Delayed Reinforcement Learning for Adaptive Image Segmentation and Feature Extraction

Jing Peng and Bir Bhanu

***Abstract*— Object recognition is a multilevel process requiring a sequence of algorithms at low, intermediate, and high levels. Generally, such systems are open loop with no feedback between levels and assuring their robustness is a key challenge in computer vision and pattern recognition research. A robust closed-loop system based on "delayed" reinforcement learning is introduced in this paper. The parameters of a multilevel system employed for model-based object recognition are learned. The method improves recognition results over time by using the output at the highest level as feedback for the learning system. It has been experimentally validated by learning the parameters of image segmentation and feature extraction and thereby recognizing 2-D objects. The approach systematically controls feedback in a multilevel vision system and shows promise in approaching a long-standing problem in the field of computer vision and pattern recognition.**

***Index Terms*— Adaptive feature extraction, adaptive image segmentation, learning for multilevel vision, learning in computer vision, model-based recognition, multiscenario object recognition, recognition feedback.**

I. Introduction

Most vision systems use a sequence of algorithms that operate at various stages of abstraction to perform a given task, such as object recognition. In earlier work that combines learning and vision [1], the inherent multistage nature of vision systems has not been addressed adequately. In this paper, an approach that takes the output of the final stage and uses it as a feedback in a reinforcement learning framework to influence the performance of the lower stages of vision algorithms is presented. The overall system performance is improved over time with this method.

Fig. 1 illustrates the typical approach for model-based object recognition, which is often unidirectional and without feedback. While different stages keep getting better algorithms, there has been little improvement in object recognition as a whole. This is due to the lack of a feedback theory of multistage object recognition. For those systems that do use feedback [2], [3], there is generally no learning from experience to improve future recognition performance, which is the subject of this correspondence.

The segmentation and feature extraction modules shown in Fig. 1 use default parameters that are usually obtained by the system designer by following a trial-and-error approach. However, the designer cannot anticipate all possible inputs to the algorithms; the content of the three-dimensional (3-D) scene and the environmental conditions are not known *a priori*. The simultaneous adjustment of even a few system parameters is time consuming and difficult and has yet to be solved satisfactorily for multistage systems. As a result, the approach shown in Fig. 1 is inadequate for real-world applications. The key to the performance improvement of a multistage object recognition system over time is the automatic adjustment of parameters of various algorithms to optimize various image processing functions for detected objects so that it is optimal and meets task- or object-specific requirements.

Manuscript received June 17, 1996; revised April 19, 1997 and December 7, 1997. This work was supported by DARPA/AFOSR Grants F49620-97-1-0184 and F49620-95-1-0424. The contents of the information do not necessarily reflect the position or the policy of the United States Government.

The authors are with the Visualization and Intelligent Systems Laboratory, College of Engineering, University of California, Riverside, CA 92521 USA (e-mail: bhanu@vislab.ucr.edu; jp@vislab.ucr.edu).

Publisher Item Identifier S 1094-6977(98)03897-8.

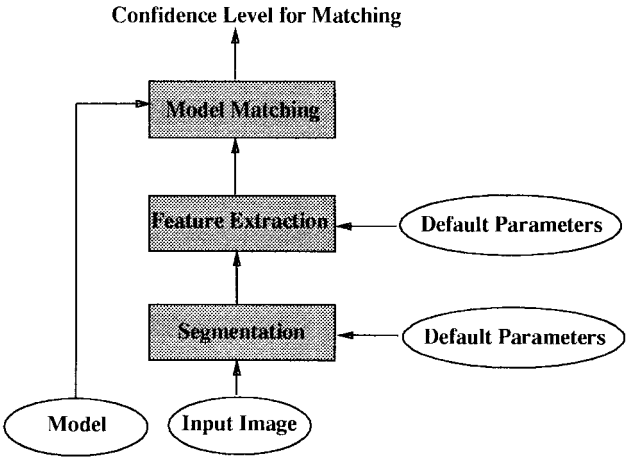


Fig. 1. Conventional system for object recognition.

II. Our Approach

If it is assumed that the model matching produces a confidence measure indicating the closeness of the selected features to the model, it is natural to use this confidence as feedback to influence the system's performance for segmentation and feature extraction. The broad goal of such a scheme is to try to find, for any given image, a set of parameters for image segmentation and feature extraction in ways that minimize recognition errors. Applying a reinforcement learning algorithm to learn parameters can be viewed as a means of doing just this when the matching confidence is used as a reinforcement. Fig. 2 shows a closed-loop reinforcement learning-based system to achieve this goal.

