Autonomous Robots 6, 187–201 (1999)
© 1999 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Learning to Perceive Objects for Autonomous Navigation

JING PENG AND BIR BHANU
Center for Research in Intelligent Systems, University of California, Riverside, CA 92521, USA
jp@vislab.ucr.edu
bhanu@vislab.ucr.edu

Abstract. Current machine perception techniques that typically use segmentation followed by object recognition lack the required robustness to cope with the large variety of situations encountered in real-world navigation. Many existing techniques are brittle in the sense that even minor changes in the expected task environment (e.g., different lighting conditions, geometrical distortion, etc.) can severely degrade the performance of the system or even make it fail completely. In this paper we present a system that achieves robust performance by using local reinforcement learning to induce a highly adaptive mapping from input images to segmentation strategies for successful recognition. This is accomplished by using the confidence level of model matching as reinforcement to drive learning. Local reinforcement learning gives rises to better improvement in recognition performance. The system is verified through experiments on a large set of real images of traffic signs.

Keywords: landmark recognition, learning in computer vision, local learning, recognition feedback, reinforcement learning, traffic sign recognition

1. Introduction

Sensing and perception are of paramount importance to any cognitive automaton. Acquiring such abilities is a prerequisite for autonomous platforms that must operate in a dynamic environment. This objective, however, can be challenging in real world navigation applications due to the presence of clutter, object occlusion, data uncertainty, limited a priori model information, and changes in the environmental conditions that cannot be controlled in the outdoor scenarios. As an example, suppose an autonomous platform is tasked with a mail delivery mission to each department on a typical university campus. For this scenario, the platform must recognize traffic signs and act accordingly, for example, stop at stop signs. Sample color images (printed here in black and white) are shown in Fig. 1. There exists a wide variety of real world conditions under which the platform may come to a stop sign. As such, it is difficult or even impossible to develop a perception strategy with fixed parameters and algorithms that performs reliably in dynamic environments. Real world considerations mentioned above must be taken into account if the platform is to act intelligently in an autonomous fashion in a dynamic environment.

In addition to recognizing landmarks/objects in a robust manner, an autonomous platform must be capable of estimating its 3D position and performing spatial-temporal reasoning within its environment. The ability to provide accurate spatial information by means of active vision, stereo/motion analysis and/or statistical techniques (Baumgartner and Skarr, 1994; Olson, 1997) allows the autonomous platform to register its successive local contexts indefinitely in one reference frame, whereby its state, i.e., its position and orientation relative to a global reference system can be determined. Higher-level vision and other cognitive activities such as path planning can then be carried out through reasoning about landmarks within a 3D model of the environment so as to navigate effectively in the world. It is also desired that the autonomous platform be capable of handling tasks having immediate importance to it, such as obstacle avoidance, and taking appropriate actions in a rapid and reactive manner.

evaluating our method. These results are discussed and analyzed. Finally, we conclude with the key aspects of this paper.

2. Why Learning?

A typical 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 processing and computer vision tasks. All subsequent image interpretation tasks including feature extraction and model matching, rely heavily on the quality of the image segmentation process. Generally, this in turn spells out the difference between success and failure in vision-based autonomous navigation.

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. Despite the large number of image segmentation algorithms available, no general methods have been found to process the wide diversity of images encountered in real world applications. Typical object recognition systems are *open-loop*. Segmentation and feature extraction modules use default algorithm parameters, and generally serve as pre-processing steps to the model matching component. These parameters are not reliable, since when the conditions for which they are designed are changed slightly, these algorithms generally fail without any graceful degradation in performance. As an example, Fig. 2 shows segmentation of images shown in Fig. 1 obtained using the *Phoenix* algorithm (Laws, 1982) with default parameters. From these segmentation results, no algorithm would be able to perform model matching with sufficient confidence to recognize the stop sign, i.e., the octagon. Moreover, purely geometric or physics-based invariant approaches, *without* learning, are not sufficient to recognize objects under a wide variety of situations encountered in real-world navigation (Forsyth et al., 1992; Healey and Jain, 1996).

