Delayed Reinforcement Learning for Closed-Loop Object Recognition*

Jing Peng and **Bir Bhanu**
College of Engineering
University of California
Riverside, CA 92521
{jp,bhanu}@vislab.ucr.edu

Abstract

Object recognition is a multi-level 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 research. A robust closed-loop system based on "delayed" reinforcement learning is introduced in this paper. The parameters of a multi-level 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 multi-level vision system and provides a potential solution to a long-standing problem in the field of computer vision.

1 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 multi-stage 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 proposed. The overall system performance is improved over time with this method.

The typical approach for model-based object recognition [2] segments the image at the first stage, then extracts and selects appropriate features from the segmented image at the second stage, and finally matches the selected features to the stored model. The segmentation and feature extraction modules use default parameters. These parameters are usually obtained by the system designer by following a trial and error method. However, the designer cannot anticipate all possible inputs to the algorithms; the content of the three-dimensional 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 multi-stage systems. As a result, this approach is inadequate for real-world applications because default parameters of segmentation and feature extraction often lead to poor recognition performance.

If it is assumed that the model matching produces a confidence measure indicating the closeness of the selected features to the model, then 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 the parameters can be viewed as a means of doing just this when the matching confidence is used as reinforcement. Figure 1 shows our closed-loop reinforcement learning-based system to achieve this goal.

An additional strength of our system is that it is capable of providing, to the extend possible, a real solution to the dilemma of the desire for robust performance vs. lack of sufficient training data in vision research. This is possible because the inherent stochastic nature of reinforcement learning allows the system to search for a large proportion of the search space. As a by-product, experiences captured at lower stages can act as a rich source of training data that can then be used at later stages to obtain robust performance.

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 as evaluation, then it is not clear how the process should proceed

*This work was supported by ARPA/AFOSR grants F49620-93-1-0624 and F49620-95-0424. The contents of the paper do not necessarily reflect the position or the policy of the U.S. Goverment.

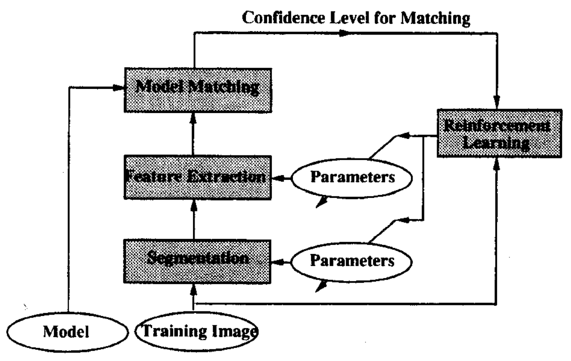


Figure 1. Reinforcement learning-based multi-stage system for object recognition.

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, and thus under many circumstances can ease overall demands on the memory and speed.

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

2 Reinforcement Learning

Reinforcement learning is a branch of artificial intelligence that studies computational approaches to learning from rewards and punishments (called reinforcement). In this paper, reinforcement corresponds to the confidence measure generated by the model matching (see Fig. 1). This type of learning has a wide variety of applications, ranging from modeling behavior learning in experimental psychology to building active vision systems.

A distinction can be made between non-associative and associative reinforcement learning. In the non-associative paradigm, reinforcement is the only information the system receives from its environment. This type of learning has been studied extensively in the fields of function optimization and learning automata theory. Whereas, in the associative paradigm, the system receives input information that indicates the state of its environment as well as reinforcement. In computer vision, this state information corresponds to current input image and our object recognition applications require us to take into account the changes appearing in the input images as a result of changing environmental conditions with time. The objective of the system is to make sequences of decisions to maximize the sum of future reinforcements (possibly discounted) over time.