In contrast, it would be difficult, if not impossible, for a conventional search method to accomplish the same task. Simply, there are no well-defined evaluation functions at each of the stages for a method to search for. Furthermore, if a method uses the confidence of model matching for evaluation, it is not clear how the process should proceed in a systematic way. Finally, at each stage, any such method will have to delay its decision as to where to search next until the confidence of model matching becomes available. However, this need not be the case for the approach presented in this paper. From a computational standpoint, therefore, our approach is more attractive since the computation can be distributed over time more evenly, which will reduce overall demands on the memory and speed.

The original contribution of this research paper is to provide an incremental method based on "delayed" reinforcement learning for inducing a general mapping from images to parameter settings in a multistage, model-based, object recognition system. A theoretical model is provided and its efficacy is validated using real-world data.

III. Reinforcement Learning

Reinforcement learning studies computational approaches to learning from rewards and punishments (called reinforcements). It is about learning optimal control in Markov decision problems. In this paper, reinforcement corresponds to the confidence measure generated by the model matching (see Fig. 2). Several factors affect reinforcement learning, the most important of which is the timing of reinforcement.

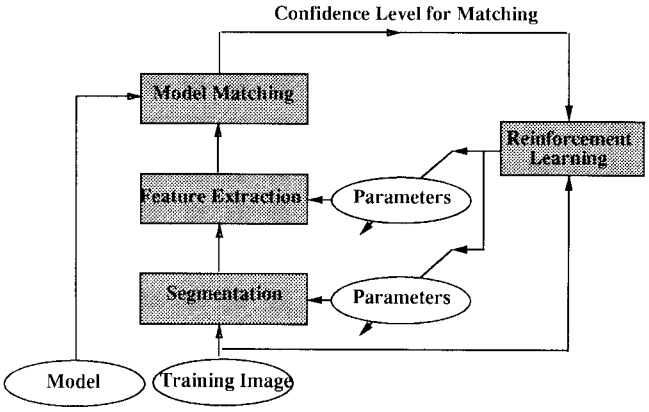


Fig. 2. Reinforcement learning-based multistage system for object recognition.

Reinforcement becomes available after each action in simple tasks. In most complex tasks, however, reinforcement is often temporally delayed. For example, in the object recognition system, the goodness of segmentation and feature extraction may not be reliably known until model matching has been performed.

One set of effective methods for delayed reinforcement learning is given by the theory of dynamic programming. Given a Markov decision problem, these methods involve first determining the "optimal action-value function," the Q function [10], that assigns to each state-action pair a value measuring the average total (discounted) reward obtained when a particular action is taken in the given state and the optimal policy is followed thereafter. That is, using the notation that x denotes the current state, a denotes the current action, r denotes the resulting immediate reward, and y denotes the resulting next state from taking a in x, then

$$Q(x,a) = R(x,a) + \gamma \sum_y P_{xy}(a) V(y) \tag{1}$$

where $R(x,a) = E\{r|x,a\}$ with E denoting the expectation operator, $V(x) = \max_a \ Q(x,a), P_{xy}(a)$ is the probability of making a state transition from x to y as a result of applying action a, and $\gamma \in [0,1)$ is a discount factor. Once the Q function is known, it is straightforward to determine the optimal policy. For any state x, the optimal action is simply $\arg \ \max_a \ Q(x,a)$. Note that both x and a can be vectors.

The particular method employed in this work for learning the Q function is the $Q(\lambda)$ algorithm [9], where $\lambda \in [0,1]$. This algorithm is an example of the "temporal difference" (TD) method of Sutton [8]. Like Q learning [10], $Q(\lambda)$ learning works by maintaining an estimate $\hat{Q}$ of the Q function and updating it so that (1) comes to be more nearly satisfied for each state-action pair encountered. More specifically, in Q learning, the action value estimate $\hat{Q}$ for state-action pair (x_t, a_t) is regressed toward the estimate $\hat{V}$ for x_{t+1}, where $\hat{V}(x) = \max_a \ \hat{Q}(x,a)$ is used as the target for adjusting $\hat{Q}$ at (x_t, a_t). In $Q(\lambda)$ learning, on the other hand, the estimate $\hat{Q}(x_t, a_t)$ is regressed not just toward the estimate $\hat{V}(x_{t+1})$, but to a weighted mixture of the estimates $\hat{V}(x_{t+1}), \hat{V}(x_{t+2}), \cdots, \hat{V}(x_{t+k})$, etc., up to and including the final outcome, where the weightings are proportional to λ^{k-1}. Let $e'_t = r_t + \gamma \hat{V}(x_{t+1}) - \hat{Q}(x_t, a_t)$ and $e_t = r_t + \gamma \hat{V}(x_{t+1}) - \hat{V}(x_t)$. Then the update rule for $Q(\lambda)$ learning can be written as