One might contemplate the idea of using color information for detecting and recognizing objects, such as stop signs. However, there are three major difficulties associated with such simple, color-based techniques. *First*, there are times at which color features cannot be reliably detected. For example, the images shown in

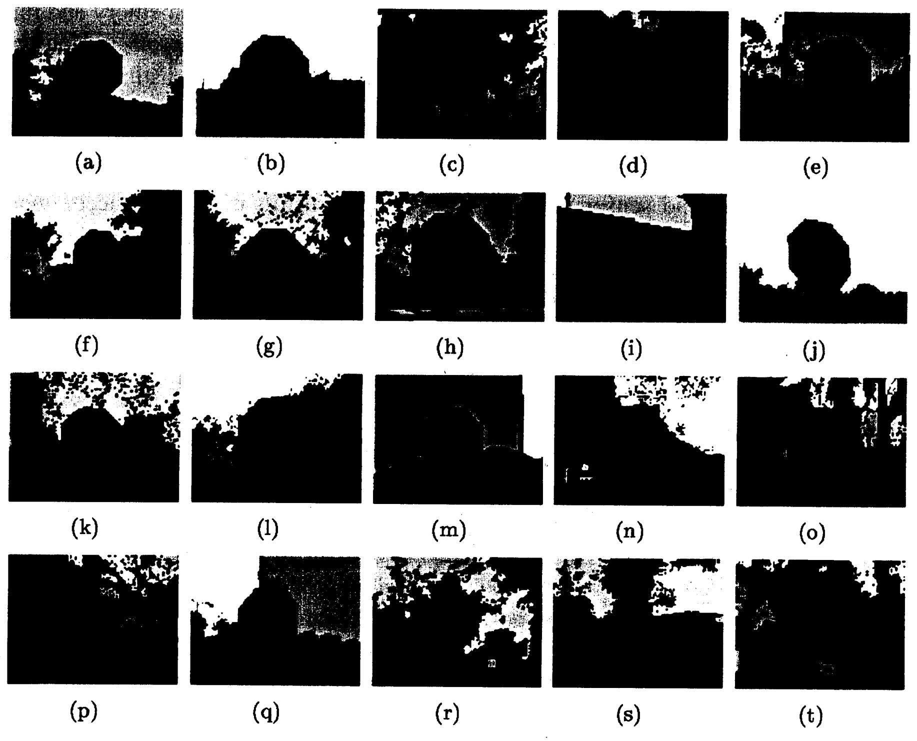


Figure 2. Segmentations of images shown in Fig. 1 using the *Phoenix* algorithm with default parameters.

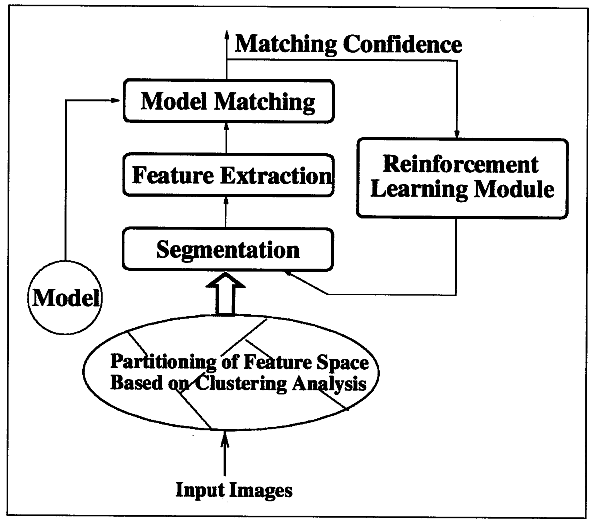

Matching Confidence
Model Matching
Feature Extraction
Reinforcement Learning Module
Segmentation
Model
Partitioning of Feature Space Based on Clustering Analysis
Input Images

Figure 3. Reinforcement learning based system for recognition.

image segmentation component extracts meaningful objects from input images, feature extraction component performs polygonal approximation of connected components, and the model matching component tells us which regions in the segmented image contain the model object by generating a real valued matching confidence indicating the degree of success. Reinforcement learning module then uses this confidence value as feedback to induce a mapping from images to segmentation strategies within each local region created from partitioning of the feature space. The goal is, therefore, to maximize the matching confidence by finding a set of segmentation algorithm parameters for the given recognition task.

