty of Oklahoma State University, Stillwater, where he is an Assistant Professor of computer science. Previously, he was a Research Scientist with the Center for Research in Intelligent Systems, University of California at Riverside. His research interests include machine learning, pattern classification, image databases, data mining, and learning and vision applications.