$$\begin{aligned} \hat{Q}(x_t, a_t) \leftarrow \hat{Q}(x_t, a_t) \\ + \alpha\{e'_t + \lambda\gamma e_{t+1} + \lambda^2\gamma^2 e_{t+2} + \lambda^3\gamma^3 e_{t+3} + \cdots\} \end{aligned} \tag{2}$$

where α is the learning rate. Thus, it can be seen from the above equation that λ controls the proportion in which future estimates are combined into overall targets. By shifting the estimate for $Q(x,a)$ toward a weighted mixture of downstream targets, $Q(\lambda)$ learning not only achieves better computational efficiency, but it also enables, under appropriate conditions, the elimination of the effect of initial bias. When $\lambda = 0, Q(\lambda)$ learning reduces to simple Q learning. The typical choice for the λ value is somewhere between zero and one, but close to zero. More details about the $Q(\lambda)$ algorithm are given in [9].

To implement the $Q(\lambda)$ learning algorithm, a memory mechanism, called the *eligibility trace* [11], is used. Every time a state is encountered, it starts a trace that marks the state as eligible for learning and which reduces gradually over time. If $Tr(x,a)$ denotes the eligibility trace of state-action pair (x,a), the $Q(\lambda)$ learning algorithm can be described in Fig. 3. The eligibility trace assigns a value to each experienced state-action pair with more recent ones having higher values.

IV. Reinforcement Learning for Object Recognition

In the multistage system for model-based object recognition described in Fig. 2, there are unknown parameters for both the segmentation and feature extraction modules. The segmentation module is based on the *Phoenix* algorithm [5], [6]. *Phoenix* was chosen because it is a well-known method for the segmentation of color images with a number of adjustable parameters. *Phoenix* uses region splitting based on histograms of color features and is critically dependent on system parameters *Hsmooth* and *Maxmin*.

Hsmooth is the width of the histogram smoothing window, where smoothing is performed with a uniformly weighted moving average. *Maxmin* defines the peak-to-valley height ratio threshold. Any interval whose peak height to higher shoulder ratio is less than this threshold is merged with the neighbor on the side of the higher shoulder. The algorithm searches for a combination of these parameters that will give rise to a segmentation from which the best recognition can be achieved. The ranges for each of the two parameters are $hsmooth = 1 + 2 * \ hsindex$ and $maxmin = \exp\{\ln(100) + 0.05 * \ mmindex\} + 0.5$, where $hsindex, mmindex \in [0{:}31]$. The resulting search space is 1024 sample points.

The feature extraction module [7] finds polygon approximation for borders of each of the regions obtained after image segmentation. It is based on a split-and-merge technique that first computes the locations of curvature maxima. The estimates of maximum curvature depend critically on neighborhood bound parameters, called $M1$ and $M2$ [12]. $M1$ and $M2$ are the nearest and farthest border pixels, respectively, from the pixel at which the curvature is estimated. For the purpose of this correspondence only, $M2$ is subject to adaptation. It takes values between six and 26 in the experiments reported here.

Our object recognition process employs a cluster-structure matching algorithm [7] that is based on the clustering of translational and rotational transformations between the object and the model for recognizing two-dimensional (2-D) and 3-D objects. The algorithm takes as input two sets of tokens (polygonal approximation of the boundary), 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. It computes, based on the number of model segments that match the segments in the data, a real number that indicates the confidence level of the matching process. This confidence level is then used as a reinforcement signal to drive the algorithm.

Note that, in general, there may be no model object or there can be multiple instances of one model object or several different model objects in the image. If the goal is to recognize multiple objects, it might be preferable to use an average of the confidence levels

1. $\hat{Q}(x,a) = 0$ and $Tr(x,a) = 0$ for all x and a, and $t = 0$
2. Do Forever:
   (a) $x_t \leftarrow$ the current state
   (b) Choose an action a_t that maximizes $\hat{Q}(x_t, a_t)$ over all a_t
   (c) Carry out action a_t in the world. Let the short term reward be r_t, and the new state be x_{t+1}
   (d) $e'_t = r_t + \gamma\hat{V}(x_{t+1}) - \hat{Q}(x_t, a_t)$, $e_t = r_t + \gamma\hat{V}(x_{t+1}) - \hat{V}(x_t)$
   (e) For each state-action pair (x,a) do
   - $Tr(x,a) = \gamma\lambda Tr(x,a)$
   - $\hat{Q}(x,a) = \hat{Q}(x,a) + \alpha Tr(x,a)e_t$

   (f) $\hat{Q}(x_t, a_t) = \hat{Q}(x_t, a_t) + \alpha e'_t$
   (g) $Tr(x_t, a_t) = Tr(x_t, a_t) + 1$
   (h) $t = t + 1$

Fig. 3. $Q(\lambda)$-learning algorithm.