There are good reasons for using reinforcement learning in our object recognition system. *First*, reinforcement learning requires knowing only the goodness of the system performance rather than the details of algorithms that produce the results. In the object recognition system, model matching confidence indirectly evaluates the performance of image segmentation and feature extraction processes. It is a natural choice to select matching confidence as a reinforcement signal. *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*, it is feasible to construct fast, parallel devices to implement this technique for real-time applications. Thus, it fits our goal nicely here. Likewise, local learning has the advantage of avoiding negative spatial cross-talk typically associated with global learning techniques, because mappings are constructed separately within each local region in the feature space. Furthermore, local reinforcement learning often gives rise to better improvement in recognition performance, as we shall see later. Note that the integration of local learning and reinforcement learning at the algorithmic level makes it possible to take advantage of some of the best features of each of the paradigms.

3.1. Related Work

Robot learning and landmark recognition are active areas of research (Connell and Mahadevan, 1993; Nasr and Bhanu, 1988; Trahanias et al., 1997; Zheng et al., 1994). The challenge is to extend operating conditions of a mobile robot. Adaptation and learning play an important role for achieving the robustness of algorithms. The work presented in this paper is most closely related to earlier work by the authors (Peng and Bhanu, 1996), in which they describe a reinforcement learning system that uses recognition output as feedback to guide the segmentation process. However, their method is global in that only a single mapping is induced over the entire input space. In addition, their system was evaluated only on a small number of images. In this work, we use simulated images with controlled statistics to show that our method can indeed learn correct segmentation strategies for recognition of objects. Further, we show that local learning, when combined with reinforcement learning, can outperform the global learning method using empirical results based on a large set of real images of traffic signs.

An adaptive approach to image segmentation is proposed by Bhanu and Lee (1994). Their system uses genetic and hybrid algorithms for learning segmentation parameters. However, the recognition algorithm is not part of the evaluation function for segmentation in their system. The genetic or hybrid algorithms simply search for a set of parameters that optimize a prespecified evaluation function (based on global and local segmentation evaluation) that may not best serve the overall goal of robust object recognition. Furthermore, their work assumes that the location of the object in the image is known for specific photointerpretation application. In our work, we do not make such an assumption. We use explicit geometric model of an object, represented by its polygonal approximation, to recognize it in the image.

4.2. *Connectionist Reinforcement Learning*

The particular class of reinforcement learning algorithms employed in each local region for our object recognition system is the connectionist REINFORCE algorithm (Williams, 1992), where units in such a network are *Bernoulli semilinear units*, in that the output of such a unit, i, is either 0 or 1, determined stochastically using the Bernoulli distribution with parameter $p_i = f(s_i)$, where f is the logistic function

$$f(s_i) = 1/(1 + \exp(-s_i)) \tag{4}$$

and $s_i = \sum_j w_{ij} x_j$ is the usual weighted summation of input values to that unit. For such a unit, p_i represents its probability of choosing 1 as its output value. The left graph in Fig. 4 depicts a connectionist reinforcement learning system and the right graph shows a Bernoulli semilinear unit in such a system.