An additional complication to reinforcement learning is the timing of reinforcement. In simple tasks, the system receives, after each decision, reinforcement indicating the goodness of that decision. Immediate reinforcement often occurs in function optimization tasks [6], and its application to computer vision can be found in [1]. *In most complex tasks, however, reinforcement is often temporally delayed because immediate reinforcement regarding the goodness of a decision is unavailable.* 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. These methods involve first determining the "optimal state-value function", V, which assigns to each state the expected total discounted reward obtained when an optimal policy is followed starting in that state, where a policy is a set of rules for selecting actions at each state. As in [7], one can define a closely related function, called *Q function*, that assigns to each state-action pair a value measuring the expected total discounted reward obtained when the given 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 the current action, r the resulting immediate reward, and y the resulting next state from taking a in x, then $Q(x,a) = R(x,a) + \gamma \sum_y P_{xy}(a)V(y)$, where $R(x,a) = E\{r|x,a\}$, $V(x) = \max_a Q(x,a)$, $P_{xy}(a)$ 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. Note that once the Q-function is known it is straightforward to determine the optimal policy. For any state x the optimal action is simply $\arg\max_a Q(x,a)$.

The method for learning the Q-function developed by Watkins [7], called the *Q-learning* algorithm, is based on maintaining an estimate $\hat{Q}$ of the Q-function and updating it so that equation defining $Q(x,a)$ above, with estimated values substituted for the unknown actual values, comes to be more nearly satisfied for each state-action pair encountered.

The advantage of the Q-learning algorithm is that when combined with sufficient exploration it can be guaranteed to eventually converge to an optimal policy [7]. The disadvantages, however, are that it is very slow to converge and may work poorly in problem domains that are non-Markovian. To overcome these weaknesses, Peng and Williams [6] have introduced the Q(λ) learning algorithm in which the current

```
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 $i$ using $\bar{a}$; $i_s$: resulting segmented image.
     (b) Update $\hat{Q}(i, \bar{a})$ according to Q($\lambda$) learning with $e' = \gamma\hat{V}(i_s) - \hat{Q}(i, \bar{a})$, $e = \gamma\hat{V}(i_s) - \hat{V}(i)$
     (c) Perform feature extraction with current value of $\bar{b}$ from $i_s$.
     (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 $\hat{Q}(i_s, \bar{b})$ and $\hat{Q}(i, \bar{a})$ according to Q($\lambda$) learning with $e' = r - \hat{Q}(i_s, \bar{b})$ and
         $e = r - \hat{V}(i_s)$
3. UNTIL terminating condition
```

Figure 2. Main steps of the delayed reinforcement learning algorithm for parameter adjustment for segmentation and feature extraction.

prediction error is used to correct previously experienced state-action pairs in addition to the current one. The parameter λ controls the proportion in which corrections are made. This is done by using the mechanism of the "activity" trace of state-action pair (x, a). It assigns a value to each experienced state-action pair with more recent ones having higher value. See [6] for details.

3 Reinforcement Learning for Object Recognition

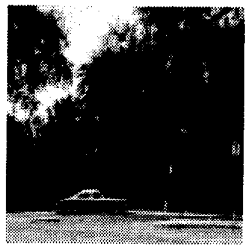
In the multi-stage system for model-based recognition described in Figure 1, there are unknown parameters for both the segmentation and feature extraction modules. The segmentation module is based on the "Phoenix" algorithm [4]. Phoenix uses region splitting based on histograms of color features and is critically dependent on system parameters *Hsmooth* and *Maxmin*. While *Hsmooth* is the width of the histogram smoothing window, *Maxmin* defines the peak-to-valley height ratio threshold. The learning system is to search 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 the same as those used in [3]. The resulting search space is about one thousand sample points.

The feature extraction module finds polygon approximation for borders of each of the regions obtained after image segmentation. The polygon approximation obtained using a split and merge technique, critically depends on neighborhood parameters, named M1 and M2, that affect maximum curvature estimations [8]. For the purpose of this paper only M2 is subject to adaptation.

Object recognition employs a cluster-structure matching algorithm [5] which is based on the clustering of translational and rotational transformation between the object and the model for recognizing 2-D and 3-D objects. 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. This confidence level is then used as a reinforcement signal to drive the algorithm.

The objective of the system is to autonomously find a set of segmentation and feature extraction parameters that achieve the maximum matching confidence for a given input image. It can be seen clearly that our model-based recognition system is a multi-stage decision process where the parameters *Hsmooth* and *Maxmin* are at the first stage of the process and the parameter M2 is at the second stage. Furthermore, 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(λ) learning algorithm with the confidence as reinforcement is used to adjust the parameters at both the first and second stages.