1. Initialization: $\hat{Q}(x, \bar{p}) \leftarrow 0$ for all $x, \bar{p}$, where x is either an image or a segmented image and $\bar{p}$ is either an instance of segmentation parameters $\bar{a}$ or feature extraction parameters $\bar{b}$.
2. LOOP:
   - For each image i in the training set do
     (a) Segment image i with segmentation parameters $\bar{a} = (a_1, a_2, \cdots, a_n)$ recommended by ϵ-greedy policy; i_s is the resulting segmented image.
     (b) Update: $\hat{Q}(i,\bar{a}) \leftarrow \hat{Q}(i,\bar{a}) + \alpha\{\gamma\hat{V}(i_s - \hat{Q}(i,\bar{a})\}$ where $\hat{V}(i_s) = \max_{\bar{b}} \hat{Q}(i_s, \bar{b})$.
     (c) Perform feature extraction with feature extraction parameters $\bar{b} = (b_1, b_2, \cdots, b_n)$ recommended by ϵ-greedy policy from the segmented image i_s.
     (d) Compute the matching of each connected component (which is close to the size of the current model) against stored model and return the highest confidence level r
     (e) Update: $\hat{Q}(i_s, \bar{b}) \leftarrow \hat{Q}(i_s, \bar{b}) + \alpha\{r - \hat{Q}(i_s, \bar{b})\}$ and $\hat{Q}(i,\bar{a}) \leftarrow \hat{Q}(i,\bar{a}) + \alpha(\lambda\gamma)\{r - \hat{V}(i_s)\}$
3. UNTIL terminating condition

Fig. 4. Main steps of the delayed reinforcement learning algorithm for parameter adjustment for segmentation and feature extraction.

resulting from each model matching. The desired result is that parameters are chosen more judiciously so as to optimize the average confidence measure that rewards parameters accommodating differences in characteristics between regions in the image. There are other possibilities as well. For example, we might design a method whose operating conditions are amended to these differences to achieve optimal performance for segmentation, feature extraction, and model matching. Such a scheme would localize its computation by ways of local segmentation to meet each individual requirement. We are currently pursuing these ideas [13]. In this paper, however, we are concerned with simple situations, in which only one model object is present in the image. Thus, the maximum confidence level suffices.

The objective of the system is to autonomously find a set of segmentation and feature extraction parameters that achieves the maximum matching confidence for a given input image. Our model-based recognition system is a multistage decision process in which the parameters *Hsmooth* and *Maxmin* are at the first stage of the process and the parameter $M2$ is at the second stage. The goodness of a particular decision, such as selecting a combination of the segmentation parameters, is not known until the model matching has been performed. To achieve the objective, therefore, the $Q(\lambda)$ learning algorithm with the confidence of model matching as reinforcement is used to adjust the parameters at both the first and second stage.

Let i be an input image to the segmentation module, $\overline{a}$ be an instance of segmentation parameters, and $\overline{b}$ be an instance of feature extraction parameters. (Note that in this paper, $\overline{b}$ is simply a scaler.) Also, let i_s be the segmented image resulting from applying *Phoenix* with $\overline{a}$ as its parameter values to image i. Then, according to the $Q(\lambda)$-learning algorithm $Q(i,\overline{a})$ measures how good the instance $\overline{a}$ is when *Phoenix* applied to image i. Likewise, $Q(i_s,\overline{b})$ measures the quality of extracted features when the feature extraction algorithm with $\overline{b}$ as its parameter values is applied to the segmented image i_s. When the $Q(\lambda)$ learning algorithm is applied to the parameters the value of $Q(i,\overline{a})$ will be corrected to look more like the value of the segmented image$V(i_s) = \max_{\overline{b}} Q(i_s,\overline{b})$, which will in turn be estimated according to the matching confidence.

Fig. 4 shows the main steps of the algorithm described, where ϵ-greedy policy is a policy that selects random actions for ϵ fraction of time. Although there is no strong theoretical foundation, this exploration strategy works well in practice [9]. The algorithm terminates when either the number of iterations has exceeded a prespecified value or the recognition confidence level has reached a given threshold.

(a) (b)

Fig. 5. (a) Sample outdoor color image (Frame 1 of a 20 frame sequence) and (b) polygonal model of the car.