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

$$\Delta w_{ij} = \alpha_{ij}(r - b_{ij}) \frac{\partial}{\partial w_{ij}} \ln(g_j) \tag{5}$$

where α_{ij} is a learning rate factor, r the immediate reinforcement, b_{ij} a baseline, and g_j is the density function for randomly generating output patterns. For the Bernoulli semilinear units used in this research, Eq. (5) reduces to

$$\Delta w_{ij} = \alpha(r - b_{ij})(y_i - p_i)x_j \tag{6}$$

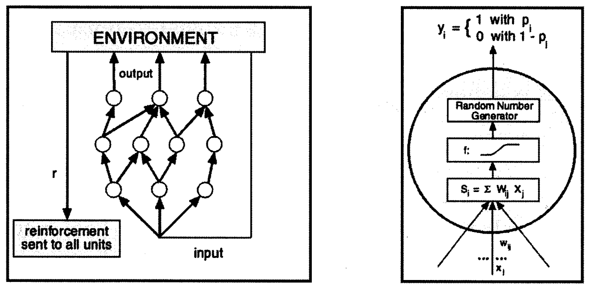

ENVIRONMENT
output
r
reinforcement sent to all units
input
$y_i = \{ \begin{smallmatrix} 1 \text{ with } p_i \\ 0 \text{ with } 1 - p_i \end{smallmatrix}$
Random Number Generator
f:
$s_i = \Sigma\, w_{ij}\, x_j$
w_{ij}
x_j

Figure 4. Left: Connectionist reinforcement learning system. Right: Bernoulli semilinear unit.

x_j is the input to each Bernoulli unit, y_i is the output of the ith Bernoulli unit, and p_i is an internal parameter to a Bernoulli random number generator.

It can be shown (Williams, 1992) that, regardless of how b_{ij} is computed[1], 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

$$E\{\Delta\mathbf{W} \mid \mathbf{W}\} = \alpha \nabla_{\mathbf{W}} E\{r \mid \mathbf{W}\} \tag{7}$$

where E denotes the expectation operator, $\mathbf{W}$ represents the weight matrix of 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 matching confidence increases.

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., the confidence level indicating the matching result, increment each weight w_{ij} by

$$\begin{aligned}\Delta w_{ij}(t) = {} & \alpha(r(t) - \bar{r}(t-1)) \\ & \times (y_i(t) - \bar{y}_i(t-1))x_j - \delta w_{ij}(t)\end{aligned} \tag{8}$$

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} (Williams, 1992). 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 Eq. (6), 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

$$\bar{r}(t) = \gamma \bar{r}(t-1) + (1-\gamma)r(t) \tag{9}$$

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 $\bar{r}$.

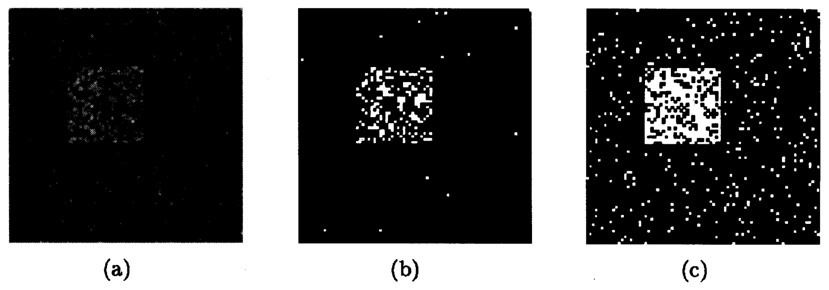


Figure 6. (a) A noisy simulated image ($SNR = 10$). (b) Learned segmentation. (c) Theoretical segmentation.