Let i be an input image to the segmentation module and $\bar{a}$ be an instance of segmentation parameters and $\bar{b}$ be an instance of feature extraction parameters. Also, let i_s be the segmented image resulting from applying $\bar{a}$ to image i. Then according to the Q-learning algorithm $Q(i, \bar{a})$ measures how good the instance $\bar{a}$ is when applied to image i. Likewise, $Q(i_s, \bar{b})$ measures the quality of extracted features when $\bar{b}$ is applied to the segmented image i_s. When the Q(λ)



(a)

(b)

Figure 3. A sample outdoor color image (Frame 1 of a 20 frame sequence) and (b) polygonal model of the car.

learning algorithm is applied to the parameters the value of $Q(i, \bar{a})$ will be corrected to look more like the value of the segmented image, $V(i_s) = \max_b Q(i_s, \bar{b})$, which will in turn be estimated according to the matching confidence.

Figure 2 shows the main steps of the algorithm described. The algorithm terminates when either the number of iterations has exceeded a prespecified value or the recognition confidence level has reached a given threshold. Note that in general there can be multiple objects in the images.

4 Experimental Validation

There are several representation schemes for the Q-function in the reinforcement learning paradigm. However, the goal here is to demonstrate the effect of learning for multi-stage recognition, a look-up table based representation suffices.

The two dimensions of the look-up table are the following: (1) input or segmented (feature-extracted) image, (2) action represented by a particular combination of system parameters. All the table entries are initialized to zero. The "activity" trace Tr is similarly indexed. The focus of the experiments is to demonstrate the feasibility of using learning for multi-stage recognition. Also, γ is set to 0.95 and λ is set to 0.3 for all the experiments reported here.

Figure 3(a) shows a sample of outdoor color images (120 by 120) obtained under varying environmental conditions. These images were collected approximately every 15 minutes over a ~ 2 and 1/2 hour period [3]. 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.

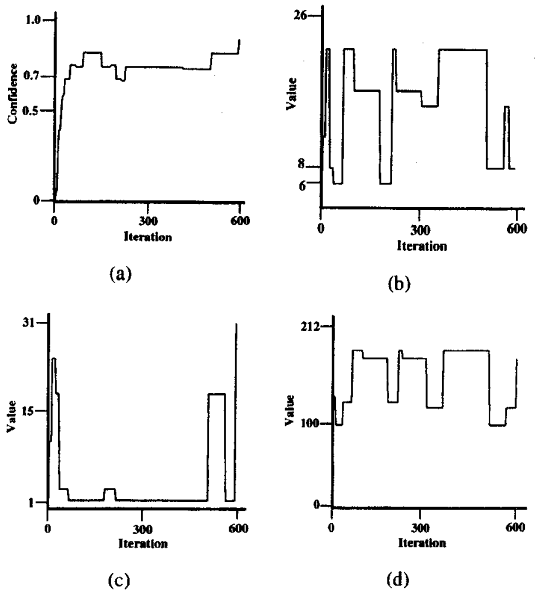


Figure 4. Experimental results (training) for the image in Fig. 5. (a) matching confidence level (b) paremeter M2 (c) parameter HSMOOTH (d) parameter MAXMIN.

Figure 3(b) shows the 2-D model of the car located in Figure 3(a). The dark squares in Figure 3(b) correspond to labels of the vertices in the polygonal approximation of the car. It is extracted manually in an interactive session from the first frame in the sequence and is used as the model in the cluster-structure matching algorithm [5].

Figure 4(a) shows how the confidence, averaged over 5 runs, changes over time for the image shown in Figure 3(a). It should be noted that over time the confidence shown in Figure 4(a) increases. At the end of the training phase the confidence of the match is over 0.9 on a scale which varies between 0 and 1. For acceptable recognition, the confidence of matching has to be greater than 0.75 in the experiments reported here.

Figures 4(b), 4(c), and 4(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 values.