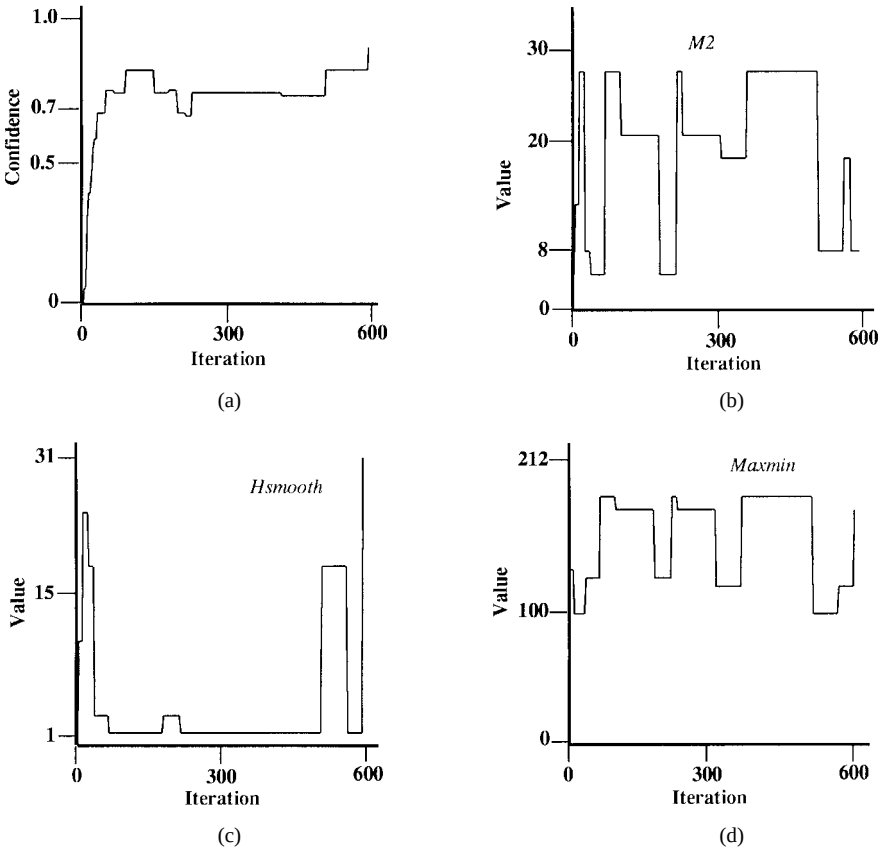


Fig. 6. Experimental results (training) for the image in Fig. 5. (a) Matching confidence level, (b) parameter $M2$, (c) parameter *Hsmooth*, and (d) parameter *Maxmin*.

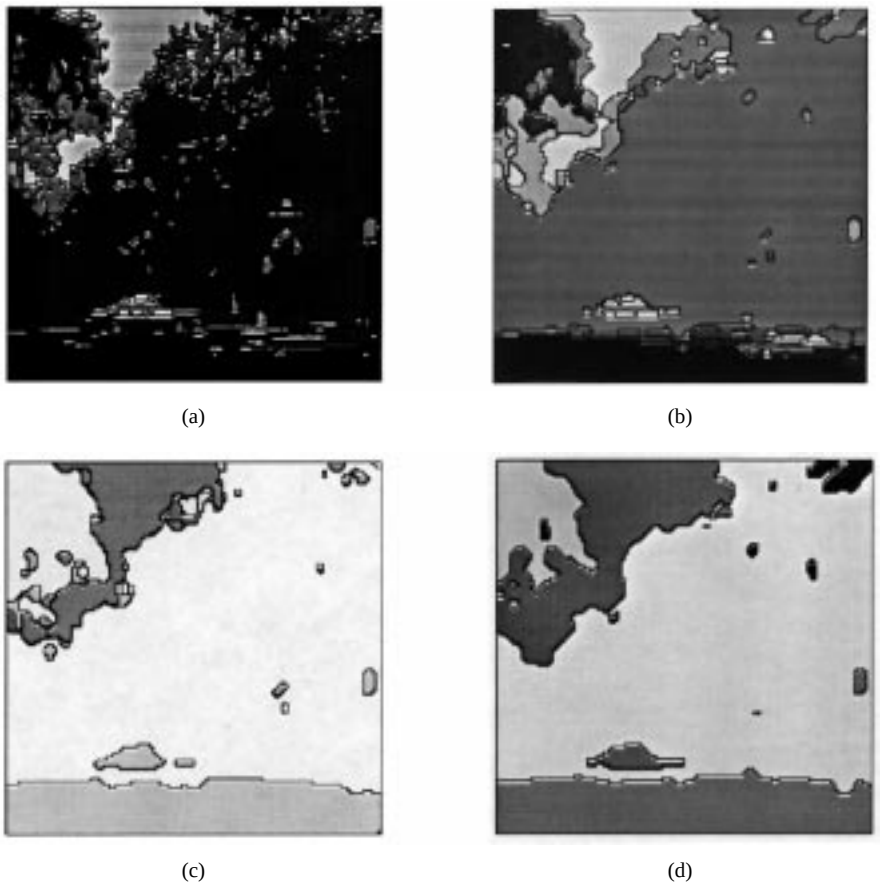
V. Experimental Validation

There are several representation schemes for the Q function in the reinforcement learning paradigm. Since the goal here is to demonstrate the effect of learning for multistage recognition, we have used a simple lookup-table-based representation.