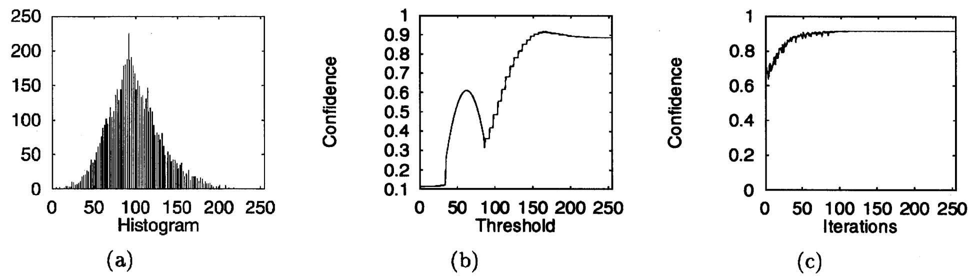


Figure 7. (a) Histogram of image in Fig. 6(a). (b) Recognition confidence as a function of threshold. (c) Recognition confidence (reward) received over time.

to an input image: For each pixel, if its value is greater than or equal to T, it is set to 1, otherwise to 0. It is the parameter T that has to be learned by the system. The "recognition algorithm" we use in this subsection computes a confidence value according to

$$r = \frac{n(I) - ((n(G) - n(G \cap R)) + (n(R) - n(R \cap G)))}{n(I)} \tag{13}$$

where G is the region of the ground truth (target), R the region of pixels having a value of 1 and I is the input image. Function $n(\cdot)$ returns the number of pixels of its argument. r is then used as reinforcement to drive learning. Note that Eq. (13) happens to be one (pixel classification) of possible criteria for image segmentation evaluation (Bhanu and Lee, 1994). Figure 7(b) plots r (Eq. (13)) against the threshold. A local maximum is added to make the problem more interesting. That is, when $T \in [35, 85]$, r follows a simple quadratic function of T.

A team of 8 Bernoulli units is used to represent the threshold (T_l) to be learned. It took on average (over 50 runs) less than 100 iterations ($\alpha = 0.2$) to learn a threshold (166) that achieves the *optimal* "recognition confidence" of 0.92 (Eq. (13)). Figure 6(b) shows the segmentation result and Fig. 7(c) the average r value received over time. As a comparison, Fig. 6(c) shows the segmentation using the theoretical threshold (136) that achieves a confidence value of 0.82, which is far from the optimal one. When the Gaussian random noise is removed, the learned threshold and the theoretical one are identical for the image in Fig. 6(a).

Experiment 2: To further evaluate our technique, 100 images were randomly generated as training data. The first 50 images have the following statistics: μ_b was set to 130, μ_t varied uniformly from 145 to 155, with $\sigma_b = \sigma_t = 2$, the size of the square target varied from 25×25 to 35×35 and the *SNR* from 10 to 15, while the statistics for the rest are: μ_b was set to 190, μ_t varied uniformly from 205 to 215, with $\sigma_b = \sigma_t = 3$, the target size varied from 40×40 to 50×50 and the *SNR* from 5 to 10. An additional set of 1000 images was generated randomly independently as testing data, whose statistics are as follows. For the first 500 images, μ_b was set to 130, μ_t varied from 140 to 160 uniformly, the size of the target from 20×20 to 40×40 and the *SNR* from 3 to 20, while for the remaining images, μ_b was set to 190, μ_t varied from 200 to 220, the target size from 35×35 to 55×55 and the *SNR* from 2 to 18. The best achievable average r value (experimentally determined) on the training data is 0.85, whereas this value is 0.83 on the testing data. The images are represented

Table 1. Sample ranges for selected *Phoenix* parameters.

Parameter	Sampling formula	Test range
Hsmooth: hsindex $\in [0:31]$	hsmooth $= 1 + 2 *$ 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 $= 9 + 2 *$ smindex	9–71
Height: htindex $\in [0:31]$	height $= 1 + 2 *$ htindex	1–63

highest peak. Table 1 shows sample ranges for each of these parameters. The resulting search space is about one million sample points. Each of the *Phoenix* parameters is represented using 5 bit binary code, with each bit represented by one Bernoulli unit. To represent 4 parameters, we need a total of 20 Bernoulli units.