To illustrate the results further, Figure 5 shows how the segmentation of the image improves over time during training. Figure 5(a) depicts the segmentation before applying the learning algorithm. Figures 5(b) and 5(c) depict the segmentation after 1/3 and 2/3 of total time (600 iterations) for

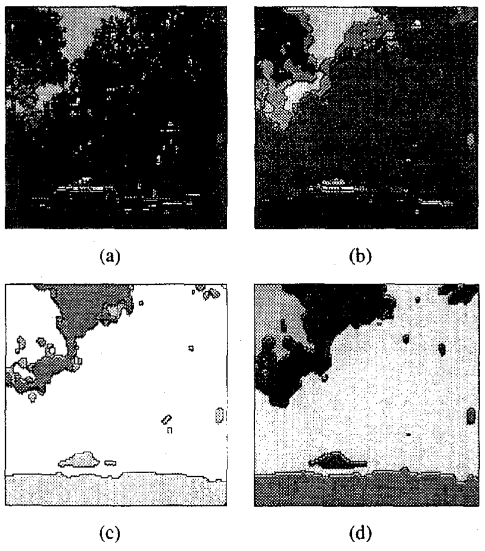


Figure 5. Improvement of the segmentation over time. (a) initial segmentation (b) segmentation at time step 200 (c) segmentation at time step 400 (d) segmentation at time step 600.

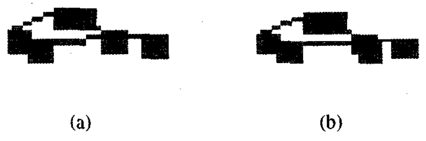


Figure 6. Polygonal approximation of the car. (a) default M2 parameter (b) learned M2 parameter.

training has elapsed, respectively. Figure 5(d) depicts the segmentation at the end of the training phase. It can be seen that the results improve considerably. While Figure 6(a) shows the extracted features (polygonal approximation of the car) using the default M2 parameter, Figure 6(b) shows the same feature using the learned M2 parameter that results in high matching confidence.

5 Conclusions

In the experiments we have used a look-up table to represent the Q function. However, look-up table representation may not be highly adequate in complex systems since search space is often too large to allocate entire memory. A potential solution to this problem is to first classify input images into representative groups using algorithms such as the K-Means algorithm, and then allocate memory only to these groups. In general, however, compact function representation schemes that can generalize across spaces must be sought.

The model-based 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. If vision systems could be designed in one-stage as a single black box, the "simple" reinforcement paradigm would have sufficed [1]. However, in reality vision systems have multiple stages for real-world tasks with parameters that need to be adjusted at each stage. Delayed reinforcement learning based approach presented here provides an elegant and effective solution to the problem of object recognition in multi-stage systems.

References

[1] J. Peng and B. Bhanu, Closed-Loop Object Recognition Using Reinforcement Learning, *Proc. ARPA Image Understanding Workshop*, Monterey, CA, (November 1994), pp. 777-780.

[2] P. Suetens, P. Fua and A. J. Hanson, Computational Strategies for Object Recognition, *ACM Computing Surveys 24(1)*, pp. 5-59, 1992.

[3] B. Bhanu and S. Lee, *Genetic Learning for Adaptive Image Segmentation.* Boston, MA: Kluwer Academic Publishers, (1994).

[4] K. Laws, *The PHOENIX Image Segmentation System: Description and Evaluation.* Tech. Rep. 289, SRI International, December (1982).

[5] B. Bhanu and J. Ming, Recognition of Occluded Objects: A Cluster-Structure Algorithm. *Pattern Recognition 20(2)* (1987) pp. 199-211.

[6] J. Peng and R. J. Williams, Incremental multi-step Q-learning, *Proc. 11th Int. Conf. on Machine Learning*, New Brunswick, (1994) pp. 226-232.

[7] C. J. C. H. Watkins, *Learning from delayed rewards.* Ph.D. Dissertation, King's College, UK (1989).

[8] T. Pavlidis, *Structural Pattern Recognition*, Springer-Verlag, Berlin-Heidelberg-New York, 1977.