The two dimensions of the lookup table are the following: 1) input or segmented (feature-extracted) image and 2) action represented by a particular combination of system parameters. The "activity" trace Tr is similarly indexed. All table entries are initialized to zero, which means that each combination of the parameter values can be selected for evaluation with equal probability in the beginning. The focus of the experiments is to demonstrate the feasibility of using learning for multistage recognition. Also, $\gamma = 0.95, \lambda = 0.3$, and $\epsilon = 0.1$ for all experiments reported here.

Fig. 5(a) shows a sample of a sequence of outdoor color images (120×120) obtained under varying environmental conditions. These images were collected approximately every 15 min over a $\sim$2 and 1/2-h period [4]. The images exhibit varying shadow and reflection on the car as the position of the sun changed and clouds came in and out the field-of-view of the camera that had auto iris adjustment turned on. The overall goal is to recognize the car in the image. It should be noted that, although the image is in color, for publication purposes, it is being shown in grayscale.

Fig. 5(b) shows the 2-D model of the car located in Fig. 5(a). The dark squares in Fig. 5(b) correspond to labels of the vertices in the



(a) (b)

(c) (d)

Fig. 7. Improvement of the segmentation over time. (a) Initial segmentation, (b) segmentation at time step 200, (c) segmentation at time step 400, and (d) segmentation at time step 600.

(a) (b)

Fig. 8. Polygonal approximation of the car. (a) Default parameter value $(M2 = 6)$ and (b) learned parameter value $(M2 = 8)$.

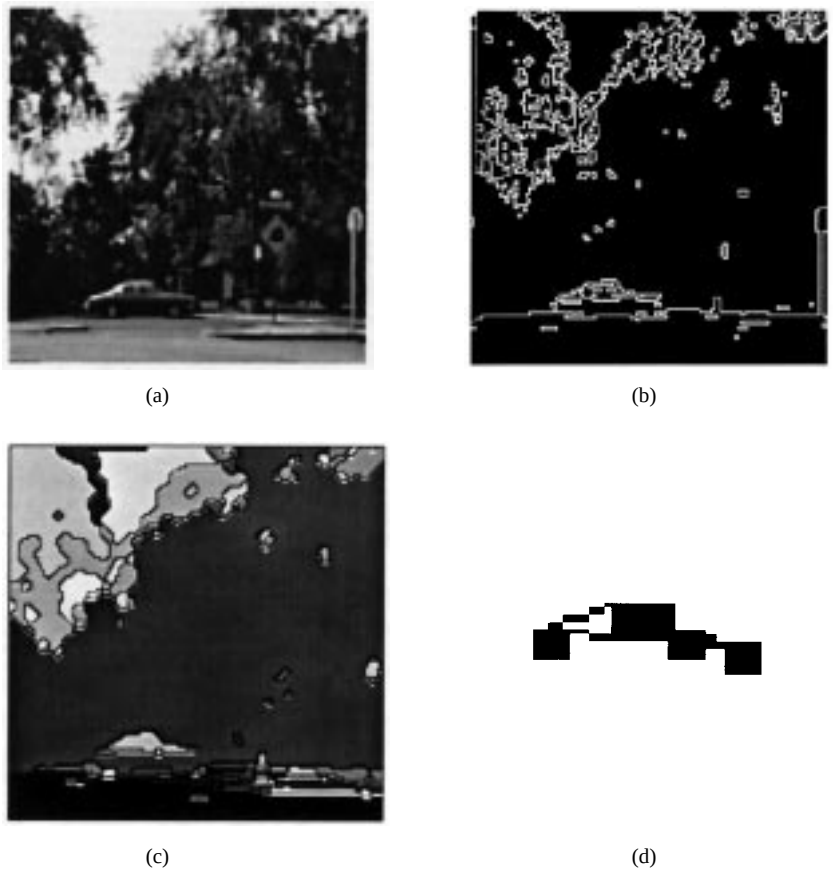
polygonal approximation of the car. The car is extracted manually in an interactive session from the first frame in the sequence, and its polygonal approximation [Fig. 5(b)] is used as the model in the cluster-structure matching algorithm.