The feature extraction consists of finding polygon approximation tokens for each of the regions obtained after image segmentation. The polygon approximation is obtained using a split and merge technique (Bhanu and Ming, 1987) that has a fixed set of parameters. Object recognition employs a cluster-structure matching algorithm (Bhanu and Ming,1987) that is based on the clustering of translational and rotational transformations between the object and the model for recognizing 2D and 3D objects. It outputs a real number indicating the confidence level of the matching process. This confidence level is then used as a reinforcement signal to drive learning. These algorithms were chosen simply because they are available in house.

5.2.1. Experimental Results on Real Data. The experiment described here consists of 500 images, some of which are shown in Fig. 1. These images are collected in late afternoon over several days (including a rainy day) using a Canon PowerShot 600 digital camera. They are taken in a variety of locations in Southern California. These images simulate an autonomous navigation scenario in which an autonomous vehicle must be able to recognize the stop sign. The size of the images is 78 by 104 pixels.

Eighty images are randomly selected as training data, and the rest (420) as testing data. A principal component analysis is carried out using the red color component. Red component of each image is projected onto the subspace spanned by the first 4 eigen vectors corresponding to four largest eigen values. These inputs are normalized to lie between 0 and 1.

Local Reinforcement Learning. The training data are first clustered using the K-means algorithm based on the eigen inputs. The K-means algorithm was repeatedly applied to the training data with varying K. The K value that attained the largest Calinski-Harabasz Index (Eq. (1)) was selected as the final cluster number (4 in this experiment). The resulting clusters contain 25, 26, 13, and 16 training images, respectively. Within each cluster, a network having 3 hidden Bernoulli units and 20 output Bernoulli units that encode the four *Phoenix* parameters was trained using the local reinforcement learning algorithm described in Fig. 5. Each hidden unit takes four eigen inputs and there are no connections from inputs to output units. Because of the independence of the output units, the effective number of weights in the network is 19 (4 (input weights) $\times$ 3 (hidden units) + 3 (hidden to output weights) + 4 (biases)). Figure 9 depicts such a network.

Global Reinforcement Learning. A global network, similar to the one shown in Fig. 9, is trained on the entire training data to construct a single mapping. The network has 8 hidden Bernoulli units, and 20 output units. The number of hidden units is determined experimentally that achieves the best performance among several trials. In comparison with local reinforcement learning, the effective number of free parameters in the global network is 49 (4 (input weights) $\times$ 8 (hidden units) + 8 (hidden to output weights) + 9 (biases)).

Simple Case-Based Learning. Instead of constructing a local mapping within each cluster, as is done in the local reinforcement learning method, the simple CBL method first learns, for each cluster, a set of segmentation parameters achieving the best performance

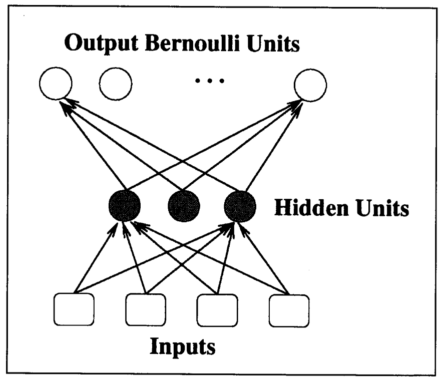


Figure 9. A connectionist reinforcement learning network.

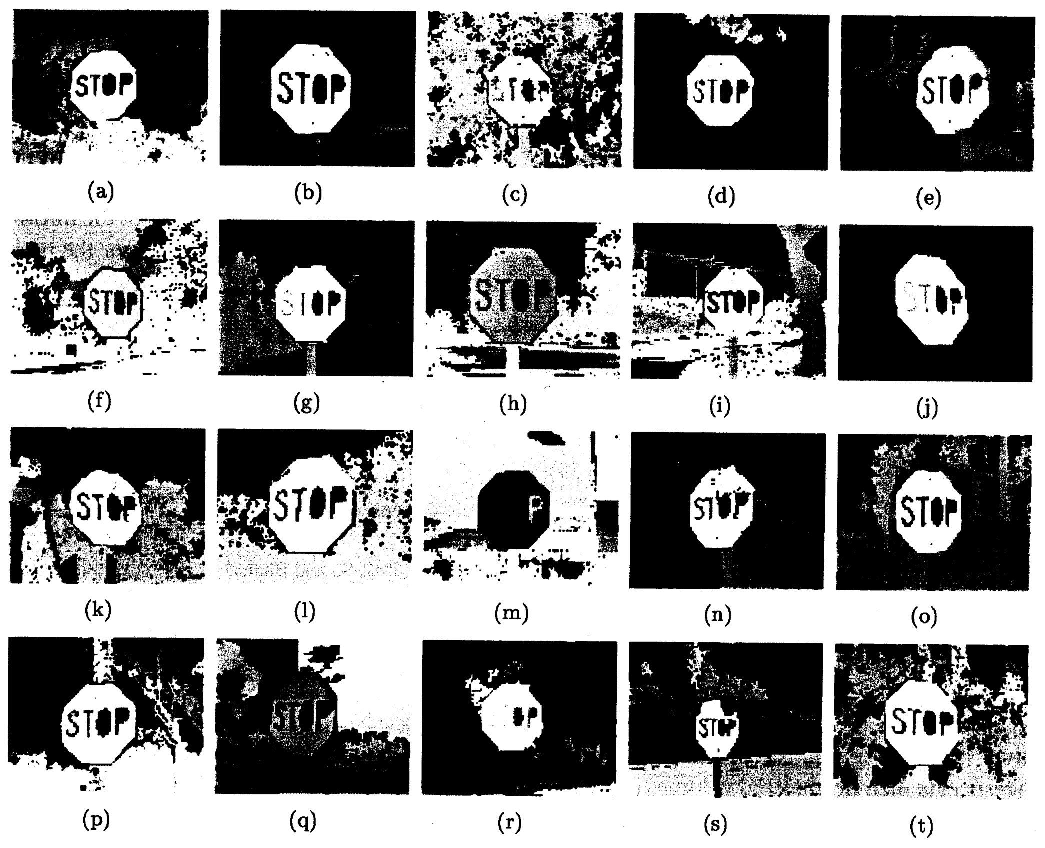


Figure 11. Segmentation of images shown in Fig. 1 using the *Phoenix* algorithm with learned parameters.

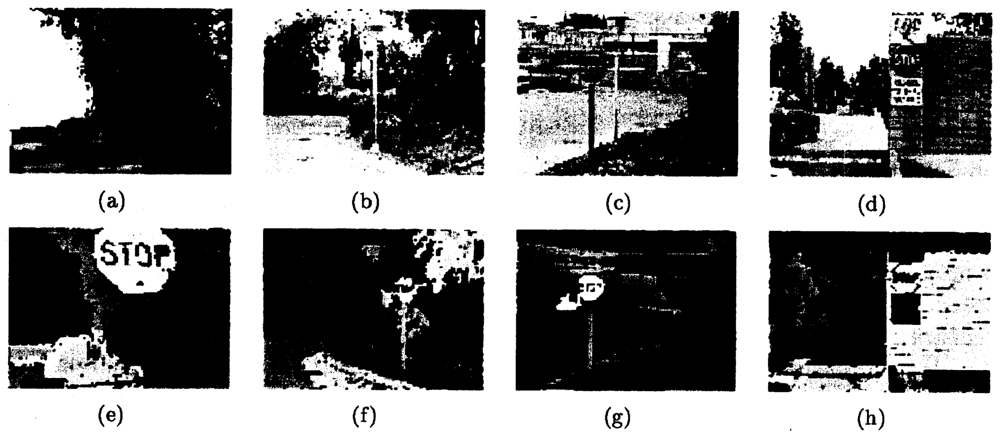


Figure 12. Row one: Sample images in which the stop sign does not take the center stage. Row two: Corresponding segmentations using the technique presented here.

take the center stage. These images represent far more difficult scenarios than those shown in Fig. 1. The second row of Fig. 12 shows the corresponding segmentations via local reinforcement learning introduced here. While the system succeeded in recognizing the stop signs in the images shown in Figs. 12(a) and (d), it failed to do so for the images shown in Figs. 12(b) and (c). These results are representative of the capabilities of the segmentation/recognition algorithms employed here.