Fig. 6(a) shows how the confidence, averaged over five runs, changes over time for the image shown in Fig. 5(a). It should be noted that over time the confidence shown in Fig. 6(a) increases. At the end of the training phase, the confidence of the match is over 0.9 on a scale that varies between zero and one. For acceptable recognition, the confidence of matching has to be greater than 0.75 in the experiments reported here.

Fig. 6(b)–(d) show how the $M2$, *Hsmooth*, and *Maxmin* change over time for a particular run, respectively. It can be seen clearly that the learned values of $M2$, *Hsmooth*, and *Maxmin* are considerably different from their starting random values.

To illustrate the results further, Fig. 7 shows how the segmentation of the image improves over time during training. Fig. 7(a) depicts the segmentation from initial random parameter settings. Fig. 7(b) and (c) depict the segmentation after 1/3 and 2/3 of total time (600 iterations) for training has elapsed, respectively. Fig. 7(d) depicts the segmentation at the end of the training phase. It can be seen that the results improve considerably. While Fig. 8(a) shows the extracted features (polygonal approximation of the car) using the default $M2$ parameter value, which is six in the experiment, Fig. 8(b) shows the same feature using the learned parameter value, which is eight, that results in high matching confidence.

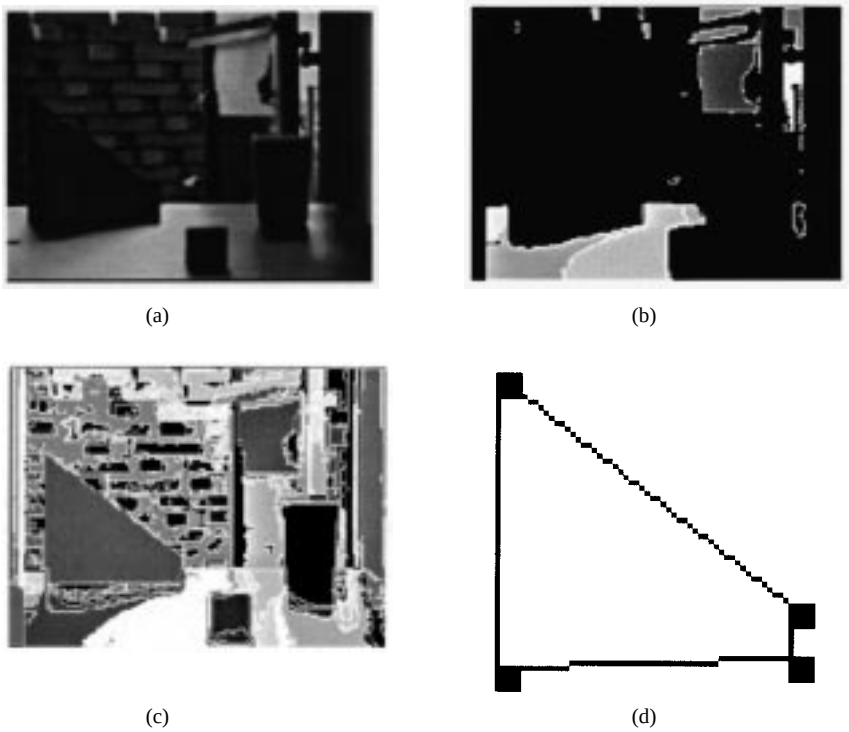
Fig. 9(a) shows another sample frame in the image sequence in which the car of Fig. 5(b) must be identified. It can be seen that the lighting conditions in this outdoor image is significantly different from the one shown in Fig. 5(a). The image is taken at a different time from Fig. 5(a). Fig. 9(b) shows the segmentation using default *Phoenix* parameter values [5], where $Hsmooth = 9$ and $Maxmin = 160$. It should be noted that, when default parameters are used, the car is broken into many small blobs from which



(a) (b)

(c) (d)

Fig. 9. Experimental results. (a) Third frame of the image sequence, (b) segmentation with default parameters, (c) segmentation with learned parameters, and (d) final polygonal approximation for the car.



(a) (b)

(c) (d)

Fig. 10. Experimental results on an indoor image. (a) The color image, (b) segmentation with default parameters, (c) segmentation with learned parameters, and (d) final polygonal approximation for the wedge.

polygonal approximation of the car cannot be accurately obtained. Fig. 9(c) and (d) show the segmentation obtained by using the parameters obtained from learning and the polygonal approximation of the car obtained from the segmented image, respectively. The confidence of model matching is 0.88.

Fig. 10(a) shows an indoor color image. The large triangular shaped object (wedge) is the object of interest for recognition. Fig. 10(b) shows the segmentation result using default parameters from which the appropriate polygonal approximation of the triangular object cannot be obtained. Fig. 10(c) and (d) show the segmentation and the polygonal approximation of the triangular object with learned parameters, respectively. In this case, the confidence of model matching is 0.90. It took about a few hundred iterations to obtain the result shown in Fig. 10(c) and (d).