Le Cun, Y., Bottou, L., and Bengio, Y. 1997. Reading checks with graph transformer networks. In *Proc. of Int. Conf. on Acoustics, Speech, and Signal Processing* (ICASSP 97), Vol. 1, pp. 151–154.

Marroquin, J.L. and Girosi, F. 1993. Some extensions of the K-means algorithm for image segmentation and pattern classification. A.I. Memo No. 1390, MIT AI Lab.

Milligan, G.W. and Cooper, M.C. 1985. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50:159–179.

Nasr, H. and Bhanu, B. 1988. Landmark recognition for autonomous mobile robots. *IEEE Int. Conf. on Robotics and Automation*, 1218–1223.

Olson, C. 1997. Mobile robot self-location by iconic matching of range maps. In *Proc. of the 8th Int. Conf. on Advanced Robotics*, pp. 447–452.

Peng, J. and Bhanu, B. 1996. Closed-loop object recognition using reinforcement learning. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, CA, pp. 538–543.

Pomerleau, D.A. 1993. *Neural Network Perception for Mobile Robot Guidance*. Kluwer Academic Publishers: Boston, MA.

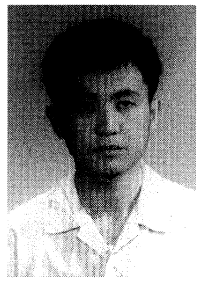
Ram, A. and Santamaria, J.C. 1998. Continuous case-based reasoning. *Artificial Intelligence*.

Trahanias, P.E., Velissaris, S., and Garavelos, T. 1997. Visual landmark extraction and recognition for autonomous robot navigation. *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Grenoble, France.

Williams, R.J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8: 229–256.

Williams, R.J. and Peng, J. 1991. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3).

Zheng, Y.J., Ritter, W., and Janssen, R. 1994. An adaptive system for traffic sign recognition. *Intelligent Vehicles Symposium*, Paris.



Jing Peng received the M.A. degree in Computer Science from Brandeis University and the Ph.D. degree in Computer Science from Northeastern University, Boston, Massachusetts. He also received the B.S. degree in Computer Science from Beijing Institute of Aeronautics and Astronautics, Beijing, China.

Recently, he has been a research scientist with Center for Research in Intelligent Systems at the University of California at Riverside. Dr. Peng's research interests include maching learning, computer vision, image databases, data mining, and learning and vision applications.

Bir Bhanu received the S.M. and E.E. degree in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology, Cambridge, and the Ph.D. degree in Electrical Engineering from the Image Processing Institute, University of Southern California, Los Angeles. Currently he is a Professor of Electrical Engineering and Computer Science and Director of Center for Research in Intelligent Systems at the University of California, Riverside. Previously, he was a Senior Honeywell Fellow at Honeywell Systems and Research Center, Minneapolis, MN. He has been the principal investigator of various programs from DARPA, NASA, NSF, AFOSR, ARO and other agencies and industries in the areas of image understanding, learning and vision, pattern recognition, navigation, target recognition, image databases, and machine vision applications. He is the co-author of books on "Computational Learning for Adaptive Computer Vision," (Plenum, Forthcoming), "Genetic Learning for Adaptive Image Segmentation" (Kluwer, 1994), and "Qualitative Motion Understanding" (Kluwer, 1992). He holds 10 U.S. and international patents and over 150 reviewed technical publications in the areas of his interest. Dr. Bhanu is a Fellow of the IEEE and the AAAS. He is a member of ACM, AAAI, Sigma Xi, Pattern Recognition Society, and SPIE.