VI. Conclusions

The learning-based object recognition system presented in this paper uses the recognition component as part of the evaluation functions for controlling feedback and learning parameters for image segmentation and feature extraction in a systematic way. In the experiments, we have used a lookup table to represent the Q function. However, lookup table representation may not be adequate in large problems since the search space is often too large to allocate entire memory. A possible solution to this problem is to first classify input images into representative clusters [4], for example, using algorithms, such as the K-Means algorithm, and then allocate memory only to the centers of these clusters. For a given image, generalization can be made by searching for the nearest cluster center. In general, however, compact function representation schemes that can generalize across states must be sought.

One additional benefit of the approach, due to the stochastic nature of reinforcement learning, is that the system is capable of exploring a significant portion of the search space, resulting in the discovery of good solutions, which, in general, cannot be achieved by any deterministic or simple supervised learning methods. Such an ability is prerequisite for any system to achieve high performance in real-world conditions. Although we have used a three-stage system to demonstrate a general approach to multistage, model-based, object recognition in a reinforcement learning paradigm, it can certainly be extended to systems having any number of stages. There is no doubt, however, that computational complexity will increase as the number of stages increases.

Finally, if vision systems could be designed in one stage as a single black box, the "simple" reinforcement learning paradigm would have sufficed. Earlier work on one-stage systems used a team of stochastic semilinear units for learning image segmentation parameters [1]. In reality, however, vision systems have multiple stages for real-world tasks with parameters that need to be adjusted at each stage. The delayed reinforcement-learning-based approach presented here shows promise in providing a potential solution to the problem of object recognition in multistage systems.

References

[1] J. Peng and B. Bhanu, "Closed-loop object recognition using reinforcement learning," in *Proc. DARPA Image Understanding Workshop*, Monterey, CA, Nov. 1994, pp. 777–780.

[2] J. R. Ullmann, "Edge replacement in the recognition of occluded objects," *Pattern Recognit.*, vol. 26, no. 12, pp. 1771–1784, 1993.

[3] S. Das, B. Bhanu, and C. C. Ho, "Generic object recognition using multiple representations," *Image Vision Comput.*, vol. 14, pp. 323–338, 1996.

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

[5] K. Laws, *The PHOENIX Image Segmentation System: Description and Evaluation*, SRI Int., Tech. Rep. 289, Dec. 1982.

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

[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] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Mach. Learn.*, vol. 3, pp. 9–44, 1988.

[9] J. Peng and R. J. Williams, "Incremental multi-step Q-learning," in *Proc. 11th Int. Conf. Machine Learning*, New Brunswick, NJ, 1994, pp. 226–232.

[10] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, U.K., 1989.

[11] A. H. Klopf, *The Hedonistic Neuron: A Theory of Memory, Learning, and Intelligence*. Washington, DC: Hemisphere/Harper & Row, 1982.

[12] T. Pavlidis, *Structural Pattern Recognition*. Berlin, Germany: Springer-Verlag, 1977.

[13] B. Bhanu, X. Bao, and J. Peng, "Reinforcement learning integrated image segmentation and object recognition," in *Proc. DARPA Image Understanding Workshop*, New Orleans, LA, 1997, pp. 1145–1154.

Minimum Entropy and Information Measure

Lin Yuan and H. K. Kesavan

***Abstract*—A previous work of Kapur *et al.* [5] introduced the MinMax information measure, which is based on both maximum and minimum entropy. The major obstacle for using this measure, in practice, is the difficulty in finding the minimum entropy. An analytical expression has already been developed for calculating the minimum entropy when only variance is specified. Here, an analytical formula is obtained for calculating the minimum entropy when only mean is specified, and numerical examples are given for illustration.**

Index Terms— MinMax measure, minimum entropy, moment constraints, probabilistic system.

I. Introduction

We briefly review the MinMax measure proposed in [5] and develop an analytical formula for it when only the mean is specified. This measure gives an unambiguous result when the knowledge about a probabilistic system is in the form of moment constraints.

Before we present the method of finding a minimum entropy distribution, we discuss the importance of this distribution and the significance of the value of the minimum entropy for information-theoretic systems.

Manuscript received July 26, 1996; revised April 20, 1997 and November 8, 1997.

L. Yuan is with the Department of Statistics, University of Waterloo, Waterloo, Ont., N2L 3G1 Canada (e-mail: lyuan@jeeves.uwaterloo.ca).

H. K. Kesavan is with the Department of Systems Design Engineering, University of Waterloo, Waterloo, Ont., N2l 3G1 Canada.

Publisher Item Identifier S 1094-6977(98)03896-6.